BUFFER 9 FORMAT

The following is the format for Buffer 9 which holds all of the CPU data for DEBUG. The Buffer resides in 7 56 bit user registers.

Nybble → 13	12	11	10	9	8	7	6	5	4	3	2	1	0	Register Number ↓
1	U	STK2				H/D	PT	P	Q	G		ST		6
							N							5
							M							4
							B							3
							A							2
							C							1
I	D	SIZE		U	U	STK1				BKPT				0

Where :

U = Unused Nybbles
ID = Buffer Identity = 99 (HEX)
SIZE = Buffer Size = 07 (HEX) REGISTERS
H/D = CPU Mode HEX = 0 DEC ≠ 0
PT = Active Pointer P = 0 Q ≠ 0
P = Position Of Pointer P
Q = Position Of Pointer Q
G = G Register (1 byte)
ST = ST Register = System Flags 0-7
BKPT = Address Of Last Breakpoint Stop
STK1 = 1st Stack Value At Last Stop
STK2 = 2nd Stack Value At Last Stop
N, M, B, A, C = Main 5*56 Bit Accumulators

MARKS ROM HEADER

```
x084    B           082
        I           031
        _           020
        S           013
        K           008
        R           012
        A           001
        M           00D
x08C    MARKS 1B:   RTN             ; Performs no operation just titles ROM
```

INCREMENT M FUNCTION - M+1

```
x096                I       0B1
                    +       02B
                    M       00D
        M+1:        C=REG   S/M     ; Read M register
                    C=C+1   ALL     ; Increment it
                    REG=C   S/M     ; Put it back
x09C                RTN             ; Done
```

DECREMENT M FUNCTION - M-1

```
x09D                I       0B1
                    -       02D
                    M       00D
        M-1:        C=REG   S/M     ; Read M register
                    C=C-1   ALL     ; Decrement it
                    REG=C   S/M     ; Put it back
x0A3                RTN             ; Done
```

SUBROUTINE - DISPLAY REGISTER, POSITION & CONTENTS

DEPENDENCIES: GETNYB
OUTHEX
ROUTINES USED: CLLCDE
PCTOC

INPUT: N REGISTER

N								B	B	B	R	ST	ST

WHERE N = NYBBLE BEING EDITED
BBB = BASE ADDRESS OF BUFFER (HEADER)
R = REGISTER BEING EDITED
ST = AVAILABLE FOR STATUS

OUTPUT: DISPLAY IN FORMAT R:P P a b c, d, e f g
WHERE R = REGISTER NAME C,A,B,M or N
PP = POSITION IN DECIMAL OF EDITOR IN REGISTER
a..g = THREE NYBBLES EITHER SIDE OF NYBBLE BEING EDITED (d)

USES: DISPLAY & REGISTERS & BUFFER 9

```
x400  DSPRPC:  C=0      ALL          ; CLEAR OUTPUT STRING
               M=C
               LDI      030          ; LOAD COUNTER VALUE WITH 30 HEX
        DL1:   A=C      S&X          ; STARTING AT -3 FROM NYBBLE
               B=A      S&X          ; GET 4 NYBBLES AND ADD THEM
               GOSUB    GETNYB       ; TO THE OUTPUT STRING
               A=C      MS
               C=M
               A<>C     MS
               RCR      13
               M=C
               C<>B     S&X
               PT=1
               C=C-1    PT
               JNC-     DL1
               LDI      001          ; STARTING AT +1 FROM NYBBLE
        DL2:   A=C      S&X          ; GET 3 NYBBLES (+1,+2,+3) AND
               B=A      S&X          ; ADD THEM TO THE OUTPUT STRING
               GOSUB    GETNYB
               A=C      MS
               C=M
               A<>C     MS
               RCR      13
               M=C
               A<>B     S&X
               LDI      004
               A<>C     S&X
               C=C+1    S&X
               ?A#C     S&X
               JC-      DL2
               C=M                   ; FINALLY ADD A 7 INTO LS Nybble
               PT=0                  ; SO OUTPUT STRING IS FULLY FORMATTED
               RCR      13           ; READY FOR OUTPUTING BY OUTHEX
               LD@R     7
               M=C
               NC XQ    CLLCDE       ; CLEAR & ENABLE LCD
               NC XQ    PCTOC        ; GET PC CURRENT VALUE
               JNC+     MLC          ; JUMP AROUND THE DISPLAY TABLE
```

```
        HEX  083        ; DISPLAY CHARACTER TABLE      'C:'
        HEX  081        ; FOR REGISTER NAMES           'A:'
        HEX  082        ;                              'B:'
        HEX  08D        ;                              'M:'
        HEX  08E        ;                              'N:'
MLC:    A=C  M          ;PUT ADDRESS OF WORD BEFORE TABLE
        C=N             ;INTO A[M] AND GET POINTERS
        C=0  M          ;MOVE REGISTER NUMBER INTO C[M]
        RCR  13         ;AND ADD IT TO TABLE ADDRESS
        C=C+A M
        FETCH           ;GET THE DISPLAY CHARACTER
        WRIT 15(e) WRABCIR ;OUTPUT IT TO RIGHT OF LCD
        PT=13
        LDER 9
        A=C  MS         ;GET NYBBLE NUMBER AND ADD A
        C=N             ;'0' OR A '1' TO THE DISPLAY AS
        ?A<C MS         ;AS APPROPRIATE
        JC+GT10
        LDI  030        ;'0'
        WRIT 15(e) WRABCIR
        JNC+ DNU
GT10:   LDI  031        ;'1'
        WRIT 15(e) WRABCIR
        A<>C MS
        C=A-C MS        ;SUBTRACT 9
        C=C-1 MS        ;AND 1 TO GIVE CORRECT 2ND DIGIT VALUE
DNU:    LDI  003        ;THEN OUTPUT THE SECOND DIGIT
        RCR  13
        WRIT 15(e) WRABCIR
        LDI  020        ;' '
        WRIT 15(e) WRABLR ;TWO SPACES
        WRIT 15(e) WRABCIR
        GOSUB OUTHEX    ;OUTPUT 7 CHARACTER OUTPUT STRING
        RABCIR          ;AND PUT COMMAS AROUND THE
        RDABCIR         ;MIDDLE DIGIT TO MAKE IT LOOK GOOD
        RDABCIR
        RDABCIR
        RDABCIR
        C<>ST
        SF   6          ;INDICATE ','
        SF   7
        C<>ST
        WRABCIR
        RDABCIL
        C<>ST
        SF   6          ;INDICATE ','
        SF   7
        C<>ST
        WRABCIL
        RDABC4L         ;RESTORE DISPLAY
        RTN             ;END WITH LCD ENABLED STILL
```

x467

SUBROUTINE - REGISTER EDITOR - RED.

```
DEPENDENCIES:   MK9          GETHXA          MENU
                DSPRPC       GETNYB
ROUTINES USED:  ENCPØØ       OFSHFT          TOGSHF
                ANNOUT
INPUT:          BUFFER 9 IN CPU FORMAT
OUTPUT:         BUFFER 9 WITH REGISTERS 1..5 CORRESPONDING TO C,A,B,M,N
                EDITED ACCORDING TO USER COMMANDS
USES:           PROBABLY EVERYREGISTER DEPENDING ON ROUTINES USED!

x47Ø  RED:      GOSUB   MK9          ;FIND BUFFER 9
                C=Ø     ALL          ;SET UP N REGISTER TO INDICATE
                A<>C    S&X          ;   START OF BUFFER
                RCR     11           ;   REGISTER BEING EDITED = 1 (C)
                C=C+1   XS           ;   NYBBLE BEING EDITED = Ø
                N=C
MLOOP:          GOSUB   DSPRPC       ;DISPLAY REGISTER,POSITION AND CONTENTS
MKEYL:          GOSUB   GETHXA       ;GET HEXADECIMAL/SPECIAL KEYPRESS
                FS?     2            ;IF SPECIAL KEY PRESSED SET CARRY
                JC+     SPEC         ;  AND JUMP TO SPEC
                A=Ø     S&X          ;OTHERWISE SET THE KEY VALUE
                GOSUB   GETNYB       ;AND PUT IT IN THE BUFFER
                C<>B    ALL          ;AT THE CURRENT POSITION. THEN
                RCR     1            ;JUMP OFF TO MONR - MOVE POINTER
                C<>B    MS           ;ONE NYBBLE RIGHT
                RCR     13
                A=C     MS
                C<>B    ALL
                RCR     13
                A=A-1   MS
                JNC-Ø2
                WRITE   DATA
                JNC+    MONR
SPEC:           C=C-1   S&X   [ON]   ;IF NOT [ON] PRESSED JUMP TO SPEC2
                C=C-1   S&X
                JNC+    SPEC2
                NC XQ   ENCPØØ       ;OTHERWISE TURN OFF SHFT FLAG AND
                NC XQ   OFSHFT       ;ANNUNCIATOR AND GO TO MAIN MENU
                GOTO    MENU
SPEC2:          C=C-1   S&X   [SHFT] ;IF NOT [SHFT] PRESSED JUMP TO SPEC3
                C=C-1   S&X
                JNC+    SPEC3
                NC XQ   TOGSHF       ;OTHERWISE TOGGLE THE SHFT FLAG AND
                NC XQ   ANNOUT       ;ANNUNCIATOR
JMKEYL:         JNC-    MKEYL        ;JUMP BACK TO MAIN KEY LOOP (OF RED)
SPEC3:          C=C-1   S&X   [USER] ;IF NOT [USER] PRESSED JUMP TO SPEC4
                JNC+    SPEC4
                NC XQ   ENCPØØ       ;OTHERWISE LOOK AT USER FLAG STATUS
                READ    14(d)        ;IF SHIFT IS NOT SET MOVE LEFT A
                RCR     2            ;NYBBLE (MONL) ELSE MOVE LEFT A
                ST=C                 ;REGISTER (MORL)
                FS?     Ø
                JNC+    MONL
```

```
MORL:   PT=2                    ;MOVE ONE REGISTER LEFT
        LD@R    5
        A<>C    XS
        C=N
        ?A#C    XS              ;IF REGISTER NUMBER IS NOT 5
        JNC+    NMR1
        C=C+1   XS              ;THEN INCREMENT REGISTER NUMBER
        N=C
NMR1:   NC XQ   TOGSHF          ;ELSE TOGGLE (OFF) SHIFT FLAG AND
        NC XQ   ANNOUT          ;  ANNUNCIATOR
JMLOOP: JNC-    MLOOP           ;JUMP TO MAIN LOOP & REDRAW DISPLAY
SPEC4:  C=C-1   S&X   PRGM      ;IF NOT PRGM PRESSED IGNORE KEY
        JNC-    JMKEYL          ;AND JUMP BACK TO MAIN KEY LOOP
        NC XQ   ENCPØØ          ;OTHERWISE INSPECT STATUS OF SHIFT
        READ    14(d)           ;FLAG AND MOVE ONE NYBBLE RIGHT
        RCR     2               ;IF IT IS OFF AND ONE REGISTER
        ST=C                    ;RIGHT IF IT IS ON
        ?FSET   Ø
        JNC+    MONR
MORR:   C=N                     ;MOVE ONE REGISTER RIGHT
        C=C-1   XS              ;DECREMENT REGISTER VALUE
        ?C#Ø    XS              ;IF NOT ZERO GOTO NMR1
        JNC-    NMR1
        N=C                     ;OTHERWISE SAVE NEW VALUE
        JNC-    NMR1            ;AND JUMP TO NMR1
MONR:   C=N                     ;MOVE ONE NYBBLE RIGHT
JMLP4:  C=C-1   MS              ;DECREMENT NYBBLE VALUE
        JNC+    JMLP3           ;IF NO CARRY THEN SAVE VALUE & CONTINUE
        C=C-1   XS              ;AT JMLP3 OTHERWISE DECREMENT
        ?C#Ø    XS              ;REGISTER VALUE. IF THAT BECOMES ZERO
        JNC+    TOFAR           ;THEN JUMP TO TOFAR
        PT=13                   ;OTHERWISE SET NYBBLE POINTER TO
        LD@R    D               ;13 AND SAVE STATUS
JMLP3:  N=C                     ;SAVE POINTERS
        JNC-    JMLOOP          ;REJOIN MAIN KEY LOOP & REFRESH LCD
TOFAR:  C=C+1   XS              ;IF TRIED TO GO TOOFAR RIGHT PUT
        C=Ø     MS              ;EVERYTHING BACK TO NORMAL
        JNC-    JMLP3           ;AND REJOIN MAIN LOOP AS IF NOTHING HAPPENED
MONL:   PT=13                   ;MOVE NYBBLE LEFT
        LD@R    E
        A=C     MS
        C=N
        C=C+1   MS              ;INCREMENT NYBBLE VALUE
        ?A#C    MS              ;IF NOT 14 THEN
        JC-     JMLP3           ;  JUMP TO JMLP3
        C=C+1   XS              ;OTHERWISE INCREMENT REGISTER VALUE
        A<>C    XS
        PT=2
        LD@R    6
        A<>C    XS
        ?A#C    XS              ;IF REGISTER VALUE BECOMES 6 THEN
        JNC+    TOFAR2          ;  JUMP TO TOFAR2
        C=Ø     MS              ;OTHERWISE SET NYBBLE VALUE TO
        JNC-    JMLP3           ;ZERO AND RETURN TO MAIN LOOP
TOFAR2: C=C-1   XS              ;IF GONE TOO FAR LEFT DECREMENT
X4E4    JNC-    JMLP4           ;REGISTER & NYBBLE VALUES & GOTO JMLP4
```

Extended Catalogue.

For the hex values given the routines must start at x1ØØ, x5ØØ, x9ØØ or xDØØ where x is the ROM Page number. Otherwise the values on the lines marked with an [R] must be changed.

```
D5ØØ            ØØ94    T
   1            ØØØ1    A
   2            ØØØ3    C
   3            ØØ18    X
   4  XCAT:     ØØØØ    NOP                          ; Entry point - non programmable
   5            3C1     *
   6            ØØBØ    NC XQ  CLLCDE [2CFØ]         ; Clear & enable LCD
   7            3DD     *
   8            ØØ1C    NC XQ  MESSL [Ø7EF]          ; Output message
   9            ØØ18    "X                           ;    "XCAT␣(Ø-F)␣␣"
   A            ØØØ3     C
   B            ØØØ1     A
   C            ØØ14     T
   D            ØØ2Ø     <SP>
   E            ØØ28     (
   F            ØØ3Ø     Ø
  1Ø            ØØ2D     -
   1            ØØØ6     F
   2            ØØ29     )
   3            ØØ2Ø     <SP>
   4            21F      —
   5            149      *
   6            ØØ24    NC XQ  ENCPØØ [Ø952]         ; Enable Chip ØØ
   7            26Ø     SETHEX                       ; Set HEX Mode
   8            379     *
   9            ØØ3C    NC XQ  GOSUB
   A  [R]       14A     DEF    GETHEX                ; Get Hexadecimal Key Value
   B            2ØC     ?FS    2                     ; If a 'SPECIAL KEY' is pressed
   C            36Ø     CRTN                         ;    then exit from routine
   D            358     ST=C                         ; Store HEX value in ST bits Ø-7
   E            1Ø6     A=C    X      (S&X)          ; Save value in A[X] for testing
   F            13Ø     LDI
  2Ø            ØØ5     HEX ØØ5                      ; Test against ØØ5
   1            3Ø6     ?A<C   X      (S&X)          ; If CAT<5 then do
   2            211     *                            ;    ordinary catalogue
   3            ØØ2F    C GO ØB84                    ;    routine
   4            MØ      ABC=Ø                        ; Otherwise CAT 2 from PAGE specified
   5            ØEØ     PT=Q                         ; Count number of functions in B[X]
   6            15C     PT=6
   7            15Ø     LC  5                        ; Test value of 5 in ADDR field
   8            ØAE     A<>C   ALL                   ;    of the A register
   9            398     C=ST                         ; Get PAGE number from LSD of ST
   A            13C     RCR    B                     ; Put into C[6], MSD of ADDR
   B            ØIØ     LC  Ø                        ; Add on ØØ1 to point to
   C            Ø1Ø     LC  Ø                        ; number of functions in page
   D            Ø5Ø     LC  1                        ;    i.e. PØØ1
```

```
D52D
D52E                33ϕ   RDROM                  ; Look to see if this page is empty
   F                2E6   ?C#ϕ   X   (S&X)       ; If it is then
  3ϕ                381   *                      ;     DISPLAY "NONEXISTANT" & END
   1                ϕϕA   NC GO  ϕ2Eϕ (ERRNE)
   2                15C   PT=6                   ; Set PTR for test & decrementing of page no
   3   Part 1 :     362   ?A#C  PT  (@R)         ; If we have added up all ROMs down
   4                ϕ43   JNC +ϕ8  Part 2        ; to page 5 then ENDLOOP & enter CAT fnc
   5                262   C=C-1  PT              ; Down to next ROM page
   C                33ϕ   RDROM                  ; Get number of functions in C[X]
   7                ϕ66   A<>B   X   (S&X)       ; Put COUNTER TOTAL in A
   8                2ϕ6   C=C+A  X   (S&X)       ; Add the number of functions in this page
   9                1ϕ6   A=C    X   (S&X)       ; Store new total in A
   A                ϕ66   A<>B   X   (S&X)       ; Restore total in B and restore A[X]
   B                3C3   JNC - ϕ8  Part 1       ; Do again for next page
   C   Part 2 :     238   READ  8 (P)            ; Get ALPHA top register / CAT
   D                ϕ9C   PT=5                   ;                working register
   E                ϕAϕ   PT=P
   F                15C   PT=6
  4ϕ                ϕ52   C=ϕ   P-Q              ; Clear  C[13:6]
   1                2DC   PT=13
   2                ϕ9ϕ   LC  2                  ; Set CAT number to ROMS 2
   3                ϕEE   C<>B  ALL              ; Get COUNTER TOTAL in C[X]
   4                ϕ7C   RCR  4                 ;                      C[12:1ϕ]
   5                ϕEE   C<>B  ALL              ;                      B[12:1ϕ]
   6                ϕ6E   A<>B  ALL              ;                      C[12:1ϕ]
   7                2ϕE   C=C+A  ALL             ; Mask to give  C= 2c cc ϕϕ ϕϕmmmmqq
   8                231   *
   9                ϕ2E   NC GO  ϕBBC            ; Jump into CAT where C is just about
                                                 ; to be put into P register
```

GET HEXADECIMAL KEY PRESS

```
D54A   GETHEX:      39C   PT=ϕ
   B                2ϕ4   CF  2
   C                379   *
   D                ϕ3C   NC XQ  GOSUB
   E   [R]          161   DEF   KEYDB           ; Debounce keyboard & Read A key
   F                ϕ26   B=ϕ   X   (S&X)
  5ϕ                379   *
  51                ϕ3C   NC XQ  GOSUB
   2   [R]          16C   DEF   INIT            ; Set up key table address
   3   GHXL1:       23A   C=C+1   M
   4                33ϕ   RDROM
   5                2E6   ?C#ϕ   X  (S&X)       ; End of table reached therefore
   6                3A3   JNC -ϕC  GETHEX        ; Invalid key so try again - not found in table
   7                366   ?C#A    X  (S&X)
   8                ϕ3B   JNC+ϕ7   FOUND         ; key in table => FOUND
   9                ϕE6   B<>C    X  (S&X)
   A                222   C=C+1   PT  (@R)       ; Increment key number
   B                ϕ13   JNC+ϕ2   GHXL2         ; If no carry (HEX mode) don't set flag 2
   C                2ϕ8   SF  2                  ; Set to indicate special key
   D   GHXL2:       ϕE6   B<>C    X  (S&X)
   E                3AB   JNC-ϕB   GHXL1         ; Compare with next entry in table
   F   FOUND:       ϕC6   C=B     X   (S&X)      ; Return the key number (from table)
  6ϕ                3Eϕ   RTN                    ; in both C[X] and B[X]
```

Debounce keyboard & Read key.

```
D561  KEYDB:   3CC   ?KEY                ; If no key is down
2              Φ23   JNC+Φ4  GETKEY      ;    then wait for one in GETKEY
3     KEYSUP:  3C8   CLRKEY              ; Else wait for the key
4              3CC   ?KEY                ;    that is pressed down
5              3F7   JC-Φ2   KEYSUP      ;    to be released
6     GETKEY:  3CC   ?KEY                ; Now no keys are down get the
7              3FB   JNC-Φ1  GETKEY      ;    next one pressed
8              22Φ   C=KEY               ; Put its value in  C[4:3]
9              Φ3C   RCR     3           ;                   C[1:Φ]
A              Φ56   C=Φ     XS  (XS)    ;                   C[2:Φ]
B              1Φ6   A=C     X   (s&x)   ;                   A[2:Φ] :e A[X]
```

Hexadecimal keyboard Look Up Table.

```
D56C  INIT:    3EΦ   RTN                 ; End of KEYDB & Start of keyTable
D     HEXTBL:  Φ37                       ; key  Φ
E              Φ36                       ;      1
F              Φ76                       ;      2
7Φ             Φ86                       ;      3
1              Φ35                       ;      4
2              Φ75                       ;      5
3              Φ85                       ;      6
4              Φ34                       ;      7
5              Φ74                       ;      8
6              Φ84                       ;      9
7              Φ1Φ                       ;      A
8              Φ3Φ                       ;      B
9              Φ7Φ                       ;      C
A              Φ8Φ                       ;      D
B              ΦCΦ                       ;      E
C              Φ11                       ;      F
D              ΦC3                       ;      ←   (DELETE)   (Φ)
E              Φ18                       ;      ON             (1)
F              Φ87                       ;      R/S            (2)
```

For relocatable values the routines must all lie in the same 1K ROM Page. The value specified in the DEF statements are the lowest 10 bits of the labels address.

```
D58Φ  EXTRAS:  Φ12                       ;      SHIFT          (3)
1              ΦC6                       ;      USER           (4)
2              ΦC5                       ;      PRGM           (5)
3              Φ13                       ;      ENTER↑         (6)
D584           ΦΦΦ   NOP                 ; End Of Table Marker
```

SUBROUTINE - GET HEX KEY WITH TIMEOUT - TKEYH.

DEPENDENCIES: INIT (HEX KEYBOARD LOOK-UP TABLE) @x56C
ROUTINES USED: NONE
INPUT: NONE
OUTPUT: RETURNS DIRECTLY WITH KEY VALUE IN B and C [S&X]
SKIPS A LINE & RETURNS IF INVALID KEY OR TIMED-OUT
USES: ACTIVE PT ⇒ SETS IT TO ϕ
C [KEY], [S&X], [M] i.e. ALL
B [S&X]
A [S&X]

```
x58C  TKEYH:  CLRKEY              ; Clear the keydown flag
              PT=ϕ
              LDI
              HEX 3FF             ; C[S&X] holds time-out constant
        L1:   C=C-1  S&X          ; Decrement time-out constant
              JC+ϕ4  L2           ; If carry then timed-out, goto L2
              ?KEY                ; Check for a key down
              JNC-ϕ3 L1           ; If no key down carry on waiting
              JNC+ϕ4 PARSEH       ;  otherwise try to parse key
        L2:   POP                 ; Time-out / Invalid key exit
              C=C+1  M            ;  Skips line after call to TKEYH
              GOTO ADR
  PARSEH:     C=KEY               ; Get key code
              RCR  3              ; Put it in C[S&X]
              C=ϕ  XS
              A=C  X&S            ;   and A [S&X]
              B=ϕ  X&S            ; Clear HEX value counter
              GOSUB D56C (INIT)   ; Get start of key table in C[ADDR]
  GXL1:       C=C+1  M            ; Point to next word in table
              RDROM               ; Get the word
              ?C#ϕ   X&S          ; If zero then we have run off the end
              JNC-ϕE  L2          ;   so Invalid key exit
              ?A#C   S&X          ; If keydown matches table then
              JNC+ϕ6 FOUND        ;   we have found the right one
              B<>C   S&X          ; If not then increment HEX value counter
              C=C+1  PT
              JC-13  L2           ; If value > FHEX then Invalid key
              B<>C   S&X
              JNC-ϕA GXL1         ; Otherwise try next table entry
  FOUND:      C=B    S&X          ; Duplicate HEX value in C[S&X]
                                  ;   WAIT for key to come up
              CLRKEY
              ?KEY
              JC-ϕ2
              RTN                 ; END
```

SUBROUTINE - MAIN STATUS DISPLAY - MAIND.

DEPENDENCIES: OUTHEX @
ROUTINES USED: MESSL @ CLLCDE@ GOSUB
INPUT: N REGISTER HOLDS STATUS INFORMATION (BUF9 FORMAT)
FLAG 7 INDICATES DISPLAY PT OR '_'
FLAG 9 " " P OR '_'
FLAG 11 " " Q OR '_'
OUTPUT: DISPLAY SHOWS : PT = r P = p Q = q
N REGISTER UNCHANGED

```
x5BØ  MAIND:  NC XQ  CLLCDE      ; Clear & enable LCD
              NC XQ  MESSL       ; Output message  PT= onto right of LCD
               ØIØ   P
               Ø14   T
               23D   =
              ?FS   7            ; If flag 7 is set display an underscore
              JC+ØB  US1         ;   rather than P or Q
               C=N               ; Get status register
               RCR 4             ; Put the P/Q nybble in C[XS]
              ?C#Ø   XS          ; Set carry if non zero  i.e if 'Q'
              JNC+Ø4  ACP        ; If not Q, display a P
               LDI               ;  otherwise display Q
               HEX  Ø11  'Q'
              JNC+Ø6  CONT1      ; Go & write digit to LCD
ACP:          LDI                ; Display P
               HEX  ØIØ  'P'
              JNC+Ø3  CONT1
US1:          LDI                ; Display '_' underscore
               HEX  ØIF  '_'
CONT1:        WRABCIR            ; Output digit
              NC XQ  MESSL       ; Output message ␣P= onto right of LCD
               Ø2Ø   ␣
               ØIØ   P
               23D   =
              ?FS   9            ; If flag 9 is set display an underscore
              JC+ØB  US2
              PT=Ø               ;  otherwise get the value of P
              C=N                ;  from the status register and indicate
              RCR  4             ;  to 'outhex' routine that here is
              LC   1             ;        1 digit
              LC   Ø             ;        0 underscores
              M=C
              NC XQ  GOSUB
              DEF  OUTHEX        ; Output P & continue below
              JNC+Ø4  CONT2
US2:          LDI                ; Output underscore '_'
               HEX  ØIF  '_'
              WRABCIR
CONT2:        NC XQ  MESSL       ; Output message ␣Q= to RHS of LCD
               Ø2Ø   ␣
               Ø11   Q
               23D   =
              ?FS   11           ; If flag 11 set display underscore
              JC+ØB  US3
```

```
                    PT=Ø                ; Otherwise get value of Q
                    C=N                 ; out of status information and
                    RCR 3               ; indicate to OUTHEX
                    LC  1               ;       1 digit
                    LC  Ø               ;       0 underscores
                    M=C
                    NC XQ GOSUB
                    DEF OUTHEX          ; Display value of Q
                    RTN                 ; End
US3:                LDI                 ; Display final '_'
                    HEX ØIF '_'
                    WRABC1R
xSEE                RTN                 ; and End
```

SUBROUTINE - ALTERNATIVE STATUS DISPLAY - ALTD.

DEPENDENCIES: OUTHEX@
ROUTINES USED: MESSL@ CLLCDE@ GOSUB
INPUT: N REGISTER HOLDS STATUS INFORMATION (RUFUS FORMAT)
FLAG 6 INDICATES DISPLAY MSN ST OR '_'
FLAG 5 " " LSN ST OR '_'
FLAG 4 " " MSN G OR '_'
FLAG 3 " " LSN G OR '_'
OUTPUT: DISPLAY SHOWS: S T = a b G = c d (H D)
N REGISTER UNCHANGED
FLAG 10 (ALT DISPLAY) SET

```
xSEF    ALTD:       NC XQ CLLCDE        ; Clear & enable LCD
                    NC XQ MESSL         ; Output ST= to RHS of LCD
                    Ø13  S
                    Ø14  T
                    23D  =
                    ?FS  6              ; If flag 6 is set display underscore
                    JC+ØA US4
                    C=N                 ; rather than the most significant nybble
                    PT=Ø                ; of ST
                    LC  1               ; 1 digit
                    LC  Ø               ; Ø underscores
                    M=C
                    NC XQ GOSUB
                    DEF OUTHEX          ; Display MSN of ST
                    JNC+Ø4 CONT3
US4:                LDI                 ; Display underscore
                    HEX ØIF '_'
                    WRABC1R
CONT3:              ?FS  5              ; If flag 5 is set display underscore
                    JC+ØB US5
                    C=N                 ; rather than the least significant nybble
                    RCR 13              ; of ST
                    PT= Ø
                    LC  1               ; 1 digit
                    LC  Ø               ; Ø underscores
                    M=C
                    NC XQ GOSUB
                    DEF OUTHEX          ; Display LSN of ST
                    JNC+Ø4 CONT4
```

```
US5:    LDI                 ; Display underscore
        HEX  01F  '_'
        WRABCIR
CONT4:  NC  XQ  MESSL       ; Output ␣G= to RHS of LCD
        020  ␣
        007  G
        23D  =
        ?FS  4              ; If flag 4 is set display underscore
        JC+0B  US6
        C=N                 ; rather than the MSN of G
        RCR  2
        PT=  0
        LC   1
        LC   0
        M=C
        NC  XQ  GOSUB
        DEF  OUTHEX
        JNC+04  CONT5
US6:                        ; Display underscore
        LDI  01F  '_'
        WRABCIR
CONT5:  ?FS  3              ; If flag 3 is set display underscore
        JC+0B  US7
        C=N                 ; rather than the LSN of G
        RCR  1
        PT=  0
        LC   1
        LC   0
        M=C
        NC  XQ  GOSUB
        DEF  OUTHEX
        JNC+04  CONT6
US7:    LDI                 ; Display underscore
        HEX  01F  '_'
        WRABCIR
CONT6:  SF   10             ; Indicate we are in the ACT display set
        C=N
        PT=  7
        ?C#0  @R            ; Get information relating to HEX/DEC mode
        JC+06  ALD
        NC  XQ  MESSL       ; If zero then HEX so display ␣H
        020  ␣
        208  H
        RTN                 ; End
ALD:    NC  XQ  MESSL       ; If non zero then DEC so display ␣D
        020  ␣
        204  D
        RTN                 ; End
```

x644

SUBROUTINE - GET KEY WITH TIMEOUT - TKEY.

```
                DEPENDENCIES :   NONE
                ROUTINES USED :  NONE
                INPUT :          NONE
                OUTPUT:          RETURNS DIRECTLY WITH KEYCODE IN A AND C[S&X]
                                 SKIPS A LINE & RETURNS IF TIMED-OUT
                USES:            ACTIVE PT ⇒ SETS IT TO Ø
                                 C[KEY],[M],[S&X] ;i.e. ALL A[S&X]

x645    TKEY:    CLRKEY          ; Clear the keydown flag
                 R=Ø
                 LDI
                 HEX 3FF         ; C[S&X] hold the time-out constant
        L4:      C=C-1 S&X       ; Decrement time-out constant
                 JC+Ø4 SKPNXT    ; IF carry then timed-out, goto SKPNXT
                 ?KEY            ; Check for a key down
                 JNC-Ø3 L4       ; IF no key down carry on waiting
                 JNC+Ø4 GOTKEY   ;   otherwise got a key so exit
        SKPNXT:  POP             ; Time-out return skips line after call TKEY
                 C=C+1 M
                 GOTO ADDR
        GOTKEY:  C=KEY           ; Key pressed so gets its value
                 RCR 3           ; and set up keycode in C[S&X]
                 C=Ø XS          ;                   and A[S&X]
                 A=C S&X
x655             RTN
```

SUBROUTINE - GET KEY & WAIT FOR IT TO COME UP - KEYDBU.

```
                DEPENDENCIES:    NONE
                ROUTINES USED:   NONE
                INPUT :          NONE
                OUTPUT :         A and C[S&X] contain keycode (MSN=Ø)

x656    KEYDBU:  ?KEY            ; If no key down skip this bit
                 JNC+Ø4          ; Otherwise
                 CLRKEY          ;   wait for key to come up
                 ?KEY
                 JC-Ø2
                 ?KEY            ; Wait for new key to go down
                 JNC-Ø1
                 C=KEY           ; key down - get its value in
                 RCR 3
                 C=Ø XS          ;       C[S&X]
                 A=C S&X         ;       A[S&X]
                 CLRKEY          ; Wait for it to come up
                 ?KEY
                 JC-Ø2
x664             RTN             ; When it is up then exit

; PATCH TO ALLOW STEDs CLEAR FUNCTION TO NOT DESTROY [12:8] OF TOP
; OF BUFFER 9 WHICH HAS NOW BEEN USED TO HOLD PART OF THE STACK

x8FF    CLR:     C=N             ; GET TOP REGISTER (STATUS)
                 PT=7            ; CLEAR [7:Ø] ONLY
                 C=Ø WPT
                 JNC+39 JLOOP    ; RETURN & RESTORE STATUS
```

SUBROUTINE - ASK USER IF HE IS SURE.

DEPENDENCIES: KEYDBU
ROUTINES USED: MESSL
CLLCDE
INPUT: NONE
OUTPUT: DISPLAYS PROMPT 'SURE (Y/N) ?' ON LCD
ACCEPTS Y AS TRUE AND ANY OTHER KEY AS FALSE
IF FALSE RETURNS IMMEDIATELY
IF TRUE SKIPS THE LINE AFTER THE CALL TO SURE
USES: C [ALL] - C[S&X] RETURNED CONTAINING Ø16
A [S&X] - RETURNS WITH KEYCODE OF KEY PRESSED
LCD ENABLED & CONTAINS MESSAGE ON EXIT

```
x665    SURE:   NC  XQ  CLLCDE      ;CLEAR & ENABLE LCD
                NC  XQ  MESSL       ;OUTPUT PROMPT
        ALPHA:: SURE␣(Y/N)␣?
                GOSUB   KEYDBU      ;GET A NEW KEY & WAIT FOR IT TO COME UP
                LDI     Ø16         ;COMPARE IT WITH Y KEY
                ?A#C    S&X         ;SET CARRY IF NOT Y KEY
                C RTN               ;  AND RETURN
                POP                 ;OTHERWISE IF IT IS Y SKIP THE
                C=C+1   M           ;NEXT LINE AND RETURN
x67E            GOTO    ADR
```

SUBROUTINE - DISABLE BREAKPOINTS - DABKPT

DEPENDENCIES: NONE EXCEPT TABLES RBØ..RB9 AND BØ..B9
ROUTINES USED: PCTOC
INPUT: NONE
OUTPUT: REPLACES ALL 'NC XQ BKPT' INSTRUCTIONS IN THE CODE UNDER TEST WITH THE ORIGINAL TWO WORDS THAT WERE THERE BEFORE THE BREAKPOINT WENT IN. THE ADDRESSES FOR THESE ARE HELD IN THE BREAKPOINT TABLE BØ..B9 AND THE CODE OVERWRITTEN IN THE TABLE RBØ..RB9 (FIRST TWO WORDS OF EACH ENTRY - KNOWN AS OVR1 & OVR2).
USES: A, B, C, G, N, P, Q
M IS UNUSED

```
x67F    DABKPT: NC  XQ  PCTOC       ;GET ROM PAGE NUMBER IN C[6]
                PT=5
                LC  F
                LC  1
                LC  2               ; xF12 IS ADDRESS OF B9 LOWER WORD
        DLOOP:  RDROM               ;GET HIGH ADDRESS BYTE (FROM LOW WORD)
                ?C#Ø    S&X         ;IF ZERO THEN BREAKPOINT IS NOT ACTIVE
                JNC+IF  DNEXT       ;SO CHECK NEXT
                N=C                 ;PUT ADDRESS AND WORD IN N
                PT=Ø
                G=C                 ;PUT WORD IN G
                C=C+1   M           ;INCREMENT ADDRESS TO GET (HIGH WORD)
                RDROM               ;LOW ADDRESS BYTE
```

```
                PT=2            ;POSITION POINTER TO PUT BACK HIGH ADDR
                C=G             ;NOW C[3:Ø] HOLDS BREAKPOINTS ADDR
                RCR  11         ;NOW IN C[ADDR]
                B<>C ALL        ;NOW IN B[ADDR]
                                ; AND IS READY TO PUT BACK OUR1 & OUR2
                C=N             ;C[6:3] HOLDS LOOP VALUE
                A=C  ALL        ;A[6:3] HOLDS LOOP VALUE DUPLICATE
                RCR  3          ;LOOP IN C[3:Ø]
                PT=1            ;DOUBLE LOWER BYTE OF ADDRESS WHILST
                C=C+C  WPT      ;KEEPING HIGH BYTE (DF) IN TACT
                C=C+1  WPT      ;ADD 4
                C=C+1  WPT
                C=C+1  WPT
                C=C+1  WPT
                C=C+1  PT       ;ADD 1Ø HEX
                RCR  11         ;AND PUT INTO C[ADDR]. THIS IS RBn
                RDROM           ;GET WORD OUR1 FROM RBn
                B<>C M          ;GET BKPT ADDRESS
                WROM            ;SAVE WORD OUR1 TO BKPTn
                C=C+1  M        ;INCREMENT BKPT ADDRESS
                B<>C M
                C=C+1  M        ;INCREMENT RBn ADDRESS
                RDROM           ;GET WORD OUR2 FROM RBn+1
                B<>C M
                WROM            ;WRITE WORD OUR2 TO BKPTn+1
                C<>A ALL        ;RESTORE BACKUP LOOP VALUE TO C[ADDR]
DNEXT:          PT=Q
                PT=4
                PT=P
                PT=3
                C=C-1  PQ       ;DECREMENT LOW BYTE OF LOOP
                C RTN           ;IF THERE IS A CARRY THEN ALL DONE: END
                C=C-1  PQ       ;ELSE DECREMENT AGAIN TO GIVE NEXT
x6AD            JNC-28 DLOOP    ;Bn ADDRESS AND REPEAT PROCESS
```

ENABLE ALL BREAKPOINTS - ENBKPT.

```
DEPENDENCIES:   WRC1    BKPT
ROUTINES USED:  PCTOC
INPUT:          NONE
OUTPUT:         ALL BREAKPOINTS WHOSE ADDRESSES APPEAR IN THE TABLE
                BØ..B9 ARE ENABLED i.e. 'NC XQ BKPT' OVERWRITES
                THE CODE THAT WAS THERE ORIGINALLY.
USES:           A,C,M,N,G,P,Q   FLAGS Ø & 1 (RETURNED CLEAR)

x6AE  ENBKPT:   NC  XQ  PCTOC   ;PUT ROM PAGE NUMBER IN C[6]
                PT=5
                LC  F
                LC  1
                LC  2           ; xF12 IS ADDRESS OF B9
        ELOOP:  RDROM           ;GET HIGH ADDRESS BYTE
                ?C#Ø    S&X     ;IF ZERO THEN BREAKPOINT IS NOT
                JNC+18  ENEXT   ;ACTIVE SO CHECK NEXT
```

```
                    N=C             ;PUT LOOP IN N[6:3]
                    PT=Ø
                    G=C             ;MSBYTE IN G
                    C=C+1 M
                    RDROM           ;LSBYTE IN C[2:Ø]
                    PT=2
                    C=G             ;C[3:Ø] HOLDS BKPT ADDRESS
                    RCR 11          ;C[ADDR] HOLDS BKPT ADDRESS
                    A<>C ALL        ;A[ADDR] HOLDS BKPT ADDRESS FOR WRC1
                    NC XQ PCTOC     ;PUT PAGE NUMBER INTO C[6]
                    PT=5
                    LC B
                    LC F
                    LC 7            ;ADDRESS OF 'BKPT' IS xBF7
                    RCR 3           ;PUT INTO C[3:Ø]
                    M=C             ;         M[3:Ø]
                    CF Ø            ;      XQ
                    CF 1            ;      NC
                    GOSUB WRC1      ;WRITES NC XQ 'BKPT' @ BKPT
                    C=N             ;C[6:3] HOLDS LOOP VALUE
ENEXT:              PT=Q
                    PT=4
                    PT=P
                    PT=3
                    C=C-1 P-Q       ;DECREMENT LOW BYTE OF LOOP
                    C RTN           ;IF THERE IS A CARRY THEN ALL DONE: END
                    C=C-1 P-Q       ;ELSE DECREMENT AGAIN TO GIVE NEXT
x6D5                JNC-21 ELOOP    ; Bn ADDRESS AND REPEAT PROCESS
```

CONTINUE AFTER BREAKPOINT - CONTBK.

DEPENDENCIES: FIND9 WRC1 KEYDBU ENTRY
MENU OUTHEX CR9

ROUTINES USED: PCTOC MESSL MSGIØ5
CLLCDE LEFTJ

INPUT: BUFFER9 CONTAINING TWO STACK VALUES & PC VALUE AT LAST BREAKPOINT

OUTPUT: CHECKS BREAKPOINT IS STILL VALID AND PUTS VALUES ON STACK (STK1 & STK2). WRITES 'NC GO RBn' INTO ENTRY AND JUMPS TO CR9 (B9=CPU) AND ENTERS CODE THROUGH BREAKPOINT. CONFIRMATION OF ADDRESS (BY R15) IS REQUIRED BY USER BEFORE EXECUTION COMMENCES.

USES: ALL REGISTERS, BUFFER9, BREAKPOINT TABLES

```
x6D6    CONTBK:     GOSUB FIND9     ;'PC' IS IN C[3:Ø]
                    A=C ALL         ;         A[3:Ø]
                    NC XQ PCTOC
                    PT=5
                    LC F
                    LC 1
                    LC 2            ;C[ADDR] = xF12 = 'B9'
```

```
CLOOP:  RDROM                   ;IF WORD @ LOOP IS ZERO THEN
        ?C#Ø    S&X             ;BREAKPOINT IS NOT ACTIVE
        JNC+ØC  CNEXT           ;TRY THE NEXT ONE
        N=C                     ;N HOLDS LOOP IN [6:3]
        PT=Ø
        C=C+1   M
        RDROM
        PT=2
        C=G                     ;C HOLDS BKPT IN [3:Ø]
        PT=3
        ?A#C    WPT             ;IF NOT SAME SET CARRY
        JNC+ØD  CFOUND          ;IF SAME THEN FOUND RIGHT BREAKPOINT
        C=N
CNEXT:  PT=Q
        PT=4
        PT=P
        PT=3
        C=C-1   PQ              ;DECREMENT LOOP LOWER BYTE
        JC+Ø3   CCEND           ;IF CARRY THEN JUMP TO CCEND
        C=C-1   PQ              ;ELSE MOVE LOOP TO NEXT Bn
        JNC-15  CLOOP           ;AND SEE IF IT IS THE RIGHT ONE
CCEND:  GOTO    MENU            ;NO CORRESPONDING BKPT FOUND
                                ;(MUST HAVE BEEN DELETED SO END)
CFOUND: C<>N                    ;PUT ENTRY ADDRESS IN N
        RCR     3               ;PUT LOOP IN C[3:Ø]
        PT=1                    ;GET ADDRESS OF
        C=C+C   WPT             ;  BREAKPOINT RETURN CODE
        C=C+1   WPT
        C=C+1   WPT
        C=C+1   WPT
        C=C+1   WPT
        C=C+1   PT
        M=C
        RCR     11              ;GET PAGE NUMBER IN C[6]
        PT=5
        LC      B
        LC      F
        LC      3               ;xBF3 IS ADDRESS OF 'ENTRY'
        A<>C    ALL
        SF      Ø               ;GO
        CF      1               ;NC
        GOSUB   WRC1            ;WRITE 'NC GO RBn' AT 'ENTRY'
        C=N                     ;GET BACK ADDRESS FOR USER
        RCR     13              ;  TO CONFIRM
        PT=Ø
        LC      4               ;4 DIGITS
        LC      Ø               ;Ø UNDERSCORES
        M=C
        NC XQ   CLLCDE
        NC XQ   MESSL
ALPHA:: CONT␣@␣
        GOSUB   OUTHEX          ;OUTPUT ADDRESS
        NC XQ   LEFTJ
```

```
                GOSUB   KEYDBU      ;GET A NEW KEY & WAIT FOR IT
                LDI     #87         ;  TO COME UP . COMPARE WITH R/S
                ?A#C    S&X         ;IF NOT R/S SET THE CARRY FLAG
                JNC+#4  CSETUP      ;R/S PRESSED SO CONTINUE SET UP
                GOTO    MENU        ;ELSE RETURN TO MENU
CSETUP:         SF      8
                NC XQ   MSG1#5      ;PRINT CONT @ xxxx ON TRACE PRINTER
                CF      5
                GOSUB   FIND9
                NC XQ   PCTOC
                PT=5
                LC      B
                LC      F
                LC      5
                PUSH                ;FOR SECURITY PUT 'RTN' ADDRESS ON STACK
                LDI     #06         ;POINT TO TOP REGISTER OF BUFFER9
                C=C+A   S&X
                RAM     SLCT
                READ    DATA
                RCR     5           ;RETRIEVE STK2 AND
                PUSH                ;     PUT IT ON THE STACK
                A<>C    S&X
                A=C     S&X
                RAM     SLCT        ;FROM THE HEADER
                READ    DATA
                RCR     1           ;RETRIEVE STK1 AND
                PUSH                ;     PUT IT ON THE STACK
x74B            GOTO    CR9         ;ENTERS B9>CPU HERE TO
                                    ;PRESERVE STACK AND AS BUFFER9
                                    ;IS ALREADY FOUND AND SELECTED
```

DISPLAY BREAKPOINT INFORMATION (ADDRESS) - DBI

```
DEPENDENCIES :  DBN     BAT     OUTHEX
ROUTINESUSED:   NONE
INPUT :         BREAKPOINT NUMBER IN A[S&X]
OUTPUT:         ADDRESS OF BREAKPOINT APPENDED TO DISPLAY
                                        .i.e. 'BKPTn ␣@␣ xxxx'
USES:           ALL REGISTERS CORRUPTED EXCEPT N

x74C  DBI:      GOSUB   DBN         ; DISPLAY BREAKPOINT NUMBER
                GOSUB   BAT         ; GET START ADDRESS OF BREAKPOINT
                C=C+1   M           ; TABLE  i.e. DF#0  (IF IN PAGE D etc.)
                A<>C    ALL         ; PUT ADDRESS IN A[ADDR]
                RCR     1           ; AND SET BREAKPOINT NUMBER
                C=#     S&X
                RCR     10
                C=C+C   M           ;MULTIPLY BY TWO TO GIVE OFFSET
                C=A+C   M           ;COMBINE OFFSET & BKPT TABLE ADDRESS
                RDROM               ;GET 1st WORD AT Bn
                ?C##    S&X         ; IF ZERO THEN NO BREAKPOINT
                JNC+#C  NOBK        ; SO JUMP & DISPLAY JUST 4 UNDERSCORES
```

```
                    PT=Ø
                    G=C                 ; PUT HIGH ADDRESS IN G
                    C=C+1   M
                    RDROM               ; GET LOW ADDRESS IN C[1:Ø]
                    PT=2
                    C=G                 ; COMBINE ADDRESSES AND ROTATE
                    RCR     13          ; READY FOR OUTHEX FORMAT
                    PT=Ø
                    LC  4               ; 4 DIGITS
                    LC  Ø               ; Ø UNDERSCORES
                    JNC+ØØ4 DBIE
          NOBK:     PT=Ø                ;IF NO BREAKPOINT THEN
                    LC  Ø               ; Ø DIGITS
                    LC  4               ; 4 UNDERSCORES
          DBIE:     M=C
x76D                GOTO    OUTHEX
```

LOW LEVEL BREAKPOINT EDITOR - EDBN

```
DEPENDENCIES:       GETHXA  BKED    DAFN    OUTHEX  ADDDIG  WRC1
                    DBI     ENBKPT  DBN     DELDIG  FINDBN
ROUTINES USED:      MSGIØ5  LEFTJ
INPUT:              HIGH LEVEL MENU COMES IN VIA EDN WITH
                    BREAKPOINT TO EDIT IN B[S&X] AND A[S&X]
OUTPUT:             EXITS THROUGH BKED (HIGH LEVEL EDITOR)
                    ALLOWS ENTRY OF NEW ADDRESS FOR BKPTn, PRINTING
                    OF BREAKPOINT ADDRESS & NORMAL LOCAL EDITING
                    FUNCTIONS ( ABORT, DELETE, FINISH )
USES:               ALL REGISTERS, DISPLAY, BREAKPOINT TABLE, ROM UNDER TEST

x76E    EDBN:       A<>C    S&X         ; GET BKPT#
                    N=C                 ; SAVE IN N
            EDBL:   C=N                 ; RETRIEVE BKPT#
                    A=C     S&X
                    GOSUB   DBI         ; DISPLAY BKPT INFORMATION
            EDBK:   GOSUB   GETHXA      ; GET HEXADECIMAL KEYPRESS
                    ?FS     2           ; IF SPECIAL KEY SET CARRY
                    JNC+    INPUT       ; IF Ø..F GO TO LOW LEVEL INPUT
                    LDI     ØØ6         ENTER↵
                    ?A#C    S&X
                    JC+     NXK1        ; IF NOT ENTER TRY NEXT KEY
                    C=N                 ; ELSE SAVE N
                    M=C
                    SF  8
                    NC  XQ  MSGIØ5      ; PRINT DISPLAY
                    CF  5
                    C=M
                    N=C                 ; RETRIEVE N
           JEDBK:   JNC-    EDBK
           NXK1:    C=Ø     S&X         ←
                    ?A#C    S&X
                    JC+     NXK2        ; IF NOT DELETE TRY NEXT KEY
```

```
EDEL:   GOSUB  DAFN      ; DISABLE & FIND BKPTn
        C=0    S&X       ; MAKE IT UNREADABLE BY MAKING
        WROM             ; Bn = 000
JEDBL:  JNC-   EDBL
NXK2:   C=C+1  S&X       ON
        ?A#C   S&X
        JC+    NXK3      ;IF NOT ON TRY NEXT KEY
EON:    SETHEX           ;REQUIRED FOR ENTRY FROM ERS
        GOSUB  ENBKPT    ; ENABLE BREAKPOINTS AND
        GOTO   BKED      ; RETURN TO HIGH LEVEL MENU
NXK3:   C=C+1  S&X       RIS
        ?A#C   S&X
        JC-    JEDBK     ;IF NOT RIS IGNORE KEY & GET ANOTHER
ERS:    PT=0             ;ELSE INCREMENT BREAKPOINT NUMBER
        C=N
        SETDEC
        C=C+1  PT        ; IF A CARRY IS CAUSED BY 9→10 THEN
        JC-    EON       ; RETURN TO TOP LEVEL OF BKED
        SETHEX           ; OTHERWISE GET NEXT BKPT INFO
        N=C
        JNC-   JEDBL
INPUT:  RCR    13        ;GET PAGE INTO C[1]
        PT=0             ; ADD ON DIGITS FOR OUTHEX
        LC     1         ;  TO INDICATE 1 DIGIT
        LC     3         ;  AND 3 REMAINING UNDERSCORES
        M=C              ;SAVE IN M READY
        RCR    9         ;PUT PAGE# INTO C[ADDR]
        PT=1             ;GET A WORD TO TEST FOR RAM OR ROM
        RDROM            ; GET CONTENTS
        A=C    WPT       ; STORE FOR COMPARISON LATER
        C=-C-1 WPT       ; TAKE 1S COMPLEMENT
        WROM             ; SAVE WORD
        RDROM            ; READ WORD. IF ROM WILL BE DIFFERENT
        C=-C-1 WPT       ; RE-COMPLEMENT
        WROM             ; WRITE ORIGINAL BACK (FOR RAM)
        ?A#C   WPT       ; COMPARE ORIGINAL & RE-READ VALUE
        JC-    JEDBL     ; SET CARRY IF ROM & SO IGNORE KEY
LLDSP:  C=N              ; GET BKPT#
        A=C    S&X       ;  INTO A[S&X] FOR DBN
        GOSUB  DBN       ; DISPLAY  BKPT␣@␣
        GOSUB  OUTHEX    ; OUTPUT PART FORMED ADDRESS
        NC XQ  LEFTJ     ; AND LEFT JUSTIFY DISPLAY
LLOOP:  GOSUB  GETHXA    ;GETHEXADECIMAL KEYPRESS
        FS?    02        ;SET CARRY IF SPECIAL KEY
        JNC+   LLIN      ;IF 0..9 THEN ORDINARY INPUT
        C=0    S&X       ←
        ?A#C   S&X       ;IF NOT ← TRY NEXT KEY
        JC+    NXK4
LLDEL:  C=M
        PT=0
        C=C-1  PT
        ?C#0   PT
        JC+    LDEL
        GOTO   EDBL      ; IF DELETE 1ST DIGIT GO UP A LEVEL
```

```
LDEL:   GOSUB   DELDIG          ;ELSE DELETE DIGIT &
        JNC-    LLDSP           ; RE-DISPLAY AT THIS LEVEL
LLIN:   C=M                     ;ORDINARY INPUT
        ?C#Ø    MS
        JNC-    LLOOP           ;CHECK THERE IS ROOM LEFT & IGNORE IF NOT
        GOSUB   ADDDIG          ;OTHERWISE ADD DIGIT AND
        JNC-    LLDSP           ;  RE-DISPLAY AT THIS LEVEL
NXK4:   C=C+1   S&X     [ON]    ;IF NOT ON THEN TRY ANOTHER KEY
        ?A#C    S&X
        JNC+    NXT             ;IF ON THEN GO & SKIP A BIT
        C=C+1   S&X     [R/S]
        ?A#C    S&X
        JC-     LLOOP           ;IF NOT R/S THEN GET ANOTHER KEY
        CF      3               ;INDICATE SHOULD RETURN TO EDBL
        C=M                     ;GET NUMBER OF DIGITS INPUT
        A=C     S&X             ;  IN  A[S&X]
        LDI     ØØ4
        PT=Ø
        ?A#C    PT              ;IF NOT 4 THEN SET CARRY
        JNC+    FOURIN          ;IF 4 DIGITS SKIP NEXT BIT
        JNC-    LLOOP           ;R/S HAS NO EFFECT UNLESS 4 DIGITS INPUT
NXT:    SF      3               ;INDICATE SHOULD RETURN TO EON
        C=M                     ;GET NUMBER OF DIGITS INPUT
        A=C     S&X             ;  INTO A[S&X]
        LDI     ØØ4
        PT=Ø
        ?A#C    PT              ;IF NOT 4 THEN SET CARRY
        JNC+    FOURIN          ;IF 4 DIGITS THEN SKIP NEXT BIT
        GOTO    EON             ;ELSE ON ABORTS ADDRESS ENTRY &
                                ;RETURNS TO TOP LEVEL OF BKED
FOURIN: GOSUB   DAFN            ;DISABLE BKPTS AND SET ADDR OF BKPTn
        A=C     M               ;STORE IN A[ADDR]
        C=M                     ;GET HIGH ADDRESS BYTE
        RCR     3               ;  INTO C[1:Ø]
        C=Ø     XS              ;  INTO C[2:Ø]
        A<>C    M               ;GET ADDRESS OF BKPT
        A=C     M               ;STORE FOR LATER USE
        WROM                    ;WRITE HIGH ADDRESS BYTE INTO Bn
        C=M                     ;GET LOW ADDRESS BYTE
        RCR     1               ;  INTO C[1:Ø]
        C=Ø     XS              ;  INTO C[2:Ø]
        A<>C    M               ;GET ADDRESS OF BKPT (Bn)
        C=C+1   M               ;  ADD ONE FOR LOW BYTE POSITION
        WROM                    ;WRITE LOW ADDRESS BYTE INTO Bn+1

;The entry for BKPTn has been updated in Bn
;M and N are still uncorrupt and contain address of BKPT
;and BKPT# respectively

;Now RBn to RBn+3 must be updated
;It is not necessary to enable the actual breakpoint as that
;will be done automatically when we return to the top level menu
;of BKED
```

```
                GOSUB   FINDBN      ; GET ADDRESS OF Bn
                RCR     3           ; PUT INTO C[7:0]
                PT=1                ; DOUBLE LOW ADDRESS BYTE
                C=C+C   WPT
                C=C+1   WPT         ; AND ADD 14HEX TO GET FROM
                C=C+1   WPT         ; Bn TO RBn
                C=C+1   WPT
                C=C+1   WPT
                C=C+1   PT          ; C[3:0] = RBn
                RCR     11          ; C [ADDR] = RBn
                B<>C    M           ; B [ADDR] = RBn
                C=M
                RCR     12          ; C[ADDR] = BKPTn
                RDROM               ; GET OUR1
                B<>C    M           ; SWAP IN ADDRESS OF RBN
                WROM                ; WRITE OUR1 INTO RBn
                C=C+1   M           ; POINT TO RBn+1
                B<>C    M           ; SWAP BACK IN ADDRESS OF BKPTn
                C=C+1   M           ; POINT TO BKPTn+1
                RDROM               ; GET OUR2
                B<>C    M           ; SWAP IN ADDRESS OF RBn+1
                WROM                ; WRITE OUR2 INTO RBn+1

; B[ADDR] = BKPTn+1
; C[ADDR] = RBn+1
; Now write 'NC GO BKPTn+2' into RBn+2

                C=C+1   M           ; C[ADDR] = RBn+2
                A=C     M           ; A[ADDR] = RBn+2 READY FOR WRC1
                C=B     M           ; C[ADDR] = BKPTn+1
                C=C+1   M           ; C[ADDR] = BKPTn+2
                RCR     3           ; C[3:0] = BKPTn+2
                M=C                 ; M[3:0] = BKPTn+2 READY FOR WRC1
                SF  0               ; GO
                CF  1               ; NC
                GOSUB   WRC1        ; WRITE 'NC GO BKPTn+2' TO RBn2 & 3
                ?FS     3           ; IF R/S PRESSED THEN JUST SET UP THIS
                JC+04   JEON        ; BKPT
                GOTO    EDBL        ; ELSE .
x82E    JEON:   GOTO    EON         ;     GO TO TOP LEVEL OF BKED VIA EON
```

PRINT CPU ACCUMULATOR CONTENTS - PRCPU

DEPENDENCIES : MENU
MK9

ROUTINES USED: ENCPØØ
IAUNA - PRINTER ROM ADDRESS 6DB2
PBYTCX - PRINTER ROM ADDRESS 6FF2
OUTPCT - PRINTER ROM ADDRESS 63DD

INPUT: BUFFER 9 CONTAINING A,B,C,M,N. PRCPU CHECKS IF A[S&X] = Ø13
OUTPUT: IF PRINTER IN TRACE MODE GIVES:

"CPU:
C= xxxxxxxxxxxxxxx
A= xxxxxxxxxxxxxxx
B= xxxxxxxxxxxxxxx
M= xxxxxxxxxxxxxxx
N= xxxxxxxxxxxxxxx"

```
x82F  PRCPU:   LDI  Ø13  ENTER     ; Entry is from MENU so see if
               ?A#C  S&X           ; enter key is down before printing
               JNC+  PRCPU1        ; If it is enter PRCPU1
        JMENU: GOTO  MENU          ; If it is not jump back to menu
        PRCPU1: NC  XQ  ENCPØØ     ; Enable chip ØØ
               NC  XQ  IAUNA       ; Try to initialise print
               JNC-  JMENU         ; No print so return immediately
               LDI  Ø43  'C'       ; Print 'CPU:'
               NC  XQ  PBYTCX
               LDI  Ø5Ø  'P'
               NC  XQ  PBYTCX
               LDI  Ø55  'U'
               NC  XQ  PBYTCX
               LDI  Ø3A  ':'
               NC  XQ  PBYTCX
               NC  XQ  OUTPCT      ; Send newline
               GOSUB  MK9          ; Locate buffer 9 (make it if it
               A=A+1  S&X          ;  does not exist already). Point to C
               A<>B  S&X
               LDI  Ø43  'C'       ; Label output with 'C'
               M=C
               GOSUB  PRREG        ; Print register & increment pointer to next
               LDI  Ø41  'A'
               M=C
               GOSUB  PRREG        ; Register 'A'
               LDI  Ø42  'B'
               M=C
               GOSUB  PRREG        ; Register 'B'
               LDI  Ø4D  'M'
               M=C
               GOSUB  PRREG        ; Register 'M'
               LDI  Ø4E  'N'
               M=C
               GOSUB  PRREG        ; Register 'N'
x87Ø   JMENU2: GOTO  MENU          ; Return to main menu
```

```
x873  PRREG:   NC  XQ  ENCPØØ      ; Enable chip ØØ
               NC  XQ  IAUNA       ; Check printing is OK
               JNC-    JMENUR      ; If not return to menu
               C=M                 ; Else get character label from M
               NC  XQ  PBYTCX      ; Send to printer
               LDI     Ø3D  '='    ; Send '=' to printer
               NC  XQ  PBYTCX
               NC  XQ  ENCPØØ      ; Enable chip ØØ
               C=B     S&X         ; Get register address
               RAM     SLCT        ; Select register
               C=C+1   S&X         ; Increment pointer for next time
               C<>B    S&X         ;   and put it back in B[S&X]
               READ    DATA        ; Get register
               SLCT    Q           ; Pointer Q is used as a counter for
               R=Ø                 ;   the number of digits output
CPR1:          RCR     13          ; Put digit to be printed in C[Ø]
               M=C                 ; Save register for next time
               SLCT    P
               R=Ø
               A=C     @R
?A<C @R  >     LDI     ØØA
               JNC+    GT9         ; If digit is A..F jump to GT9
               LDI     Ø3Ø         ; Prepare mask for numbers Ø..9
               JNC+    CPR2
GT9:           C=C-1   S&X         ; If A..F then subtract 9
               A=A-C   @R
               LDI     Ø4Ø         ; And prepare mask for letters A..F
CPR2:          A<>C    @R
               NC  XQ  PBYTCX      ; Output character
               C=M
               SLCT    Q           ; Pointer Q is the digit counter
               R=R-1               ; When it returns to Ø all the digits
               ?R=Ø                ;   have been output
               JNC-    CPR1        ; Until then go around the loop again
xBAØ           NC  GO  OUTPCT      ; Finally send end of line and return
```

STATUS EDITOR (BUF9 FORMAT) - STED.

DEPENDENCIES: MAIND, ALTD, KEYDBU, TKEY, TKEYH, MK9

ROUTINES USED: GOSUB, MSG10S, GOTO

INPUT: BUFFER 9 CONTAINNING BUFFER ID, REGISTERS C,A,B,M,N AND STATUS (IN BUF9 FORMAT)

OUTPUT: UPDATED BUFFER9 WITH NEW STATUS INFORMATION

```
x903  STED:     ST=ϕ                    ; Clear all flags ϕ..7 (used for prompting)
                CF  11                  ; Clear Q prompt
                CF  9                   ; Clear P prompt
                GOSUB   MK9             ; Find buffer 9, Addr in A
                LDI     ϕϕ6             ; Add 6 to point to top register containning
                C=C+A   S&X             ; status information
                RAM  SLCT               ; Get this into into C
                READ  DATA
                N=C                     ; and keep it in N throughout edit session.
LOOP1:          CF  1ϕ                  ; Indicates display main set of status
LOOP:           ?FS  1ϕ                 ; If flag 10 is set then display alternative
                JC+ ϕ5  DALT            ; set below
DMAIN:          GOSUB   MAIND           ; else show main set
                JNC+ϕ4  CONT
DALT:           GOSUB   ALTD
CONT:           GOSUB   KEYDBU          ; Get key (debounced) and wait for it to come up
                LDI     ϕ18     [ON]    ; key pressed is in A[S&X]
                ?A#C    S&X             ; If key down was not [ON] then try next key
                JC+ϕF   CONTA
                GOSUB   FIND9           ; On pressed: find the buffer and top register
                LDI     ϕϕ6
                C=C+A   S&X
                RAM  SLCT
                C=N                     ; Get edited status
                C=ϕ  MS                 ; Ensure [13] is 1 so register cannot
                C=C+1  MS               ; be all zero
                WRITE  DATA             ; Write edited values into buffer
                GOTO  MENU              ; Exit STED : GOTO MAIN DEBUG MENU
CONTA:          LDI     ϕ87     [R/S]
                ?A#C    S&X             ; Was the key [R/S], if not try next key
                JC+ϕS   CONTB
                ?FS     1ϕ              ; If ALT set displayed then change to MAIN
                JC-24   LOOP1
                SF      1ϕ              ; else change to ALT set
                JNC-25  LOOP
CONTB:          LDI     ϕ7ϕ     [C]     ; If [C] not pressed then try next key
                ?A#C    S&X
                JC+ϕ4   CONTC
                                        ; Reset all status values to default
JLOOP:          N=C                     ; and continue on same display set
                JNC-2C  LOOP
CONTC:          C=C+1   S&X     [H]     ; If [H] not pressed then try next key
                ?A#C    S&X
                JC+ϕ6   CONTD
                C=N                     ; Get status register
```

```
                PT=7                    ; Nybble 7 indicates HEX if ϕ, DEC if 1
                C=ϕ   @R                ; Set to HEX i.e. to ϕ
JLOOP3:         SF  1ϕ                  ; Ensure the appropriate (ALT) set displayed
                JNC-ϕ9  JLOOP           ; and continue - writing STATUS to N
CONTD:          LDI   ϕ8ϕ       [D]
                ?A#C  S&X               ; If not [D] pressed then try next
                JC+ϕ6   CONTE
                C=N                     ; Get status
                PT=7
                C=ϕ   @R
                C=C+1 @R                ; Set to DEC i.e. C[7] = 1
                JNC-ϕA  JLOOP3          ; Display ALT set & write STATUS to N
CONTE:          LDI   ϕ13       [ENTER↑]
                ?A#C  S&X               ; If not [ENTER↑] then try next key
                JC+ϕ7   CONTF
                C=N                     ; Save status
                M=C                     ;   in M
                SF    8                 ; Indicate printer is to be used if F IS set
                GOTO  PATCH             ; Print chars on screen and restore STATUS.
CONTF:          C=C+1 S&X       [Q]
                ?A#C  S&X               ; If not [Q] then try next key
                JC+1D   CONTG
QIN:            GOSUB   MAIND           ; Show main display
                GOSUB   TKEYH           ; Get a Hex key value within a second or so
                JNC+ϕB  QQIN            ; RTN1: key pressed so process it below
                SF    11                ; RTN2: No key or invalid one so change
                GOSUB   MAIND           ; value of Q to an underscore (Flag 11)
                GOSUB   TKEYH           ; display and try for another key
                JNC+ϕ3  QQIN            ; RTN1: key pressed so process it below
                CF    11                ; RTN2: No key so change '_' back to value
                JNC-1ϕ  QIN             ; and try again until a valid one is pressed
QQIN:           RCR   1ϕ                ; Put Hex value input into C[4] => 'Q'
                B<>C  ALL               ; Save in B
                C=N                     ; Get status
                PT=4                    ; Put nybble 4 back into status
                B<>C  @R                ;   register
                N=C                     ; And save status
                CF    11                ; Ensure that Q is displayed
JLOOP4:         CF    1ϕ                ;   and that the main display set is on
JLOOP2:         GOTO  LOOP              ; Long jump back to start of routine
CONTG:          LDI   ϕ83       [P]
                ?A#C  S&X               ; If not [P] then try next key
                JC+1A   CONTH
PIN:            GOSUB   MAIND           ; Show main display
                GOSUB   TKEYH           ; Get a Hex key value within a second or so
                JNC+ϕB  PPIN            ; RTN1: key pressed so process it below
                SF    9                 ; RTN2: No key or invalid key so change
                GOSUB   MAIND           ; value of P to an underscore (Flag 9)
                GOSUB   TKEYH           ; display and try for another key
                JNC+ϕ3  PPIN            ; RTN1: key pressed so process it below
                CF    9                 ; RTN2: No key so change '_' back to
                JNC-1ϕ  PIN             ; value and try again until a valid one down
PPIN:           RCR   9                 ; Put value into C[5]
                B<>C  ALL               ;   and save in B
                C=N                     ; Get status
                PT=5                    ; Put nybble 5 back into status
                B<>C  @R                ;   register
```

```
                CF  9                       ; Ensure P is displayed not underscore
JLOOP5:         N=C                         ; Save status register & jump back
                JNC-2Ø  JLOOP4              ; with main display set up
CONTH:          LDI  Ø34        [R]
                ?A#C  S&X                   ; If not [R] then try next key
                JC+24  CONTJ
RIN:            GOSUB  MAIND                ; Show main set
                GOSUB  TKEY                 ; Get a key within a second or so
                JNC+ØB  RRIN                ; RTN1: key pressed so process it below
                SF  7                       ; RTN2: No key so change PT value
                GOSUB  MAIND                ; to underscore (Flag 7) and display set
                GOSUB  TKEY                 ; Wait for another key for about a second
                JNC+Ø3  RRIN                ; RTN1: key pressed so process it below
                CF  7                       ; RTN2: No key so change PT back
                JNC-1Ø  RIN                 ; from '_' to a value P/Q and try again
RRIN:           CF  7                       ; key pressed so clear underscore flag
                LDI  Ø83        [R][P]      ; Check if it was P pressed
                ?A#C  S&X
                JNC+ØA  PDOWN               ; If so skip down a bit
                LDI  Ø14        [R][Q]      ; Check if it was Q pressed
                ?A#C  S&X
                JC-19  RIN                  ; If neither P nor Q wait for another key
QDOWN:          C=N                         ; Q pressed so set nybble 6
                PT=6                        ; of status to 1 to indicate active
                C=Ø  @R                     ; pointer is Q
                C=C+1  @R
                JNC-24  JLOOP5
PDOWN:          C=N                         ; P pressed so set nybble 6 of
                PT=6                        ; status to Ø to indicate active
                C=Ø  @R                     ; pointer is P
                JNC-28  JLOOP5
CONTJ:          LDI  Ø31        [G]
                ?A#C  S&X                   ; If not [G] then try next key
                JC+33  CONTK
G1:             GOSUB  ALTD                 ; Show ALTernative display set
                GOSUB  TKEYH                ; Get a hex key within a second or so
                JNC+ØB  G2                  ; RTN1: key pressed so process it below
                SF  4                       ; RTN2: No valid hex key so underscore
                GOSUB  ALTD                 ; 1st nybble of G's value and try again
                GOSUB  TKEYH                ; Get hex key
                JNC+Ø3  G2                  ; RTN1: key pressed so process it below
                CF  4                       ; RTN2: Reset '_' to 1st nybbles value
                JNC-1Ø  G1                  ; and try again
G2:             CF  4                       ; Turn of 1st nybble underscore
                RCR  11                     ; and put hex value into status
                B<>C  ALL                   ; nybble 3 corresponding to top
                PT=3                        ; nybble of G
                C=N
                B<>C  @R
                N=C
G3:             GOSUB  ALTD                 ; Now get 2nd nybble of G
                GOSUB  TKEYH                ; Get a hex key within a second or so
                JNC+ØB  G4                  ; RTN1: key pressed so process it below
                SF  3                       ; RTN2: No valid hex key so underscore
                GOSUB  ALTD                 ; 2nd nybble of G's value and try again
                GOSUB  TKEYH                ; Get hex key
                JNC+Ø3  G4                  ; RTN1: key pressed so process it below
```

```
                CF 3            ; Reset underscore under 2nd nybble
                JNC-1ϕ  G3      ; and try again
G4:             CF 3            ; Reset underscore
                RCR 12          ; Put 2nd nybble of G into C[2]
                B<>C ALL        ;  "  "  "  "  "  B[2]
                C=N             ; Get status register
                B<>C XS         ; Put in 2nd nybble
                N=C             ; Save status register
JLOOP6:         GOTO LOOP       ; Wait for another key
CONTK:          LDI ϕ74  [5]
                ?A#C S&K        ; If not [5] then invalid key so
                JC-ϕ6 JLOOP6    ; wait for another
S1:             GOSUB ALTD      ; Display alternative set
                GOSUB TKEYH     ; Get a hex key within a second or so
                JNC+ϕ7 S3       ; RTN1: Hex key pressed so process it below
                FS? ϕ6          ; RTN2: If underscore under 1st nybble reset it
                JC+ϕ3  S2       ;  below
                SF ϕ6           ; otherwise change to underscore
                JNC-ϕA S1       ; and try again
S2:             CF 6            ; Reset underscore to value
                JNC-ϕC S1       ;  and try again
S3:             CF 6            ; Reset underscore
                RCR 13          ; Put top nybble of ST into C[1]
                B<>C ALL
                PT=1
                C=N             ; Get status
                B<>C @R         ;  swap in nybble
                N=C             ; Store status
S4:             GOSUB ALTD      ; 2nd nybble : Shows alternative display set
                GOSUB TKEYH     ; Get a hex key within a second or so
                JNC+ϕ7 S6       ; RTN1: Hex key pressed so process it below
                ?FS 5           ; RTN2: If underscore under 2nd nybble reset it
                JC+ϕ3  S5       ;  below
                SF 5            ; otherwise change to underscore
                JNC-ϕA S4       ;  and try again
S5:             CF 5            ; Reset underscore to value
                JNC-ϕC S4       ;  and try again
S6:             CF 5            ; Reset underscore
                PT=ϕ            ; Put bottom nybble of ST into C[ϕ]
                C=N             ; Get status
                B<>C @R         ; Swap it into status
JLOOP7:         N=C             ; Save status
                JNC-2D JLOOP6   ; Go and get another key
PATCH:          NC XQ MSG1ϕ5    ; Patch to make print routine work correctly
                CF 5            ; N.B. Only prints if printer is in trace mode.
                C=M             ; Restore status
xA2ϕ            JNC-ϕ6 JLOOP7   ;  and continue
```

MAIN DEBUG MENU & FAT ENTRY

DEPENDENCIES: ALL ROUTINES !
ROUTINES USED: CLLCDE, CLDSP, MESSL, OFF, ENCPØØ
INPUT: POSSIBLY BUFFER 9
ENTRY FROM NORMAL XROM ROUTINE AND RETURNS FROM EDITORS AND TEST ROUTINES
OUTPUT: POSSIBLY BUFFER 9
CONTROLS ENTRY TO ALL DEBUG ROUTINES

```
xA27 MEN2:      CLRKEY                  ; SOME FUNCTIONS MAY RETURN WITH A KEY
                ?KEY                    ; STILL DOWN SO THEY COME IN HERE AND
                JC-Ø2  MEN2             ; THE WAITING IS DONE BY THESE LINES
                JNC+   MENU             ; WHEN THE KEY COMES UP 'MENU' IS ENTERED
     FUNCTION:: DEBUG                   ; FAT NAME
xA3Ø MENU:      NC XQ  CLLCDE           ; CLEAR & ENABLE LCD
                NC XQ  MESSL            ; DISPLAY PROMPT
     ALPHA:     DEBUG␣CMD␣?␣            ;     'DEBUG CMD ?'
                NC XQ  ENCPØØ           ; ENABLE CHIP ØØ
                CLRKEY                  ; FORCE OUT ANY KEYDOWN FLAG
                C=Ø    ALL              ; SET UP 2'½ MINUTE TIMEOUT VALUE
                PT     4                ; IN THE C REGISTER (2:42)
                LC     4
     LOOP:      C=C-1  ALL              ; DECREMENT TIMEOUT VALUE
                JC+17  TIMEOUT          ; IF CARRY IS SET THEN TURN CALCULATOR OFF
                ?KEY                    ; OTHERWISE LOOK TO SEE IF A KEY IS DOWN
                JNC-Ø3 LOOP             ; IF NO KEY THEN CONTINUE WAITING
                C=KEY                   ; GET THE KEY INTO C[S&X]
                RCR    3
                C=Ø    XS
                A=C    S&X              ; AND FORMATTED INTO A[S&X] FOR TESTING
                CLRKEY                  ; WAIT FOR KEY TO COME UP BEFORE
                ?KEY                    ; DOING ANYTHING ELSE
                JC-Ø2
                LDI    Ø18    ON
                ?A#C   S&X              ; IF IT WAS NOT THE ON KEY TRY THE NEXT
                JC+13  CON1
     EXIT:      C=REG  13(c)            ; OTHERWISE TIDY UP EVERYTHING AND EXIT
                C=Ø    M                ; TO NORMAL OPERATIONAL MODE
                PT=3
                LC     3
                CF     1Ø
                SF     13
                REG=C  12(b)
                NC GO  CLDSP (1ØEØ)
     TIMEOUT:   C=REG  13(c)            ; IF TIMED OUT THEN TIDY EVERYTHING UP
                C=Ø    M                ; AND TURN CALCULATOR OFF TO PRESERVE
                PT=3                    ; BATTERY LIFE. EVERYTHING IS IN BUFFER 9
                LC     3                ; SAFE AND SOUND, BREAKPOINTS ARE STILL
                CF     1Ø               ; IN TACT AND THE POWER DOWN HP-IL
                SF     13               ; ROUTINE IS RUN IF IT IS PRESENT
                REG=C  12(b)            ; PRESERVING BATTERIES IN ANY LOOP DEVICES
                NC GO  OFF (11C8)
```

```
CON1:   LDI   Φ31   G
        ?A#C  S&X
        JC+Φ4 CON2
START1: GOTO  START       ;IF G PRESSED THEN ENTER CODE USING START
CON2:   LDI   ΦCΦ   E
        ?A#C  S&X
        JNC-Φ6 START1     ;IF E PRESSED THEN ENTER CODE USING START
        LDI   Φ74   S
        ?A#C  S&X
        JC+Φ4 CON3
        GOTO  STED        ;IF S PRESSED THEN ENTER STATUS EDITOR
CON3:   LDI   Φ11   F
        ?A#C  S&X
        JC+Φ4 CON4
        GOTO  FLAGE       ;IF F PRESSED ENTER USER FLAG EDITOR
CON4:   LDI   Φ7Φ   C
        ?A#C  S&X
        JC+Φ4 CON5
        GOTO  CONTBK      ;IF C PRESSED TRY CONTINUING FROM A
CON5:   LDI   Φ3Φ   B     ;                    BREAKPOINT STOP
        ?A#C  S&X
        JNC+  BKED        ;IF B PRESSED ENTER BREAKPOINT EDITOR
        LDI   Φ34   R
        ?A#C  S&X
        JC+Φ4 CON6
        GOTO  RED         ;IF R PRESSED ENTER REGISTER EDITOR
xA94 CON6:

        GOTO  PRCPU       ;SEE IF ENTER↑ PRESSED & PRINT CPU REGISTERS
                          ;IF IT WAS, ELSE JUMP BACK TO MENU
```

BREAKPOINT EDITOR - BKED.

DEPENDENCIES: EDN, EDBN, GETHXA, SURE, DABKPT, ENBKPT, MENU, DBS

ROUTINES USED: GOSUB, PCTOC, MSG10S

INPUT: NONE - Enters via MENU

OUTPUT: Top level of breakpoint editor merely controls use of lower levels such as DABKPT (disable breakpoints), EDN (edit breakpoint N), ENBKPT (enable breakpoints). Includes facility to erase all breakpoints after checking using routine SURE. Also allows printing of breakpoint status.

```
xA95  EDN:   GOTO   EDBN          ; Enter low level edit of breakpoint N
xA98  BKED:  GOSUB  DBS           ; Indicate status BKPTS ON/OFF/NONE
             GOSUB  GETHXA        ; Get hexadecimal keypress
             ?FSET  2             ; IF special keypressed jump
             JC+2D  SPEC          ;    to special key handler
             LDI HEX ØØA          ; If key is Ø..9 then set conv
             ?A<C   S&X           ;    and edit BKPTn
             JC-    EDN
             LDI HEX ØØC    [C]   ; If not C pressed then jump to next
             ?A#C   S&X
             JC+17  BC1
             GOSUB  SURE          ; Otherwise ask if user is sure if he
             JNC-   BKED          ; wants to clear BKPTs and ignore if not
             GOSUB  DABKPT        ; To delete, disable breakpoints and
             NC XQ  PCTOC         ; Clear every word in range xFØØ - xF3B
             PT=5
             LD@R   F
             LD@R   3
             LD@R   B
             C=Ø    S&X
             SLCT   Q
             R=4
             SLCT   P
             R=3
      CLBL:  WROM
             C=C-1  P-Q
             JNC-Ø2 CLBL          ; Done when conv of xFQPØ - xEQPFF
      JBKED: JNC-   BKED          ; Jump back to status display
      BC1:   C=C+1  S&X    [D]    ; D pressed
             ?A#C   S&X
             JC+Ø5  BC2
             GOSUB  DABKPT        ;    disable breakpoints
             JNC-Ø7 JBKED
      BC2:   C=C+1  S&X    [E]    ; IF E pressed
             ?A#C   S&X
             JC-ØA  JBKED
```

```
                GOSUB   ENBKPT          ;   enable breakpoints
                JNC-ØE  JBKED
SPEC:           LDI HEX ØØ6     [ENTER] ; If ENTER pressed
                ?A#C    S&X
                JC+Ø5   SPEC1
                SETF    8
                NC XQ   MSG1Ø5          ;   then print the screen
JBKED2:         JNC-16  JBKED
SPEC2:          LDI HEX ØØ1     [ON]    ; If ON pressed
                ?A#C    S&X
                JC-Ø4   JBKED2
xAD8            GOTO    MENU            ;   then jump back to main menu
```

SUBROUTINE - GET NYBBLE FROM BUFFER 9

DEPENDENCIES: NONE
ROUTINES USED: NONE
INPUT : A[S&X] = HEX Ø {+ offset} {- offset}
N REGISTERS CONTENTS
MS = NYBBLE BEING EDITED
BBB IN [M] = BASE ADDRESS OF BUFFER 9
R = REGISTER BEING EDITED IN [XS]
BUFFER 9 IN CPU FORMAT
OUTPUT: N UNCHANGED
BUFFER UNCHANGED
NYBBLE IS IN C[MS]
APPROPRIATE REGISTER IN BUFFER 9 IS SELECTED
B[MS] INDICATES NUMBER OF ROTATIONS -1 REQUIRED
TO PUT REGISTER CONTENTS BACK
USES: N, A, B, C,
M IS UNUSED

```
x ADB   GETNYB:   PT=13          ; PUT CONSTANT 14dec[MS] INTO A[MS]
                  LC   E
                  A=C  MS
                  C=N            ; GET EDITOR POINTERS INTO C
                  PT=1           ; POSITION ACTIVE POINTER TO NYBBLE 1
                  LOOP1:  ?A#Ø  PT   ; IF A[1] = Ø THEN DONE +ve offsets
                  JNC+Ø8  PART2      ;  SO GO TO PART2
                  C=C+1   MS         ; ELSE INCREMENT NYBBLE NUMBER
                  A=A-1   PT         ; DECREMENT OFFSET
                  ?A#C    MS         ; IF C[MS] ≠14DEC THEN GOTO LOOP1
                  JC-Ø5   LOOP1
                  C=C+1   XS         ; ELSE INCREMENT REGISTER NUMBER
                  C=Ø     MS         ; AND POSITION NYBBLE POINTER TO
                  JNC-Ø8  LOOP1      ; BOTTOM OF REGISTER. GO TO LOOP1
                  PART2:  PT=Ø       ; POSITION POINTER
                  LOOP2:  ?A#Ø  PT   ; IF A[Ø] = Ø THEN DONE -VE OFFSETS
                  JNC+Ø8  PART3      ;  SO GO TO PART3
                  A=A-1   PT         ; DECREMENT OFFSET
                  C=C-1   MS         ; DECREMENT NYBBLE NUMBER
                  JNC-Ø4  LOOP2      ; IF CARRY CLEAR GO TO LOOP2
                  C=C-1   XS         ; OTHERWISE POSITION POINTERS TO
                  PT=13              ; PREVIOUS REGISTER AND TOP NYBBLE
                  LC   D
                  JNC-Ø9  PART2      ; GO TO PART2 (AS POINTER WAS MOVED)
                  PART3:  A=C  ALL   ; PUT NYBBLE & REGISTER INFO IN A
                  B=A  MS            ; PUT NYBBLE NUMBER IN B FOR LATER USE
                  C=Ø  M
                  RCR  13
                  C=C+A  M           ; ADD BASE ADDRESS OF BUFFER & REGISTER
                  C=Ø  S&X           ; SELECT PERIPHERAL Ø
                  PRPH SLCT
                  RCR  3             ; GET ADDRESS OF REGISTER
                  RAM  SLCT          ; SELECT IT
                  READ DATA          ; AND READ IT
                  LOOP3:  RCR  1     ; ROTATE CONTENTS A[MS] TIMES
                  A=A-1  MS          ; UNTIL DIGIT REQUIRED IS IN C[MS]
                  JNC-Ø2 LOOP3       ; REGISTER IS STILL SELECTED
x BØØ             RTN                ; B[MS] CAN BE USED TO PUT NEW VALUE IN
```

SUBROUTINE - OUTPUT HEX DIGITS - OUTHEX

```
DEPENDENCIES:   NONE
ROUTINES USED:  NONE
INPUT       :   M REGISTER CONTAINING
                    MS:NUMBER OF UNDERSCORES
                    Φ :-NUMBER OF CHARACTERS (HEX DIGITS)
                    1 :- 1st HEX DIGIT
                    :
                    N :- Nth HEX DIGIT.....  . DISPLAY ENABLED & RIGHT JUSTIFIED.
USES :          A   MS,S&X
                B   S&X,M
                C   ALL
                G , PT , M , ST
OUTPUT :        M   UNCHANGED          N   UNCHANGED
                ST  UNCHANGED
                PT= Φ
                APPENDS  HEX DIGITS  ONTO  DISPLAY

xB14   OUTHEX: C=ST                    ; SAVE STATUS IN B[M]
               RCR 3
               C<>B M
               C=M                     ; GET DISPLAY INFΦ
               A=C  S&X                ; A[S&X] = NUMBER OF DIGITS USED SO FAR
               PT=1
               A=Φ  PT
               A=Φ  XS
               A=C  MS                 ; A[MS] = NUMBER OF UNDERSCORES TO BE SHOWN
        OUH    B=A  S&X                ; B[S&X] = NUMBER OF DIGITS NOT YET DISPLAYED
               ?A#Φ S&X                ; IF NO MORE DIGITS TO BE DISPLAYED
               JNC+24  UNS             ;   GO DOWN AND OUTPUT UNDERSCORES
               PT=Φ                    ;OTHERWISE POSITION POINTER AT NEXT DIGIT
        OUL:   ?A#Φ S&X                ;  TO BE OUTPUT, BY COUNTING DOWN A[S&X]
               JNC+Φ4                  ;  AND MOVING PT UP FROM Φ UNTIL
               +PT                     ;  A[S&X] = Φ
               A=A-1 S&X
               JNC-Φ4  OUL
               G=C                     ; GET DIGITS AT PT AND PT+1
               PT=Φ
               C=Φ  ALL
               C=G                     ; WRITE THEM TO C[1:Φ]
               +PT
               LC  Φ                   ; CLEAR C[1] LEAVING PT@ Φ & DIGIT [Φ]
               ST=C                    ; PUT DIGIT IN STATUS REGISTER
               ?FS 3                   ; IF 8 OR ABOVE SET THE CARRY FLAG
               JNC+Φ8  NUM             ; IF Φ-7 THEN DIGIT IS JUST A NUMBER-JUMP
               C=C-1 PT                ; LEAVES DIGITS IN RANGE 8-F ONLY
               C=C-1 PT                ; SUBTRACT 2 FROM DIGIT GIVING 6-D
               ST=C                    ; STORE IN STATUS
               C=C+1 PT                ; ADD 2 TO
               C=C+1 PT                ;   RESTORE VALUE
               ?FS 3                   ; IF 8(A) OR ABOVE SET THE CARRY FLAG
               JC+Φ4  LET              ; MUST BE A-F SO GO & DO LETTER OUTPUT
        NUM:   +PT                     ; PT=1, WRITE
               LC  3                   ;   IN 3 GIVING CORRECT DISPLAY
               JNC+Φ5  OUT             ;   CODE 3Φ-39 FOR NUMBERS & OUTPUT
```

```
LET:    A<>C   S&X      ; MUST BE A LETTER A-F SO
        LDI             ;   SUBTRACT 9 TO GIVE ΦΦ1 - ΦΦ6
        ΦΦ9             ;   THE CORRECT DISPLAY CODE
        C=A-C  S&X      ;   IN C[S&X]
OUT:    WRABC1R         ; OUTPUT DIGIT TO RHS OF DISPLAY
        C=B    S&X      ; GET NUMBER OF DIGITS NOT YET DISPLAY
        C=C-1  S&X      ; SUBTRACT ONE FOR DIGIT JUST OUTPUT
        A=C    S&X      ; MAKE THAT THE NEXT ONE TO BE SHOWN
        C=M             ; PUT M INTO C READY FOR IT
        JNC-25  OUH     ; GO BACK & DO IT AGAIN
UNS:    C<>B   M        ; GET ST BACK FROM B[M]
        RCR    11       ; PUT IN C[1:Φ]
        ST=C            ; RESTORE STATUS
USL:    ?A#Φ   MS       ; ANY MORE UNDERSCORES TO OUTPUT ?
         NC RTN         ; NO :
                        ;       EXIT
        LDI             ; YES : LOAD CHARACTER READY
        ΦIF    '_'
        WRABC1R         ; OUTPUT UNDERSCORE
        A=A-1  MS       ; DECREMENT UNDERSCORE COUNTER
xB4C    JNC-Φ7  USL     ; GO AND DO IT AGAIN
```

SUBROUTINE - ENTRY ROUTINE 'START'

DEPENDENCIES:

ADDDIG	DELDIG
OUTHEX	GETHEX
WRC1	GETDRS
B9>CPU	ENTRY

ROUTINES USED: MESSL, CLLCDE, GOSUB, PCTOC, LEFTJ

INPUT : NO REGISTERS
PROMPTS FOR ADDRESS OF CODE TO BE ENTERED, ACCEPTS HEX DIGITS AND DELETE KEY. R/S TERMINATES ENTRY AND SETS UP JUMP TO CODE (SBR CALL) USING WRC1 (WRITE CLASS 1 INSTRUCTION TO ROM). BUFFER 9 IS USED BY B9>CPU WHICH IS JUMPED TO FROM THIS ROUTINE.

USES : ALL CPU REGISTERS

OUTPUT : EXITS FROM CPU>B9 EVENTUALLY.

```
xBAD START:  C=0   ALL      ;LOAD M WITH 4 IN [MS]
             PT=13          ; AND REST OF WORD CLEAR
             LC    4        ; READY FOR OUTHEX TO INDICATE
             M=C            ; 4 UNDERSCORES & NO DIGITS YET
      ENT :  NC XQ CLLCDE   ; CLEAR & ENABLE LCD
             NC XQ MESSL    ; OUTPUT MESSAGE
    ALPHA:   ENTRY␣@␣       ;     'ENTRY␣@␣'
             GOSUB OUTHEX   ; THE ADDRESS (OR UNDERSCORES)
             NC XQ LEFTJ    ; AND LEFT JUSTIFY THE WHOLE LCD
             C=M            ; IF THERE ARE NO UNDERSCORES LEFT
             ?C#0  MS       ; THEN FINISH WITH EITHER R/S
             JNC+10 DONE    ; OR DELETE KEY
      KEY :  GOSUB GETHEX   ; GET A HEX KEY OR CONTROL KEY
             ?FS   2        ; SPECIAL KEY PRESSED ?
             JNC+07 NORM    ; NO: ADD THE DIGIT TO THE DISPLAY
             ?C#0  S&X      ; YES: TEST TO SEE IF IT IS THE ON
             JC+29 EXIT     ; KEY OTHERWISE IGNORE KEY
      DEL :  GOSUB DELDIG   ; IF IT WAS THE DELETE KEY DELETE
             JNC-1E ENT     ; THE LAST DIGIT
     NORM:   GOSUB ADDDIG   ; IF NORMAL HEX KEY PRESSED ADD
             JNC-22 ENT     ; THE APPROPRIATE VALUE TO DISPLAY STRING
     DONE :  GOSUB GETDRS   ; ALL 4 DIGITS INPUT SO WAIT FOR
             ?FS   4        ; DELETE OR R/S TO BE PRESSED
             JC-0D  DEL     ; IF FLAG 4 SET MUST HAVE BEEN ←
             NC XQ MSG10S   ; OTHERWISE: PRINT ENTRY ADDRESS ON
             NC XQ CLLCDE   ; ANY PRINTING DEVICES, CLEAR & ENABLE
             NC XQ ENCP00   ; THE DISPLAY, ENABLE CHIP 00 AND
             NC XQ RSTSQ    ; RESET SYSTEM FLAGS ETC.
             CF  0          ; INDICATE XQ
             CF  1          ; INDICATE NC
             C=0   ALL
             NC XQ PCTOC    ; PUT THIS ROMS PAGE NUMBER IN C[6]
             PT=5           ; PUT IN C[ADDR]
             LC    B        ;   THE ADDRESS OF
             LC    F        ;   ROUTINE 'ENTRY' WHERE
             LC    3        ;   THE INSTRUCTION IS TO BE WRITTEN
             A<>C  ALL      ; SAVE IT IN A[ADDR]
             C=M            ; GET ADDRESS FROM M AND TIDY
             C=0   MS       ; IT UP FOR WRC1 SO THAT
             RCR 1
             C=0   MS
```

```
                    M=C                 ; M[3:Ø] IS THE ENTERED ADDRESS
                    GOSUB  WRC1         ; WRITE THE INSTRUCTION TO ROM
                    JNC+   B9>CPU       ; PREPARE TO ENTER CODE UNDER TEST
        EXIT:       C=C-1  S&X          ;IF THE ON KEY WAS PRESSED THEN
                    ?C#Ø   S&X          ; ABORT START AND RETURN TO MAIN
                    JC-31  KEY          ;MENU HAVING WAITED FOR ON KEY
xB97                GOTO   MEN2         ;TO BE RELEASED
xB99
```

SUBROUTINE - GET DELETE OR R/S 'GETDRS'

```
DEPENDENCIES:   KEYDBU
ROUTINES USED:  NONE
INPUT:          NONE
OUTPUT:         FLAG 4 SET INDICATES DELETE PRESSED
                FLAG 4 CLR INDICATES R/S    PRESSED
USES:           FLAG 4 AS ABOVE
                A [S&X]
                C ALL
```

```
xB9A GETDRS:    GOSUB  KEYDBU       ;GET KEY & WAIT FOR IT TO COME UP
                LDI    ØC3          ; LOAD C[S&X] WITH VALUE FOR DELETE
                SF     4            ; INDICATE DELETE
                ?A#C   S&X          ; IF NOT DELETE SET CARRY
                NC     RTN          ;IF DELETE THEN RETURN WITH FLAG 4 SET
                CF     4            ; INDICATE NOT DELETE
                LDI    Ø87          ; LOAD C[S&X] WITH VALUE FOR R/S
                ?A#C   S&X          ;IF NOT R/S SET CARRY
                NC     RTN          ;IF R/S THEN RETURN WITH FLAG 4 CLEAR
xBA7            JNC-   GETDRS       ;ELSE GET ANOTHER KEY
```

SUBROUTINE - BUFFER 9 TO CPU & CODE ENTRY & RTN

DEPENDENCIES: MLOCB @
ROUTINES USED: GOSUB
INPUT : BUFFER 9 CONTAINS CPU STATUS
OUTPUT : CPU REGISTERS SET UP, CODE ENTERED AT 'ENTRY'
USES: ALL REGISTERS AFFECTED

```
xBA8  B9>CPU:  *
               NC XQ GOSUB      ; (MAKE &) LOCATE
               DEF  MK9         ; BUFFER 9
        CR9:   LDI              ; ADDR OF HEADER IS IN A[S&X]
               HEX  ϕϕ6         ; POINT TO TOP REGISTER
               C=C+A  S&X       ;   BY ADDING 6
               RAM  SLCT
               A<>C  S&X        ; PUT ADDR BACK IN A[S&X]
               READ  DATA
     RESTST:   ST=C             ; RESTORE STATUS FLAGS ϕ-7
               PT=  2
     RESTG:    G=C              ; RESTORE G
     PTQ:      PT=  Q           ; POSITION Q
               RCR  5           ; POSITION IS IN C[MS]
               PT=  ϕ           ; RESET POSITION
     LPQ:      ?C≠ϕ  MS         ; DONE?
               JNC+ϕ4  PTP      ; YES: DO P NEXT
               +PT              ; NO: INC PT POSITION
               C=C-1  MS        ; DEC COUNT
               JNC-ϕ4  LPQ
     PTP:      RCR  1           ; POSITION IS IN C[MS]
               PT=  P
               PT=  ϕ           ; RESET P
     LPP:      ?C#ϕ  MS         ; DONE?
               JNC+ϕ4  CR92     ; YES: CONTINUE RESTORE
               +PT              ; NO: INC PT POSITION
               C=C-1  MS        ; DEC COUNT
               JNC-ϕ4  LPP
     CR92:     RCR  1           ; PT P IS ACTIVE
               ?C#ϕ  MS         ; GET ACTIVE PT IN C[MS]
               JNC+ϕ2  CR93     ; IF NO CARRY ACTIVE PT = P ANYWAY SO JUMP
               PT=  Q           ; SELECT Q
     CR93:     CF   1ϕ          ; CLEAR HEX/DEC FLAG
               RCR  1           ; GET MODE INDICATOR
               ?C#ϕ  MS         ; SET CARRY IF DECIMAL
               JNC+ϕ2  CR94     ; JUMP IF NOT
               SF   1ϕ          ; SET FLAG 1ϕ IF DECIMAL
     CR94:     A=A-1  S&X       ; ADDR IN A[S&X]
               A<>C  S&X        ; POINT TO NEXT REGISTER
               RAM  SLCT        ;   SELECT IT
               A<>C  S&X        ;   PUT ADDR BACK IN A[S&X]
               READ  DATA       ;   AND READ VALUE
```

```
                RESTN:  N=C             ; PUT IT IN N
                        A=A-1   S&X
                        A<>C    S&X
                        RAM     SLCT
                        A<>C    S&X
                        READ    DATA
                RESTM:  M=C             ; PUT NEXT IN M
                        A=A-1   S&X
                        A<>C    S&X
                        RAM     SLCT
                        A<>C    S&X
                        READ    DATA
                RESTB:  C<>B    ALL     ; PUT NEXT IN B
                        A=A-1   S&X
                        A<>C    S&X
                        RAM     SLCT
                        A<>C    S&X
                        READ    DATA
                RESTA:  A<>C    ALL     ; PUT NEXT IN A
                        C=C-1   S&X
                        RAM     SLCT
                RESTC:  READ    DATA    ; LAST ONE IN C
                        PUSH            ; USING PART OF C [ADDR] BUT WITHOUT
                        RCR     3       ; DESTROYING IT ENABLE PERIPHERAL AND
                        C=Ø     S&X     ; CHIP ØØ
                        PRPH    SLCT
                        RAM     SLCT
                        RCR     11
                        POP             ; RESTORE C
                        ?FS     1Ø      ; RESTORE HEX/DEC MODE
                        JNC+    ENTRY   ; IF NO CARRY THEN HEX MODE - ENTER
                        SETDEC          ; OTHERWISE SET DECIMAL MODE
                        CF      1Ø      ; RESTORE FLAG 1Ø
xBF3    ENTRY:  NOP                     ; TWO WORDS WRITTEN TO BY WRCL THAT
   4            NOP                     ; CONTAIN THE TWO WORD NC XQ TO CODE UNDER TEST
   5    RTN:    CF      8               ; RTNS IN THE CODE COME BACK HERE
   6            JNC+Ø2  RB2             ; & CLEAR FLAG 8
   7    BKPT:   SF      8               ; BKPTS XQ BACK HERE & SET FLAG 8
   8    RB2:    NOP                     ; FREE SPACE
   9            NOP
   A            NOP
   B            NOP
   C            NOP
   D            NOP
   E            NOP
   F            NOP
xCØØ    CPU>B9:                         ; SEE NEXT ROUTINE
```

SUBROUTINE CPU REGISTERS > BUFFER 9

DEPENDENCIES: MLOCB @ FPT @
ROUTINES USED:
INPUT : FROM CODE UNDER TEST - ALL REGISTERS
OUTPUT : BUFFER 9 CONTAINS CPU REGISTERS IN DEFINED FORMAT

```
xCØØ  CPU>B9:  PUSH            ;SAVE C[6:3] ON STACK
               RCR 3           ;ROTATE THEM DOWN TO [3:Ø]
               C=Ø  S&X        ;USE [2:Ø] TO SELECT
               PRPH SLCT       ;     PERIPHERAL AND
               RAM  SLCT       ;     CHIP ØØ
               RCR  11         ;RESTORE TO [6:3]
               POP             ;RESTORE ORIGINAL VALUE
               WRITE 11 (a)    ;SAVE C IN (a)
               A<>C  ALL       ;SAVE A
               WRITE 12 (b)    ;        IN (b)
               CF    1Ø        ;CLEAR HEX/DECIMAL FLAG
               C=Ø   S&X       ;DO Ø-1 IN CURRENT MODE
               C=C-1 S&X
               C<>ST           ;PRESERVE STATUS W C, ANSWER IN ST
               ?FS   1         ;IF BIT 2 IS SET MUST BE IN HEX MODE
               JC+Ø2 CONT      ;  SO CONTINUE (AS RESULT MUST BE F NOT 9)
               SF    1Ø        ;OTHERWISE SET FLAG TO INDICATE DECIMAL
CONT:          C<>ST           ;RESTORE STATUS REGISTERS
               SETDEC          ;FIND BUFFER 9 (DONE LIKE THIS TO
               A=Ø   MS        ;PRESERVE THE STACK & ENSURE THAT
               A=A-1 MS        ;IF BUFFER 9 DOES NOT EXIST IT IS
               SETHEX          ;NOT AUTOMATICALLY CREATED - BUT THE
               *               ;USER IS INFORMED BY THE NO BUFFER
               NC XQ GOSUB     ;ERROR)
               DEF   MLOCB
               JNC+Ø4 CS9      ;FOUND : CONTINUE BELOW WITH ADDR IN A[S&X]
               *               ;NF: GIVE "NO BUFFER" ERROR :NOTE THIS DOES
               NC XQ GOTO      ;NOT EXIT IN THE BEST WAY SO A GTO.. REQUIRED
               DEF ERRNB       ;TO PUT EVERYTHING BACK IN ORDER !
CS9:           C=Ø   S&X       ;SELECT CHIP ØØ
               RAM   SLCT
               A=A+1 S&X       ;POINT TO 1ST REG OF BUFFER
               READ 11 (a)     ;RESTORE CPU.C
               A<>C  ALL
               RAM   SLCT      ;SELECT REGISTER
               A<>C  ALL
               WRITE DATA      ;SAVE CPU.C IN BUFFER
               C=Ø   S&X       ;SELECT CHIP ØØ
               RAM   SLCT
               A=A+1 S&X       ;POINT TO 2nd REG OF BUFFER
               READ 12 (b)     ;RESTORE CPU.A
               A<>C  ALL
               RAM   SLCT      ;SELECT REGISTER
               A<>C  ALL
               WRITE DATA      ;SAVE CPU.A IN BUFFER
               A<>C  ALL
               C=C+1 S&X
               RAM   SLCT
               C<>B  ALL
               WRITE DATA      ;SAVE CPU.B IN BUFFER
               C<>B  ALL
```

```
          C=C+1  S&X
          RAM    SLCT
          C<>M
          WRITE  DATA      ; SAVE CPU.M IN BUFFER
          C<>M
          C=C+1  S&X
          RAM    SLCT
          C<>N
          WRITE  DATA      ; SAVE CPU.N IN BUFFER
          C<>N
          C=C+1  S&X       ; POINT TO FINAL REGISTER
          C<>B   S&X       ; SAVE IN B[S&X]
          CF     12        ; FLAG 12: C=> P#Q      S => P=Q
          CF     9         ; FLAG 9:  C=> PT=P     S => PT=Q
          ?P=Q             ; IF POS(P) = POS(Q) THEN SET CARRY
          JNC+Ø3 PNQ       ; NOT THE SAME SO DON'T
          +PT              ;   INCREMENT POINTER
          SF     12        ; IF SAME INC PT & SET FLAG 12
PNQ:      *
          NC XQ  GOSUB
          DEF    FPT       ; FIND POSITION OF ACTIVE (PT) => C[MS]
          A<>C   MS                                     => A[MS]
          PT=P             ; SELECT P
          *
          NC XQ  GOSUB
          DEF    FPT       ; FIND POSITION OF P  => C[MS]
          ?A#C   MS        ; IF POS(P)# POS(PT) THEN SET CARRY
          JNC+Ø2 CLPT      ; IF SAME CONTINUE BELOW
          SF     9         ;   OTHERWISE INDICATE PT=Q ACTIVE
CLPT:     A<>C   MS        ; PUT POS(P) IN A[MS]
          PT=Q             ; SELECT Q
          *
          NC XQ  GOSUB
          DEF    FPT       ; FIND POSITION OF Q => C[MS]
          ?FS    12        ; IF P=Q ORIGINALLY THEN SET CARRY
          JNC+Ø5 DLPT      ; IF THEY WEREN'T THEN END
          ?FS    9         ; IF PT=Q ORIGINALLY THEN SET CARRY
          JNC+Ø2 PTP       ; IF PT WAS P JUMP
PTQ:      A<>C   MS        ; SWAP POS(P) & POS(Q) (WHICH WAS INCREMENTED)
PTP:      A=C    MS        ; SET POS(P) = POS(Q)
DLPT:     CF     12        ; END PT PART
          C<>B   S&X       ; SELECT TOP REGISTER OF BUFFER
          RAM    SLCT
          C<>B   S&X
          PT=    12        ; CLEAR C[12:Ø]
          C=Ø    WPT (R←)  ;   LEAVING C[13] UNCHANGED
          RCR    1         ; Q IN C[12]
          A<>C   MS        ; P IN C[13]
          RCR    1         ; P IN C[12], Q IN C[11]
          ?FS    9
          JNC+Ø2 CWS1      ; IF FLAG 9 CLEAR INDICATE PT=P (Ø)
          C=C+1  MS        ; IF FLAG 9 SET INDICATE PT=Q (1)
CWS1:     RCR    1         ; C[12]=PT, C[11]=P, C[1Ø]=Q
          ?FS    1Ø        ; IF DEC SET CARRY
          JNC+Ø3 CWS2      ; IF HEX LEAVE CLEAR C[12]=Ø
          C=C+1  MS        ; IF DEC SET C[13]=1
          CF     1Ø        ; CLEAR FLAG READY FOR EXIT
```

```
CWS2:   RCR  6          ; [7] = HEX/DEC      [6] = PT  P/Q
        C=C+1  MS       ; [5] = P            [4] = Q
        PT=2            ; [13] = 1           [12] = UNUSED
        C = G           ; [3:2] = G          [11:8] = STK2
        C = ST          ; [1:0] = ST
        WRITE  DATA     ; WRITE FINAL REGISTER TO BUFFER
xC74    GOTO   CE9      ; LOOK FOR BKPT/NORMAL RETURN
```

MAKE BUFFER 9 SUBROUTINE - MK9

DEPENDENCIES: MMKBF
ROUTINES USED: NONE
INPUT: NONE
OUTPUT: BUFFER 9
USES: ALL REGISTERS

```
xC9A    MK9:    C=0    ALL      ; CREATES BUFFER 9 IF IT DOES NOT
                PT=13           ; ALREADY EXIST. CHECKS FOR CORRECT
                LC  9           ; SIZE AND FALLS THROUGH TO
                LC  9           ; FIND9 TO FIND IT AUTOMATICALLY
                LC  0
                LC  7
                N=C
xCA3            GOSUB  MMKBF
```

FIND BUFFER 9 SUBROUTINE - FIND9

```
xCA4    FIND9:  SETDEC          ; FINDS BUFFER 9. IF IT CANNOT
                A=0             ; BE FOUND THEN EXITS VIA ERRNB
                A=A-1  MS
                SETHEX
                GOSUB  MLOCB
                RTN
xCAC    ERRNB:  NC  XQ  ERRSUB 22E8 ; GIVES 'NO BUFFER' ERROR
                NC  XQ  CLLCDE 2CF0 ; CLEAR & ENABLE LCD
                NC  XQ  MESSL  07EF ; OUTPUT ERROR MESSAGE
        ALPHA:: NO BUFFER
                NC  XQ  LEFTJ  28F7 ; LEFT JUSTIFY DISPLAY
                NC  XQ  MSG105 1C80 ; PRINT MESSAGE ON TRACE PRINTERS
xCC0            NC  GO  ERR110 22FB ; ERROR EXIT
```

FIND USER FLAG SUBROUTINE - FINDF

DEPENDENCIES: NONE
ROUTINES USED: INTINT
ENCPØØ
INPUT: BCD FLAG NUMBER IN M[2:1] FOR FINDF
BINARY FLAG NUMBER IN A[S&X] FOR IF
OUTPUT: LEAVES A CONTAINING d REGISTER AND APPROPRIATE
BIT MASK IN C . B HOLDS C AND A.
ADD BIT MASK (IF B IS ALL ZERO) TO SET BIT (CLR → SET)
SUBTRACT BIT MASK (IF B IS NOT ALL Ø) TO CLEAR BIT (SET → CLR)
USES: A,B,C, M , d

```
xC75  FINDF:  C=M          ;GET BCD FLAG NUMBER INTO C[2:1]
              RCR 2        ;PUT MSN IN C[Ø] AND LSN IN
              NC XQ INTINT ;C[MS]. COMBINE TWO INTO C[1:Ø] BINARY
              A=C S&X      ;BINARY FLAG NUMBER IN A[S&X]
xC7A  IF:     NC XQ ENCPØØ ;ENABLE CHIP ØØ (STATUS REGISTERS)
              LDI ØØ8
              C<>B S&X     ;8 IN B[S&X]
              C=Ø ALL      ;CLEAR C
              C=C+1 S&X    ;SET BOTTOM BIT TO 1
              RCR 2        ;MOVE TO NEXT BYTE AND SUBTRACT 8
              A=A-B S&X    ;FROM FLAG NUMBER EACH TIME
              JNC-Ø2       ;IF NOT THERE YET MOVE TO NEXT BYTE
              JNC+Ø2       ;CARRY: DON'T SHIFT FOR FIRST FLAG
              C=C+C ALL    ;SHIFT BIT ONE PLACE LEFT
              A=A+1 S&X    ;ADD 1 TO (-ve) VALUE UNTIL THERE
              JNC-Ø2       ;IS A CARRY BACK TO ZERO
              A=C ALL      ;PUT MASK INTO A
              READ d       ;SET FLAG REGISTER
              C=C AND A    ;'AND' THE FLAGS & MASK
              C<>B ALL     ;PUT RESULT IN B
              READ d       ;GET FLAGS & PUT THEM IN A[ALL]
              A<>C ALL     ; WHILST MASK GOES INTO C[ALL]
xBE           RTN
```

GET HEXADECIMAL KEY PRESS IN A,B & C - GETHXA

DEPENDENCIES : GETHEX
ROUTINES USED : NONE
INPUT : NONE
OUTPUT : DECODE VALUE (AS GETHEX) IN A,B,C [S&X]
WAITS FOR KEY TO COME UP BEFORE EXITING
USES : ALL REGISTERS & FLAGS AS GETHEX

```
xC8F  GETHXA: GOSUB GETHEX  ;GET HEX KEY PRESS
              A=C   S&X     ;IN ADDITION TO B & C PUT IT IN A[S&X]
              CLRKEY        ;WAIT FOR KEY TO COME UP
              ?KEY
              JC-Ø2
xC96          RTN
```

SUBROUTINE - WRITE CLASS 1 INSTRUCTION - WRC1

DEPENDENCIES: NONE
ROUTINES USED: NONE
INPUT: M [3:Ø] CONTAINS ADDRESS FOR CLASS 1 INSTRUCTION
A [ADDR] CONTAINS ADDRESS THAT WILL HOLD THE INSTRUCTION
FLAG Ø CLEAR: XQ | SET: GO
FLAG 1 CLEAR: CARRY CLEAR | SET: CARRY SET
OUTPUT: M UNCHANGED
A [ADDR] DESTROYED
C [ALL] 2ND WORD OF INSTRUCTION
FLAGS Ø & 1 UNCHANGED
ADDR SPECIFIED IN A[ADDR] CONTAINS 1st WORD
" " " " +1 " 2nd WORD

```
xCC2  WRC1:  C=M          S&X     ; GET LOWER BYTE
             C=Ø          XS
             C=C+C        S&X
             C=C+C        S&X     ; MULTIPLY BY 4 TO ARSHIFT LEFT 2 BITS
             C=C+1        S&X     ; SET BIT Ø TO GIVE CLASS 1
             A<>C         M       ; GET ADDR
             WROM                 ; WRITE 1st WORD
             C=C+1        M       ; INCREMENT
             A<>C         M       ;    AND SAVE ADDRESS
             C=M                  ; GET
             RCR          2       ;    UPPER BYTE
             C=Ø          XS
             C=C+C        S&X
             C=C+C        S&X     ; ARITHMETIC SHIFT LEFT 2 BITS
             ?FS          Ø       ; IF 'GO' THEN SET CARRY
             JNC+Ø3       CARRY   ; IF NO CARRY - JUMP
             C=C+1        S&X     ; ADD 2 (SET BIT 1) TO GIVE 'GO'
             C=C+1        S&X     ;          RATHER THAN 'XQ'
        CARRY: ?FS        1       ; IF 'C' THEN SET CARRY
             JNC+Ø2       WRITE   ; IF NO CARRY WRITE WORD
             C=C+1        S&X     ; ELSE ADD 1 (SET BIT Ø) TO GIVE 'C'
        WRITE: A<>C       M       ;          RATHER THAN 'NC'
             WROM                 ; WRITE 2nd WORD OF INSTRUCTION
xCD8         RTN                  ; END
```

SUBROUTINE - DELETE DIGIT - DELDIG

DEPENDENCIES: NONE
ROUTINES USED: NONE
INPUT: M CONTAINS NUMBER OF UNDERSCORES IN [MS]
NUMBER OF DIGITS IN [Φ]
DIGITS IN [1] UPWARDS
OUTPUT: AS INPUT BUT WITH DELETED DIGIT

```
xCE5  DELDIG:  C=M          ; GET DIGIT ENTRY
               PT=Φ
               ?C#Φ  PT     ; IF NO DIGITS PUT IN YET
               NC    RTN    ;   THEN IGNORE DELETE REQUEST & EXIT
               A=C   MS     ; SAVE NUMBER OF PROMPTS IN A[MS]
               C=Φ   MS
               A=C   PT     ; SAVE NUMBER OF DIGITS IN A[Φ]
               RCR   1      ; REMOVE LAST DIGIT
               A<>C  MS     ; RESTORE
               A<>C  PT
               C=C-1 PT     ; ONE LESS DIGIT
               C=C+1 MS     ; ONE MORE PROMPT
               M=C          ; RESTORE M
xCF2           RTN
```

SUBROUTINE - ADD DIGIT - ADDDIG

DEPENDENCIES, ROUTINES USED, INPUT AS ABOVE
OUTPUT: AS INPUT BUT WITH DIGIT FROM B[Φ] ADDED

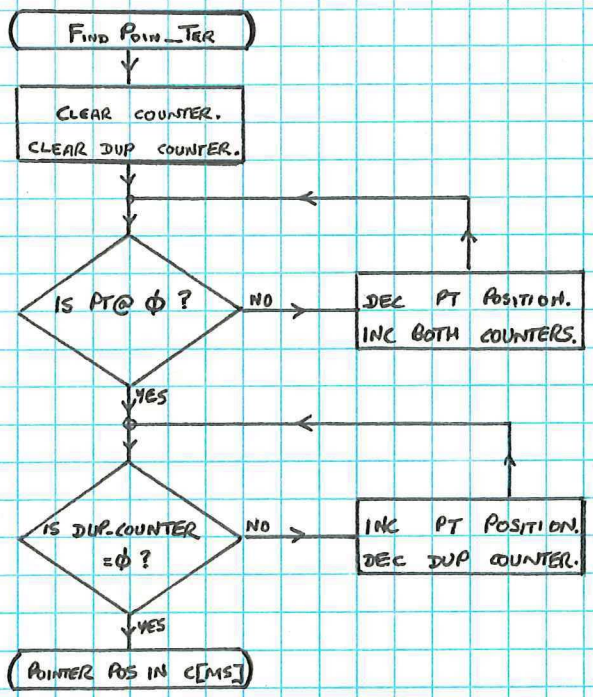
```
xCD9  ADDDK:   C=M          ; GET DIGIT ENTRY
               PT=Φ
               A=C   MS     ; SAVE NUMBER OF PROMPTS
               A=C   PT     ;   AND NUMBER OF DIGITS
               C=B   PT     ; PUT IN NEW DIGIT FROM B[Φ]
               RCR   13     ;   AND MOVE LEFT 1 DIGIT
               A<>C  MS     ; RESTORE
               A<>C  PT
               C=C+1 PT     ; ONE MORE CHARACTER
               C=C-1 MS     ; ONE LESS PROMPT
               M=C          ; RESTORE M
xCE4           RTN
```

FIND POINTER SUBROUTINE - FPT

DEPENDENCIES	:	NONE
ROUTINES USED	:	NONE
INPUT	:	NONE
OUTPUT	:	C[XS] CLEAR
		C[MS] HEX DIGIT REPRESENTING POSITION OF ACTIVE POINTER
		PT's ACTIVE POINTER, P and Q UNCHANGED
REGISTERS USED	:	C[XS], C[MS] ONLY

FPT :

```
xCF3  FPT:   C=0   MS      ; CLEAR COUNTER IN C[MS]
   4         C=0   XS      ; CLEAR DUP COUNTER IN C[XS]
   5  LFPT:  ?PT=0          ; SET CARRY IF PT=0
   6         JC+05 DFPT    ; PT @ 0, PUT IT BACK & END
   7         -PT           ; DEC PT POSITION
   8         C=C+1 MS      ; INC COUNTERS
   9         C=C+1 XS
   A         JNC-05 LFPT
   B  DFPT:  ?C#0  XS      ; POSITION IN C[MS] & C[XS], USE
   C         NC RTN        ;  C[XS] TO PUT PT BACK TO CORRECT
   D         +PT           ; POSITION USING THIS LOOP AND
   E         C=C-1 XS      ; ENDING WHEN C[XS] IS ZERO
xCFF         JNC-04 DFPT
```



FROM HERE DOWN RESTORES PT TO ORIGINAL POSITION USING THE DUPLICATE COUNTER AND LEAVING ORIGINAL COUNTER UNCHANGED.

LOCATE BUFFER SUBROUTINE - MLOCB

```
                DEPENDENCIES:   NONE
                ROUTINES USED:  NONE
                INPUT:          BUFFER ID IS IN A[MS]
                OUTPUT:         BUFFER HEADER SELECTED AND IN C[ALL] AND
                                ADDRESS OF HEADER IN A[S&X] IF FOUND, ELSE SKIPS STEP
                USES:           C[ALL], A[ALL]
                                ALL OTHER REGISTERS, FLAGS & POINTERS UNCHANGED
                ASSUMES:        HEXMODE

xdØØ    MLOCB:      LDI   ØBF       ;BUFFER AREA STARTS AT ØCØ
                    A=C   S&X       ;SAVE REGISTER ADDRESS IN A[S&X]
                INCADR: A=A+1 S&X   ;INCREMENT RAM ADDRESS
                CONTLB: C=Ø   S&X   ;ENABLE CHIP ØØ  TURN OFF PERIPHERALS
                        PRPH  SLCT  ;CANNOT CALL SBR HERE AS STACK
                        RAM   SLCT  ;MUST BE LEFT UNCHANGED
                        READ  13(c) ;LOAD USER STATUS c INTO C
                        ?A<C  S&X   ;IF RAM ≥ .END. THEN NO SUCH BUFFER
                        JNC+11 NOBUFR;                AND NO ROOM FOR ONE
                        A<>C  S&X   ;COPY NEW RAM ADDR INTO C[s&x]
                        A=C   S&X   ;                      AND A[s&x]
                        RAM   SLCT  ;SELECT THIS REGISTER AND
                        READ  DATA  ;     READ IT INTO C
                        ?C#Ø  ALL   ;IF ZERO THEN NO MORE BUFFERS
                        JNC+ØB NOBUFR;      (BUT AT LEAST 1 FREE REGISTER)
                        C=C+1 MS    ;INCREMENT TOP NYBBLE TO GET A
                        JC-ØE INCADR; CARRY IF STILL IN KEY AREA
                        C=C-1 MS    ;RESTORE IT TO CORRECT VALUE
                        RCR   13    ;PUT LOWER NYBBLE OF ID BYTE INTO
                                    ; C[MS] AS O/S MAY HAVE SET TOP TO Ø
                        ?A#C  MS    ;IS IT THE BUFFER WERE LOOKING FOR
                        JNC+Ø8 LBEND;END IF IT IS
                        RCR   11    ;OTHERWISE GET SIZE INTO C[s&x]
                        C=Ø   XS    ;ENSURING C[XS] IS CLEAR
                        A=A+C S&X   ;ADD SIZE & RAM ADR TO GIVE NEXT HEADER
                        JNC-15 CONTLB;AND CONTINUE AT NEW HEADER
                NOBUFR: POP         ;IF NOT FOUND THEN RETURN TO
                        C=C+1 M     ;     RETURN ADDRESS + 1
                        GOTO  ADR
                LBEND:  RCR   1     ;FOUND BUFFER SO RESTORE IT AND
xDIE                    RTN         ;RETURN TO NORMAL RETURN ADDRESS
```

USER FUNCTION - LCBUF

DEPENDENCIES: MLOCB
ROUTINES USED: BCDBIN
ERRNE
ENCPØØ
APNDNW

USER INPUT : BUFFER ID VALUE Ø - 15 (INCLUSIVE) IN X REGISTER
USER OUTPUT : X - UNCHANGED
ALPHA - LAST TWO DIGITS ARE ADDRESS OF BUFFER
IN THE FORM TO BE USED BY NRCLM, RAMED etc.
ERRORS: ALPHA DATA - IF X CONTAINS NON-NUMERIC DATA
NONEXISTENT - IF BUFFER DOES NOT EXIST OR
VALUE IN X IS OUT OF RANGE Ø - 15

```
xDIF   Ø86      F
       Ø15      U
       ØØ2      B
       ØØ3      C
       ØØC      L
xD24   LCBUF :  READ    3(x)      ;GET BUFFER NUMBER
                NC XQ   BCDBIN    ;CONUERT TO BINARY (ERRNE IF x>999 etc)
                RCR     1         ;PUT LSN INTO C[MS]
                C=Ø     XS        ;CLEAR C[XS]
                ?C#Ø    S&X       ;IF ID>15 THEN
                JC+Ø6   ELBC      ;   GIUE ERROR - NONEXISTENT
                A=C     MS        ;PUT THE VALID ID INTO A[MS]
                GOSUB   MLOCB     ;LOCATE THE BUFFER
                JNC+Ø3  UBC       ;FOUND IT - SO CONTINUE BELOW
       ELBC:    NC GO   ERRNE     ;NOT FOUND - GIUE ERROR - NONEXISTENT
       UBC:     NC XQ   ENCPØØ    ;DESELECT BUFFER HEADER ID
                A<>C    S&X       ;GET RAM ADDR IN C[S&X]
                C=Ø     M         ;ENSURE NO RUBBISH IN C[3]
                PT=2              ;GET TOP BYTE INTO G
                N=C               ;SAUE REST IN N
                G=C
                NC XQ   APNDNW    ;APPEND (WITH NO WARNINGS) TOP BYTE
                C=N               ; INTO ALPHA REGISTER
                PT=Ø
                G=C               ;GET BOTTOM BYTE INTO G
xD3F            NC GO   APNDNW    ;APPEND (WITH NO WARNING) BOTTOM BYTE
```

MAKE & INITIALISE BUFFER - MKBUF USER FUNCTION
- MMKBF M-CODE SUBROUTINE

MKBUF

INPUT FROM USER: M REGISTER CONTAINS BUFFER HEADER
OUTPUT TO USER: BUFFER CREATED & REGISTERS WITHIN IT SET TO '10 00 00 00 00 00 00' (EXCEPT HEADER)
ERRORS: TRY AGAIN - NOT ENOUGH ROOM
DATA ERROR - SIZE INVALID EITHER BECAUSE BUFFER WITH SPECIFIED ID ALREADY EXISTS WITH A DIFFERENT SIZE OR BECAUSE SIZE WAS SPECIFIED AS Ø.

MMKBF

DEPENDENCIES: MLOCB
ROUTINES USED: ENCPØØ PACKE ERRDE
INPUT: BUFFER HEADER IN N
OUTPUT: AS MKBUF
ERRORS: AS MKBUF

```
;NOTE - IF BUFFER EXISTS AND IS CORRECT SIZE IT IS NOT CLEARED.
;      IF IT IS THE WRONG SIZE 'DATA ERROR' IS DISPLAYED.
;      IF IT DOES NOT EXIST, IT IS CREATED AND CLEARED i.e
;         ALL REGISTERS IN IT ARE SET TO '10 00 00 00 00 00 00'
;         EXCEPT FOR THE HEADER WHICH IS AS SPECIFIED

xDØØ  FUNCTION  MKBUF                ;FUNCTION NAME
xD45  MKBF:     READ   5(M)          ;GET HEADER FROM ALPHA M REGISTER
                N=C                  ;STORE IN N
xD47  MMKBF:    C=N                  ;GET HEADER FROM N
                A=C      MS          ;PUT ID READY FOR MLOCB
                RCR      1Ø          ;CHECK BUFFER SIZE ≠ Ø AND
                C<>B     S&X         ;  PUT IT INTO B[S&X]
                B=Ø      XS
                ?B#Ø     S&X
          EMM:  NC GO    ERRDE       ;IF SIZE=Ø OR INVALID SIZE 'DATAERROR'
                GOSUB    MLOCB       ;FIND BUFFER
                JNC+11   CHKSIZ      ;IF IT EXISTS CHECK ITS SIZE
                A=A+B    S&X         ;ELSE CHECK THERE IS ENOUGH RAM
                A=A-1    S&X         ;   FOR A BUFFER THIS SIZE
                NC XQ    ENCPØØ
                READ     13(C)
                ?A<C     S&X         ;IF (REG+SIZE-1) ≥ .END. THEN THERE
                NC GO    PACKE       ;   IS NOT ENOUGH ROOM SO 'PACK'
                A=A+1    S&X         ;   AND 'TRY AGAIN'
                A=A-B    S&X         ;ELSE PUT START OF BUFFER
                A<>C     S&X         ;   INTO C[S&X] AND A[S&X]
                A=C      S&X
                RAM      SLCT        ;SELECT (WRITE) REGISTER TO PUT
                C=N                  ;   HEADER IN
                WRITE    DATA
                JNC+Ø7   CLRBF1      ;CLEAR BUFFER
```

```
                CHKSIZ8     A<>B     S&X     ; C CONTAINS HEADER, B[S&X]=ADDR, A[S&X]=
                            RCR      1Ø      ; C[1:Ø]=PRESENT SIZE    (REQUIRED SIZE
                            C=Ø      XS      ; C[S&X]=    "
                            ?A#C     S&X     ; COMPARE PRESENT & REQUIRED SIZE
                            JC-1A    EMM     ; IF THEY ARE NOT THE SAME GIVE 'DATA ERROR'
                            JNC+Ø5   EXIT    ; IF CORRECT THEN EXIT
                CLRBF1 :    C<>B     ALL     ; A[S&X]=ADDR, B->C[S&X]=SIZE
                CLRBF2 :    C=C-1    S&X     ; DECREMENT SIZE
                            C<>B     ALL     ; RESTORE SIZE TO B
                            ?B#Ø     S&X     ; IF  ZERO THEN ALL DONE
                EXIT :      NC  GO   ENCPØØ  ; SO END WITH CHIP ØØ ENABLED
                            A=A+1    S&X     ; OTHERWISE INCREMENT ADDRESS
                            C=Ø      ALL     ; CREATE 'BLANK' REGISTER CONTENTS
                            C=C+1    MS      ;      '10 00 00 00 00 00 00'
                            A<>C     ALL     ; SELECT REGISTER
                            RAM      SLCT    ;        TO WRITE TOO
                            A<>C     ALL     ; PUT ADDR BACK & GET 'BLANK'
                            WRITE    DATA    ; WRITE 'BLANK' BUFFER REGISTER
xD76                        JNC-ØD   CLRBF1  ; GO BACK & CONTINUE UNTIL ALL DONE
```

USER FUNCTION - BUF?

DEPENDENCIES : MLOCB @ xDØØ
ROUTINES USED : NOSKP, SKP, BCDBIN, GOSUB
INPUT : BUFFER ID (SINGLE) DIGITS Ø-15 IN X
OUTPUT : ID IS UNAFFECTED IN X
IF THE BUFFER EXISTS - DISPLAYS 'YES' FROM KEYBOARD
OR DOES NOT SKIP NEXT LINE IN PROGRAM.
IF THE BUFFER DOES NOT EXIST - DISPLAYS 'NO' OR SKIPS NEXT LINE.

```
xD77          ØBF   ?:
   8          ØØ6   F
   9          ØØ15  U
   A          ØØ2   B
xD7B  BUF?    READ 3 (X)          ; GET BUFFER NUMBER FROM X
   C          NC XQ BCDBIN        ; CONVERT TO BINARY
   E          RCR 1               ; PUT INTO [MS]
   F          C=Ø XS              ; ENSURE LESS
  BØ          ?C#Ø S&X            ;      THAN 16
   1          JC+Ø6 SK            ; BUFFER CANNOT EXIST IF X>15
   2          A=C MS              ; ENTRY PARAMETER FOR MLOCB
   3          GOSUB MLOCB         ; LOCATE BUFFER
   6          JNC+Ø3 NSK          ; DON'T SKIP IF IT EXISTS
   7  SK:     NC GO SKP           ; SKIP IF IT DOESN'T
xD89/A NSK:   NC GO NOSKP         ; NO SKIP
```

USER FUNCTION - CLRBUF

DEPENDENCIES : MLOCB @ xDØØ , CLRBUF2 @ xD6A {Part of MMIKBF}
ROUTINES USED : BCDBIN, GOSUB, ERRNE
INPUT : BUFFER ID (SINGLE) DIGITS Ø-15 IN X
OUTPUT : ID IS UNAFFECTED IN X
IF THE BUFFER SPECIFIED DOES NOT EXIST - GIVES ERROR 'NONEXISTENT'
OTHERWISE - WRITES ' 1Ø ØØ ØØ ØØ ØØ ØØ ØØ' INTO EACH REGISTER IN
THE BUFFER EXCEPT THE HEADER WHICH IS UNCHANGED.

```
xD8B          Ø86   F:
   C          Ø15   U
   D          ØØ2   B
   E          Ø12   R
   F          ØØC   L
  9Ø          ØØ3   C
   1  CLRBUF: READ 3 (X)          ; GET BUFFER NUMBER FROM X
   2          NC XQ BCDBIN        ; CONVERT TO BINARY
   4          RCR 1               ; PUT INTO [MS]
   5          C=Ø XS              ; ENSURE LESS
   6          ?C#Ø S&X            ;      THAN 16
   7          JC+Ø6 ERCLB         ; BUFFER CANNOT EXIST IF X>15
   8          A=C MS              ; ENTRY PARAMETER FOR MLOCB
   9          GOSUB MLOCB         ; LOCATE BUFFER
   C          JNC+Ø3 CLBC         ; FOUND SO CLEAR IT
   D  ERCLB:  NC GO ERRNE         ; NOT FOUND GIVE 'NONEXISTENT'
   F  CLBC:   RCR 1Ø              ; ADDR IS IN A[S&X]
  AØ          C=Ø XS              ; SIZE IS IN C[S&X]
xDA1          JNC-37 CLRBUF2      ; CLEAR BUFFER
```

USER FUNCTION - DELBUF

```
DEPENDENCIES  :  MLOCB @ xDØØ , ERCLB @ xD9D
ROUTINES USED :  BCDBIN, GOSUB, ENCPØØ, PKIOAS
INPUT         :  BUFFER ID (SINGLE) Ø-15 IS IN X
OUTPUT        :  ID IS UNAFFECTED IN X
                 IF THE BUFFER EXISTS IT IS DELETED AND THE AREA
                   PACKED TO FREE THE PREVIOUSLY USED REGISTERS
                 OTHERWISE 'NONEXISTENT' IS DISPLAYED AS NORMAL ERROR.

xDA2                  ØB6   F:
   3                  Ø15   U
   4                  ØØ2   B
   5                  ØØC   L
   6                  ØØ5   E
   7                  ØØ4   D
xDA8  DELBUF:         READ 3 (x)          ; GET BUFFER NUMBER OUT OF X
   9                  NC XQ BCDBIN        ; CONVERT TO BINARY
   B                  RCR   1             ; PUT INTO [MS]
   C                  C=Ø   XS            ; ENSURE LESS
   D                  ?C#Ø  S&X           ;      THAN 16
   E                  JC-11 ERCLB         ; BUFFER CANNOT EXIST IF X>15
   F                  A=C   MS            ; SET UP ID
  BØ                  NC XQ GOSUB         ; LOCATE BUFFER AND RETURN
   2                  DEF   MLOCB         ; WITH HEADER REGISTER SELECTED
   3                  JNC+Ø2 DLBC         ; FOUND SO CLEAR IT'S MSNybble FOR DELETION
   4                  JNC-17 ERCLB        ; GIVE ERROR AS IT'S NOT FOUND
   5  DLBC:           C=Ø   MS            ; CLEAR MSNybble
   6                  WRITE DATA          ; WRITE IT BACK
   7                  NC XQ ENCPØØ        ; ENABLE CHIP ØØ
xDB9/A                NC GO PKIOAS        ; PACK I/O & ASN AREA
```

HEADER FOR DEVELOPEMENT FUNCTIONS

```
DEPENDENCIES  :  NONE
ROUTINES USED :  NONE
INPUT         :  NONE
OUTPUT        :  NONE

xØ8D                  Ø93   S:
   E                        N
   F                        F
   9Ø                       _
   1                        V
   2                        E
   3                        D
   4                        -
xØ9S  HEADER2:        RTN                 ; TOTALLY NOTHINGNESS !
```

SUBROUTINE - USER FLAG EDITOR.

DEPENDENCIES: OUTHEX, GETHEX, ADDDIG, FINDF, MEN2, DELDIG

ROUTINES USED: CLLCDE, LEFTJ, ANN+14, MESSL, MSGIØ5

INPUT: CORRECT USER FLAG STATUS IN REGISTER d

OUTPUT: EDITED STATUS IN REGISTER d
ANY FLAGS EDITED HAVE STATUS AUTOMATICALLY PRINTED (WITH PRINTER IN TRACE MODE)

```
xDBB  FLAGE:  C=Ø   ALL          ; M holds information for
              C=C+1 MS           ; OUTHEX - start off with
              C=C+1 MS           ; two underscores
              M=C
        FLAGS: NC XQ  CLLCDE     ; Clear & enable LCD
              NC XQ   MESSL      ; Push message into right
        ALPHA:: FLAG␣            ; of display
              GOSUB   OUTHEX     ; Output flag number and/or underscores
              C=M                ; Get number of digits displayed
              PT=Ø
              C=C-1   PT         ; Subtract two
              C=C-1   PT
              ?C#Ø    PT         ;If result ≠ Ø then don't indicate flag
              JC+15   FDSP       ; SET/CLR
              GOSUB     FINDF    ;Find the flag bit we're looking at
              NC XQ     ENLCD    ;Turn on the display ready for SET/CLR
              ?B#Ø      ALL      ;If any bit is set then the flag
              JC+Ø8     FSET     ; is set so jump off and say so
              NC XQ     MESSL    ; otherwise indicate flag is clear
        ALPHA:: ␣CLR
              JNC+Ø7    FDSP
        FSET:  NC XQ    MESSL     ;Indicate flag is set
              ␣SET
        FDSP:  NC XQ    ENKEY     ;Enable
              NC XQ     LEFTJ     ; and left justify display
        FLOOP: GOSUB    GETHEX    ;Get hexadecimal keypress
              A=C       S&X       ;Store in A[S&X] as well as B&C
              ?FS       2         ; Set carry if special key pressed
              JNC+17    FNORM     ; If not set then just an ordinary key
              ?C#Ø      S&X ←     ; If delete key pressed then jump
              JNC+31    FDEL      ; to delete routine
              LDI       ØØ2 R/S   ; If R/S key pressed the toggle
              ?A#C      S&X       ; current flag if there is one
              JNC+2E    FRS
              C=C-1     S&X
              ?A#C      S&X ON    ; If not ON pressed then get
              JC-ØE     FLOOP     ; another key
              C=M                 ; If ON pressed then check for
              PT=Ø                ; final printing of the flag status
              C=C-1     PT        ; and only print it if there are
              C=C-1     PT        ; two digits on the screen
              ?C#Ø      PT        ; If there are are two then print
              JC+Ø5     WAYOUT    ; if only one then don't bother
```

```
                SF  8           ; Print what is on the screen
                NC  XQ  MSG1Ø5
                CF  5
WAYOUT:         GOTO    MEN2    ; Jump back into DEBUG menu
FNORM:          LDI     ØØA     ; If a normal key is pressed then
                ?A<C    S&X     ; only allow Ø..9 (input is decimal)
                JNC-1F  FLOOP   ; If A..F pressed then ignore key
                C=M             ; Get the number of digits displayed
                PT=Ø            ; If one digit then don't jump
                C=C-1   PT      ; else
                ?C#Ø    PT
                JC+2E   FISTD   ; jump to fist digit input
                PT=1            ; 2nd digit input : if first
                C=M             ; digit is less than 5 then
                A=C     ALL     ; Ø..9 are valid as second
                LDI     Ø5Ø     ; key pressed so go to Ø..9 bit
                ?A<C    PT
                JC+Ø7   FØ9
FIDUP:          LDI     ØØ6     ; Accept Ø..5 only
                A<>B    S&X
                ?A<C    S&X
FLP2:           JNC-3Ø  FLOOP   ; anything else won't do
                A<>B    S&X
FØ9:            GOSUB   ADDDIG  ; Accept Ø..9 & update display register
FCON:           GOTO    FLAGS   ; Go back and update display itself
FDEL:           JNC+12  FDL     ; Jump won't reach so this one does
FRS:            C=M             ; R/S pressed so check to see if
                PT=Ø            ; there is a complete flag number
                C=C-1   PT      ; to work on
                C=C-1   PT
                ?C#Ø    PT
                JC-ØE   FLP2    ; If not ignore R/S and get another key
                GOSUB   FINDF   ; Valid number so get flag status
                ?B#Ø    ALL     ; If not clear set the carry
                JNC+Ø3          ; If clear then add mask (i.e set bit)
                C=A-C   ALL
                JNC+Ø2
                C=A+C   ALL     ; else subtract mask (i.e clear bit)
                NC  XQ  ANN+14  ; Update d register and display annunciators
                JNC-14  FCON    ; Redisplay flag information
FDL:            C=M             ; Check if there are any digits on the
                PT=Ø            ; display
                ?C#Ø    PT      ; If not then re-join the main loop
                JNC-1D  FLP2
                GOSUB   DELDIG  ; else delete the last digit
                JNC-1C  FCON    ; and re-join the loop
FISTD:          C=M             ; First digit input so check to
                PT=Ø            ; see if there are two digits on
                C=C-1   PT      ; the screen already and if there
                C=C-1   PT      ; are then print the display
                ?C#Ø    PT
                JC+Ø5   FIST2   ; Otherwise fist flag number being input
                SF      8       ; so don't print 'FLAG--'
                NC  XQ  MSG1Ø5
                CF      5
```

```
FIST2:  C=0     ALL     ; Then clear the last flags data
        C=C+1   MS      ; and allow input of 0..5 in
        C=C+1   MS      ; the same way as for a second
        M=C             ; digit following a 5 first digit
xE49    JNC-34  FIDUP
```

RETURN FROM BREAKPOINT - CE9, RBKPT

DEPENDENCIES: MENU, ERRNB, MLOCB, FIND9, OUTHEX, KEYDBU

ROUTINES USED: MESSL, ENLCD, LEFTJ, MSG10S

INPUT: From CPU > B9, tests to see if return from user code was a breakpoint (Flag 8 set) or a normal return. If flag 8 is set, displays stack and breakpoint information otherwise jumps immediately back to menu.

OUTPUT: If return from breakpoint then displays (& prints) bottom two stack values and PC address.

```
xE4A  CE9 :   ?FSET  8            ;If flag 8 is clear jump immediately
              JC+Ø4  RBKPT        ;    to the main menu else display
              GOTO   MENU         ;    breakpoint info
        RBKPT:SETDEC              ;Locate buffer 9
              A=Ø    MS           ;    Note could not use FIND9 as
              A=A-1  MS           ;    it would destroy one of the values
              SETHEX              ;    on stack
              GOSUB  MLOCB
              JNC+Ø4 RBKPT1
              GOTO   ERRNB        ;Give 'No BUFFER' error if not found
       RBKPT1:RCR    11           ;Put buffer header ready for BKPT address
              POP                 ;Pop breakpoint address off stack
              C=C-1  M
              C=C-1  M            ;Subtract 2 to give correct word
              RCR    4            ;Rotate header for STK 1
              POP                 ;Pop STK1 off stack
              RCR    13           ;Put into header
              WRITE  DATA         ;Write new header containing BKPT and STK1
              LDI    ØØ6          ;Get top buffer register ready for STK2
              C=C+A  S&X
              RAM    SLCT
              READ   DATA
              RCR    5
              POP                 ;Get STK2 off stack
              RCR    9            ;Write back register to
              WRITE  DATA         ;    buffer
       DSTK2 :NC     XQ   CLLCDE  ;Prepare to display STK2
              NC     XQ   MESSL
       ALPHA::STK2␣@␣             ;Message will read  STK2 ␣@␣ wxyz
              GOSUB  FIND9        ;Find buffer with STK2
              LDI    ØØ6
              C=C+A  S&X
              RAM    SLCT
              READ   DATA
              RCR    7            ;Prepare value for output with
              R=Ø
              LD@R   4            ;    4  digits
              LD@R   Ø            ;    Ø  underscores
              M=C
```

```
xE83            NC XQ ENLCD         ; Enable LCD for output
                GOSUB OUTHEX        ; Output address
                NC XQ LEFTJ         ; Left justify the screen
                SETF 8
                NC XQ MSG10S        ; Print full message on TRACE printer
                CLRF 5
                GOSUB KEYDBU        ; Wait for a key to be pressed
DSTK1:          NC XQ CLLCDE        ; Same as DSTK2 but for STK1
                NC XQ MESSL
ALPHA::         STK1␣@␣
                GOSUB FIND 9
                RCR 3
                R=0
                LD@R 4
                LD@R 0
                M=C
                NC XQ ENLCD
                GOSUB OUTHEX
                NC XQ LEFTJ
                SETF 8
                NC XQ MSG10S
                CLRF 5
                GOSUB KEYDBU        ; Wait for a key to be pressed
                LDI 0C3             ; If delete then return to display
                ?A#C S&X            ;       of STK2
                JC+04 DBKPT         ; Else progress to display of BkPT
                GOTO DSTK2
DBKPT:          NC XQ CLLCDE        ; Display BkPT (same as DSTK2)
                NC XQ MESSL
ALPHA::         BkPT␣@␣
                GOSUB FIND9
                RCR 13
                R=0
                LD@R 4
                LD@R 0
                M=C
                NC XQ ENLCD
                GOSUB OUTHEX
                NC XQ LEFTJ
                SETF 8
                NC XQ MSG10S
                CLRF 5
                GOSUB KEYDBU        ; Wait for a key to be pressed
                LDI 0C3             ; If delete then return to display
                ?A#C S&X            ;       of STK1
                JC+04 DBEND         ; Else return to main menu
                GOTO DSTK1
xEE1  DBEND:    GOTO MENU
```

DISPLAY BREAKPOINT NUMBER - DBN.

```
DEPENDENCIES :      NONE
ROUTINES USED:      CLLCDE
                    MESSL

INPUT:              BKPT NUMBER IN A[S&X] VALUE ΦΦΦ - ΦΦ9 INCLUSIVE
OUTPUT:             A[S&X] UNCHANGED
                    DISPLAY RIGHT JUSTIFIED SHOWING 'BKPTn ␣@␣' & ENABLED

xEE4  DBN:          NC  XQ  CLLCDE
                    NC  XQ  MESSL
      ALPHA::       BKPT
                    A<>C    S&X
                    A=C     S&X
                    R=2
                    LD@R    Φ
                    LD@R    3
                    WRABCIR
                    NC  XQ  MESSL
      ALPHA::       ␣@␣
xEF7                RTN
```

BREAKPOINT TABLES - BAT, Bn, RBn.

DEPENDENCIES : NONE
ROUTINES USED: NONE
USE : BAT - is used to return the table address
Bø-B9 - used to hold high / low breakpoint address pairs
RBø-RB9 - used to hold the words overwritten by a breakpoint and a jump back to the word after the breakpoint so that when execution continues all operations are successfully completed by re-starting at the appropriate RBn.

```
xEFF  BAT:   RTN                    ; Called using GOSUB to return address of table
xFøø  Bø:    high                   ; High / low byte pair giving address of
             low                    ; breakpoint ø
xFø2  B1:    high                   ; As above for breakpoint 1 etc.
             low
xFø4  B2:    high
             low
xFø6  B3:    :
             :
xFø8  B4:                           ; Note if high byte of a breakpoint
                                    ; is zero BKED assumes the breakpoint
xFøA  B5:                           ; is invalid (or does not exist) as
                                    ; addresses øøxx are in the operating
xFøC  B6:                           ; system.

xFøE  B7:

xF1ø  B8:

xF12  B9:

xF14  RBø:   ovr1                   ; Two words from where the breakpoint
   5         ovr2                   ; instruction is written
   6         *                      ; Jump back to word after breakpoint
   7         NC  GO  (BkPTø)+2      ; instruction to continue execution
xF18  RB1:   ovr1                   ; (For each breakpoint)
   9         ovr2
   A         *
   B         NC  GO  (BkPT1)+2
xF1C  RB2:   ovr1                   ; Note it is assumed that ovr2
   D         ovr2                   ; instructions do not set the carry
   E         *                      ; flag otherwise the NC GO will be
   F         NC  GO  (BkPT2)+2      ; ignored. Also it is assumed that
xF2ø  RB3:                          ; breakpoints will not be placed over
   :         :                      ; 3 word jumps and gosubs, or
   :         :                      ; message outputs etc.
xF38  RB9:   ovr1
   9         ovr2
   A         *
xF3B         NC  GO  (BkPT9)+2
```

DISPLAY BREAKPOINT STATUS - DBS.

DEPENDENCIES : NONE
ROUTINES USED: CLLCDE
PCTOC
MESSL
LEFTJ

INPUT: BREAKPOINT TABLE B0 - B9 CORRECTLY FORMATED
OUTPUT: DISPLAY LEFT JUSTIFIED INDICATING :
'NO BKPTS' No breakpoints set up
'BKPTS ON' Breakpoints set up and enabled
'BKPTS OFF' Breakpoints set up and disabled

```
xF3C  DBS:  NC XQ CLLCDE     ;Clear and enable display
            NC XQ PCTOC      ;Get address of B9 word 0
            R=5              ;   i.e  xF12
            LD@R F           ;   where x is the current purge
            LD@R 1
            LD@R 2
       DBSL: RDROM           ;Get high address word
            ?C#0 S&X         ;If word is not zero then set carry
            JNC+1C DBSN      ;ZERO: Try next breakpoint
            PT=0
            G=C              ;Save high addr in G
            C=C+1 M          ;Get low address
            RDROM
            PT=2
            C=G              ;Combine high and low
            RCR 11
            RDROM            ;Get word at BKPTn
            A<>C S&X         ;Put word in A[S&X]
            LDI 3DD          ;If word is not first part of NC XQ BKPT
            ?A#C S&X         ;  then set the carry flag
            JC+22 DIS        ;  and indicate BKPTS are disabled
            C=C+1 M          ;Else get next word and
            RDROM            ;  shift to give PAGE/SUBPAGE in
            C=C+C S&X        ;  A[2:1]
            C=C+C S&X
            A<>C S&X
            NC XQ PCTOC      ;Get this purge in
            RCR 4            ;  C[2]
            PT=1
            LC B             ;Subpurge B in C[1]
            LC 0
            ?A#C S&X         ;If not the same then set carry
            JC+15 DIS        ;If carry set then must be disabled
            JNC+20 EN        ;Else enabled
       DSBN: PT=Q            ;Point to next word
            PT=4
            PT=P
            PT=3
            C=C-1 PQ                          Q P      Q P
            JC+03 BOFF       ;If carry xF00 -> xEFF then done - no bkpts
            C=C-1 PQ         ;else try next entry
            JNC-25 DBSL
```

```
BOFF:   NC XQ MESSL     ; Indicate 'NO␣BKPTS'
ALPHA:: NO␣BKPTS
        JNC+17 DBSE
DIS :   NC XQ MESSL     ; Indicate 'BKPTS␣OFF'
ALPHA:: BKPTS␣OFF
        JNC+ØB DBSE
EN:     NC XQ MESSL
ALPHA:: BKPTS␣ON        ; Indicate 'BKPTS␣ON'
DBSE:   C<>B  S&X       ; Put last character in B[S&X]
xF8C    NC GO LEFTJ     ; Left justify display and end
```

DISABLE AND FIND BREAKPOINT N - DAFN/FINDBN

```
DEPENDENCIES:   DABKPT
                BAT
ROUTINES USED:  NONE
INPUT:          N REGISTER HOLDS N IN BOTTOM NYBBLE
OUTPUT:         DAFN - DISABLES BREAKPOINTS AND FALLS INTO FINDBN, uses ST
                FINDBN - LEAVES M & N UNCHANGED, A CORRUPT
                         C[ADDR] CONTAINS Bn ADDRESS

xF8E DAFN:  C=N             ; Saves N in ST
            ST=C
            GOSUB DABKPT    ; Disables Breakpoints
            C=ST
            N=C             ; Restores N
xF95 FINDBN: GOSUB  BAT     ; Get Bn table
            C=C+1   M       ;     start address
            A=C     M       ;     into A[ADDR]
            C=N             ; Get breakpoint number from N
            RCR  1          ; Mask off unnecessary digits
            C=Ø     S&X
            RCR  1Ø
            C=C+C   M       ; Double breakpoint number and
            C=C+A   M       ; Add to table address
xFAØ        RTN
```

PRESERVE BUFFER 9, SIGN ON MESSAGE, POWER DOWN PROCEDURE, ROM TRAILER, CHECKSUM.

The following routines are called by the operating system at various times and are used as follows:

PRSV9 : Called at system power up - enables buffer 9 to be located and prohibits operating system from deleting it by ensuring top byte of header = 99.

ONMSG : Prints a user defined message on the screen when the machine is powered up - entered from PRSV9.

PWRDN : Called at system power down - to automatically power down all devices on the HP-IL, this also means that when DEBUG times out all HP-IL devices are powered down.

TRAILER : Used by some ROM verification modules. Indicates the ROM is MP-IB.

CHKSUM : As TRAILER but will normally only verify OK when the ROM has been loaded - the ROM modifies itself as it runs (breakpoint table changes) and so verification of the checksum will fail.

```
xFAE  PRSV9:   N=C              ; Preserve Poll Data
               SETDEC           ; Find Buffer 9
               A=Ø   MS
               A=A-1 MS
               SETHEX
               GOSUB MLOCB
               JNC+Ø2 PCONT
               JNC+Ø6 PEXIT     ;If not found exit
       PCONT:  SETDEC           ; Else make top nybble a 9 as
               C=Ø   MS         ; the operating system will have polled
               C=C-1 MS         ; the ROM to see if owns buffer Ø9.
               SETHEX           ; The ROM does and indicates so by
               WRITE DATA       ; changing the header to 99. If this were
       PEXIT:  C=Ø   S&X        ; not done then the operating system would
               RAM   SLCT       ; delete it after checking all of the other
               C=N              ; ROMS.
               JNC+  ONMSG      ; Restore C and jump to any other routines.
       PWRDN:  C=ST             ; Save entry Poll Data
               N=C
               READ  14(d)
               RCR   6
               ST=C
               SETF  6          ; Set error ignore flag
               C=ST
               RCR   8
               WRITE 14(d)
               C=Ø   M
               LD@R  7          ;Get XROM number of ROM in Page 7
               FETCH
```

```
                        A=C   S&X
                        LDI   #0C       ;Check that the ROM is HP-IL ROM
                        ?A#C  S&X
                        JC+05 DFD6      ;If not then don't call it
                        NC XQ 77C0      ;Power Up All Loop Devices
                        NC XQ 7FCA      ;Power Down All Loop Devices
                        C=N             ;Restore Poll Data
                        ST= C
                        JNC+1A EXIT     ;Exit
              LINK:     JNC-2B PRSV9    ;Link back from ON to PRSV9
              ONMSG:    DSPTOG          ;Turn on message
                        N=C             ;Save Poll Data
                        NC XQ ENLCD     ;Enable display
                        NC XQ MESSL     ;Output the following
              ALPHA::   DON'T␣␣PANIC    ;   12 character user message
                        NC XQ ENCP00    ;Enable chip 00
                        NC XQ MSGDLY    ;Blink display and set message flag
                        C=N             ;Restore Poll Data
                        DSPTOG          ;Turn on display to show message
              EXIT:     NC GO ROMCK10   ;Exit back to operating system Poll Routine
xFF4          PSE:      NOP
   5          RUN:      NOP
   6          WUPNK:    NOP
   7          OFF       JNC-36 PWRDN
   8          I/O:      NOP
   9          ON:       JNC-20 LINK
   A          COLDS:    NOP
   B          TRAILER:  'B'
   C                    '1'
   D                    'P'
   E                    'M'
xFFF          CHKSUM:   checksum word
```