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

<table>
<thead>
<tr>
<th>Register Number</th>
</tr>
</thead>
<tbody>
<tr>
<td>13 12 11 10 9 8 7 6 5 4 3 2 1 0</td>
</tr>
<tr>
<td>U STK2 %P PT P Q G ST</td>
</tr>
<tr>
<td>N M B A C</td>
</tr>
<tr>
<td>I D SIZE U U STK1 BKPT</td>
</tr>
</tbody>
</table>

Where:

- **U** = Unused Nibbles
- **ID** = Buffer Identity = 99 (HEX)
- **SIZE** = Buffer Size = PT (HEX) REGISTERS
- **M** = 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** = Adden 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
\[ \text{MARKS} \quad \text{ROM} \quad \text{HEADER} \]

\[
\begin{align*}
x084 & \quad \text{B} & \quad \text{BE2} \\
x085 & \quad \text{I} & \quad \text{BE1} \\
x086 & \quad \text{J} & \quad \text{BE0} \\
x087 & \quad \text{K} & \quad \text{BE3} \\
x088 & \quad \text{L} & \quad \text{BE8} \\
x089 & \quad \text{M} & \quad \text{BE12} \\
x08C & \quad \text{M} & \quad \text{BE1} \\
x08D & \quad \text{M} & \quad \text{BE0D} \\
\end{align*}
\]

\[ \text{x08C MARKS 18: RTN} \quad ; \text{Perform no operation, just title ROM} \]

**INCREMENTS M FUNCTION - M+1**

\[
\begin{align*}
x096 & \quad \text{1} & \quad \text{BE1} \\
x097 & \quad \text{+} & \quad \text{BE2} \\
x098 & \quad \text{M} & \quad \text{BE8D} \\
\end{align*}
\]

\[ \text{x09E M+1: C = REG 5/M \quad ; \text{Read M register}} \]
\[ \text{x09F C = C + 1 ALL \quad ; \text{Increment it}} \]
\[ \text{x0A0 REG = C 5/M \quad ; \text{Put it back}} \]
\[ \text{x0A1 RTN \quad ; \text{Done}} \]

**DECREMENTS M FUNCTION - M-1**

\[
\begin{align*}
x09D & \quad \text{1} & \quad \text{BE1} \\
x09E & \quad - & \quad \text{BE2D} \\
x09F & \quad \text{M} & \quad \text{BE8D} \\
\end{align*}
\]

\[ \text{x0A0 M-1: C = REG 5/M \quad ; \text{Read M register}} \]
\[ \text{x0A1 C = C - 1 ALL \quad ; \text{Decrement it}} \]
\[ \text{x0A2 REG = C 5/M \quad ; \text{Put it back}} \]
\[ \text{x0A3 RTN \quad ; \text{Done}} \]
SUBROUTINE - DISPLAY REGISTER, POSITION & CONTENTS

DEPENDENCIES: GETNY8
               OUTHEX

ROUTINES USED: CLLCDE
                PCTDC

INPUT:

n REGISTER

WHERE

N = NVYBBLE BANK, EDITED
BBB = BASE ADDRESS OF BUFFER (HEADER)
R = REGISTER BEING EDITED
ST = AVAILABLE FOR STATUS

OUTPUT:

DISPLAY IN FORMAT RIPP = b c, a, d, e, f, g

WHERE

R = REGISTER NAME C, A, B, M or N
PPI = POSITION IN DECIMAL OF EDITOR IN REGISTER
a...g = THREE NYGBBLES EITHER SIDE OF NYYBBLE BEING EDITED (d)

USES:

DISPLAY & REGISTERS & BUFFER 9

x4ΦΦ  DISPLC:  C=Φ  ALL  ; CLEAR OUTPUT STRING

M=C
LDI  Φ3Φ  ; LOAD COUNTER VALUE WITH 3Φ HEX

DL1:
A=C  SΦX  ; STARTING AT -3 FROM NYYBBLE
B=A  SΦX  ; SET 4 NYYBBLES AND ADD THEM
COSUB  GETNY8  ; TO THE OUTPUT STRING
A=C  MS
C=M
A=0C  MS
RCR  13
M=C
C=XΦ  SΦX
PT=1
C=C-1  PT
JNC - DL1

DL2:
A=C  SΦX  ; STARTING AT +1 FROM NYYBBLE
B=A  SΦX  ; SET 2 NYYBBLES (+1,+2,+3) AND
COSUB  GETNY8  ; ADD THEM TO THE OUTPUT STRING
A=C  MS
C=M
A=0C  MS
RCR  13
M=C
A=XΦ  SΦX
LDE  ΦΦ4
A=XΦ  SΦX
C=C+1  SΦX
PA=ΦC  SΦX
JC - DL2
C=M
PT=Φ  ; SO OUTPUT STRING IS FULLY FORMATTED
RCR  13
LDEAR  7

M+C
NC XQ  CLLCDE  ; CLEAR & ENABLE LCD
NC XQ  PCTDC  ; SET RC CURRENT VALUE
JNC + M+C  ; 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
CN ; INTO A[M] AND GET POINTERS
C=E M ; MOVE REGISTER NUMBER INTO C[M]
ACR 13 ; AND ADD IT TO TABLE ADDRESS
C=FA M
FETCH ; SET THE DISPLAY CHARACTER
WRITE 15(e) WRABCIR ; OUTPUT IT TO RIGHT OF LCD
PT=13
LCR A 9
A=C MS ; GET MIDDLE NUMBER AND ADD A
C=CN ; '0' OR A '1' TO THE DISPLAY AS
?IN=A MS ; AS APPROPRIATE
JCP+ST10
LDI 030 ; '0'
WRITE 15(e) WRABCIR
JNC+ DNV
ST10:
LDI 031 ; '1'
WRITE 15(e) WRABCIR
A=C+MS
C=FA MS ; SUBTRACT 9
C=FA-1 MS ; AND 1 TO GIVE CORRECT 2ND DIGIT VALUE
DNV:
LDI 083 ; THEN OUTPUT THE SECOND DIGIT
ACR 13
WRITE 15(e) WRABCIR
LDI 020 ; '2'
WRITE 15(e) WRABCIR ; TWO SPACES
WRITE 15(e) WRABCIR
GOSUB OUTHEX ; OUTPUT 7 CHARACTER OUTPUT STRING
WRABCIR ; AND PUT COMMAS AROUND THE
WRABCIR ; MIDDLE DIGIT TO MAKE IT LOOK GOOD
RDABCIR
RDABCIR
RDABCIR
RDABCIR
C=ELST
SF 6 ; INDICATE ','
SF 7
C=ELST
WRABCIR
RDABCIR
C=ELST
SF 6 ; INDICATE ','
SF 7
C=ELST
WRABCIR
RDABCIR
RTN ; END WITH LCD ENABLED STILL
SUBROUTINE - REGISTER EDITOR - RED.

DEPANDENCIES: MK9 SETHKXA MENU

Routines used: DSPRPC SETNY8

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 EVERY REGISTER DEPENDING ON ROUTINES USED!

x470 RED:

; FIND BUFFER 9
C=0 ALL ; SET UP N REGISTER TO INDICATE
A<=>C SXK ; START OF BUFFER
RCR 11 ; REGISTER BEING EDITED = 1 (C)
C=C+1 XS ; NYBBLE BEING EDITED = 0
N=C

MLOOP:

; DISPLAY REGISTER, POSITION AND CONTENTS
MKEYL:

; SET HEXDECIMAL/SPECIAL KEYPRESS
FS? 2 ; IF SPECIAL KEY PRESSED SET CARRY
JCT SPEC ; AND JUMP TO SPEC
A=0 SXK ; OTHERWISE SET THE KEY VALUE
GSUB SETNY8 ; AND PUT IT IN THE BUFFER
CX>B ALL ; AT THE CURRENT POSITION THEN
RCR 1 ; JUMP OFF TO MONR - MOVE POINTER
CX>B MS ; ONE NYBBLE RIGHT
RCR 13
A=C MS
CX>B ALL
RCR 13
A=A-1 MS
JNC-02 WRITE DATA
JNC+ MONR

SPEC:

C=C-1 SXK ON ; IF NOT 002 PRESSED JUMP TO SPEC2
C=C-1 SXK
JNC+ SPEC2

NC XQ ENCPAB ; OTHERWISE TURN OFF SHIFT FLAG AND
NC XQ OFSHIFT ; ANNUNCIATOR AND GO TO MAIN MENU
GOTO MENU

SPEC2:

C=C-1 SXK SHFT ; IF NOT SHIFT PRESSED JUMP TO SPEC3
C=C-1 SXK
JNC+ SPEC3

NC XQ TOSSHF ; OTHERWISE TTSHE THE SHIFT FLAG AND
NC XQ ANNOUT ; ANNUNCIATOR
JNC-02 MKEYL ; JUMP BACK TO MAIN KEY LOOP (OF RED)

SPEC3:

C=C-1 SXK USEL ; IF NOT USED PRESSED JUMP TO SPEC4
JNC+ SPEC4

NC XQ ENCPAB ; OTHERWISE LOOK AT USER FLAG STATUS
READ 14 (A) ; IF SHIFT IS NOT SET MOVE LEFT A
RCR 2 ; NYBBLE (MONL) ELSE MOVE LEFT A
ST=C ; REGISTER (MONL)
FS? 0
JNC+ MONL
MORL: PT=2
LDGER S
A=>C XS
C=N
?A#C XS ;IF REGISTER NUMBER IS NOT 5
JNC+ NMRI
C=C+1 XS ;THEN INCREMENT REGISTER NUMBER
N=C

NMRI: NC XQ TOGGHF ;ELSE TOGGLE (OFF) SHIFT FLAG AND
NC XQ ANHOUT ;ANNUNCIATOR

JMLLOOP: JNC- MLOOP ;JUMP TO MAIN LOOP & REDRAW DISPLAY
SPEC4: C=C-1 S#X ;IF NOT PGM PRESSED IGNORE KEY
JNC- JMKEYL ;AND JUMP BACK TO MAIN KEY LOOP
NC XQ EXCPG4 ;OTHERWISE INSPECT STATUS OF SHFT
READ 14 (4) ;FLAG AND MOVE ONE NUBBLE RIGHT
RCC 2 ;IF IT IS OFF AND ONE REGISTER
ST= C ;RIGHT IF IT IS ON
?FSET φ ;MOVE ONE REGISTER RIGHT
JNC+ MORR

MORR: C=N ;MOVE ONE REGISTER RIGHT
C=C-1 XS ;DECREMENT REGISTER VALUE
?C#φ XS ;IF NOT ZERO GOTO NMRI
JNC- NMRI ;AND JUMP TO NMRI
N=C

MONK: C=N ;MOVE ONE NUBBLE RIGHT
JMLP4: C=C-1 MS ;DECREMENT NUBBLE VALUE
JNC+ JMLP3 ;IF NO CARRY THEN SAVE VALUE & CONTINUE
C=C-1 XS ;AT JMLP3 OTHERWISE DECREMENT
?C#φ XS ;REGISTER VALUE IF THAT BECOMES ZERO
JNC+ TOFAR ;THEN JUMP TO TOFAR
PT= 13 ;OTHERWISE SET NUBBLE POINTER TO
LDQ E D ;13 AND SAVE STARTS

JMLP3: N=C ;SAVE POINTERS
JNC- JMLLOOP ;REJOIN MAIN KEY LOOP & DRAW LCD

TOFAR: C=C+1 XS ;IF TRIED TO GO TOFAR RIGHT PUT
C#φ MS ;EVERYTHING BACK TO NORMAL
JNC- JMLP3 ;AND REJOIN MAIN LOOP AS IF NOTHING HAPPENED
MONK: PT=13 ;MOVE NUBBLE LEFT
LDQ E
A=C MS
C=N
C=C+1 MS ;INCREMENT NUBBLE VALUE
?A#C MS ;IF NOT 14 THEN
JL- JMLP3 ;JUMP TO JMLP3
C=C+1 XS ;OTHERWISE INCREMENT REGISTER VALUE
A=>C XS
PT=2
LDQ R 6
A=>C XS
?A#C XS ;IF REGISTER VALUE BECOMES 6 THEN
JNC+ TOFAR2 ;JUMP TO TOFAR2
C#φ MS ;OTHERWISE SET NUBBLE VALUE TO
JNC- JMLP3 ;ZERO AND RETURN TO MAIN LOOP

TOFAR2: C=C-1 XS ;IF SAME TOO FAR LEFT DECREMENT
JNC- JMLP4 ;REGISTER & NUBBLE VALUES & GOTO JMLP4
EXTENDED CATALOGUE

For the hex values given the routines must start at $x180$, $x580$, $x960$ or $xD60$ where $x$ is the ROM Page number. Otherwise the values on the lines marked with an $R$ must be changed.

DEPΦ

1 $Φ4$ T
2 $Φ1$ A
3 $Φ3$ C
4 $Φ8$ X
5 $ΦC$ NOP
6 $Φ8$ NC XR CDDC [ΦCF] ; Clear & enable LCD
7 $Φ8$ NC XR MESSL [Φ7EF] ; Output message
8 $Φ8$ "X"
9 $Φ8$ "(Φ-F)"$
10 $Φ8$ A
11 $Φ8$ T
12 $Φ2$ T
13 $Φ2$ C
14 $Φ2$ D
15 $Φ2$ F
16 $Φ2$ E
17 $Φ2$ F
18 $Φ3$ C
19 $Φ3$ GCUB
20 $Φ4$ DEF GETHEX ; Get Hexadecimal Key Value
21 $Φ4$ STC ; If a 'special key' is pressed
22 $Φ6$ CRTN ; then exit from routine
23 $Φ8$ ST=S ; Store HEX value in ST bits 0-7
24 $Φ8$ A=C X (S&X) ; Save value in ASCII for testing
25 $Φ8$ LDI ; ordinary catalogue
26 $Φ8$ C $Φ8$ R84 ; routine
27 $Φ6$ ABC=$Φ$ ; Otherwise CAT 2 from PAGE specified
28 $Φ6$ PT=Q ; Count number of functions ; $Φ8$ [X]
29 $Φ6$ PT=6 ; Test value of 5 in ADDR field
30 $Φ6$ A=XR ALL ; of the A register
31 $Φ6$ C=ST ; Get PAGE number from LSD of ST
32 $Φ6$ RCR B ; Add on $Φ8$ to point to
33 $Φ6$ LC $Φ$ ; number of functions in page
34 $Φ6$ LC $Φ$ ; i.e., PΦΦ
35 $Φ6$ LC 1

33φ
RDROM

2E6
?C+Φ X (S)x

38φ
*

15C
PT = 6

362
?A B C PT (ER)

43
JNC +Φ8 Part 1

4C
C = C - 1 PT

66
AC = (S)x

606
C = C + A X (S)x

C3
JNC - Φ8 Part 1

238
READ 8 (P)

4C
PT = 5

15C
PT = 6

2DC
PT = 13

0C
NC 2

6E
C = (S)x ALL

7C
RCR 4

6E
C = (S)x ALL

6E
A = B ALL

6E
C = C + A ALL

2B1
*

60E
NC 40 ΦE BC

GET HEXADECIMAL KEY PRESS

B
34C
PT = φ

264
CF = 2

37φ
κ

0C
NC XQ GOSUB

51
DEF INIT

2C
DEF KEYDB

6C
NC XQ GOSUB

35B
JNC - ΦC GETHEX

366
?C+Φ X (S)x

03B
JNC +Φ7 FOUND

OEG
B = Φ X (S)x

222
C = C + 1 PT (ER)

13
JNC +Φ2 GHXL2

08
SF 2

6C
B = C X (S)x

3AB
JNC -Φ8 GHXL2

6C
C = B X (S)x

3E0
RTN

; look to see if this page is empty
; if it is then
; DISPLAY "NONEXISTENT" & END
; Set PTR for testing decrementing of page no.
; If we have added up all ROMs down to page 5 then GOTO P 1 to read CAT for
; Down to next ROM page
; Get number of functions in C[S]
; Put counter TOTAL in A
; Add the number of functions in this page
; Store new total in A
; Restore total in B and restore A[S]
; Do again for next page
; Get ALPHA top register / CAT working register
; Clear C[13:6]
; Set CAT numbers to ROMS 2
; Get counter TOTAL in C[S]
; Ω[12:11]φ
; Ω[12:11]φ
; Ω[11:10]φ
; Mask to give C = 2e.c φ φ φ
; Jump into CAT where C is just about to be put into P register
; Debounce keyboard & head A Key
; Set up key table address
; End of table reached. Therefore
; Invalid key so try again - not found in table
; Increase key number
; If no carry (HEX mode) don't set flag 2
; Set to indicate special key
; Compare with next entry in table
; Return the key number (from table)
; in both C[EX1 and EX2]
Debounce keyboard & read key.

D561 KEYDB: 3CC ?KEY ; If no key is down
    2 φ23 JNC +φ4 SETKEY ; Then wait for one in SETKEY
    3 KEYSUP: 3C8 CLRKEY ; Else wait for the key
    4 φ3C ?KEY ; that is pressed down
    5 3F7 JNC -φ2 KEYSUP ; to be released
    6 SETKEY: 3CC ?KEY ; Now no keys are down get the
    7 3FB JNC -φ1 SETKEY ; next one pressed
    8 22F C=KEY ; Put the value in C[4:3]
    9 38C RCR 3 ; C[1:φ]
    A φ56 C=φ XS (XS) ; C[2:φ] ;
    B φ16 A=C X (SBX) ; A[2:φ]; i.e. A[X]

Hexadecimal Keyboard Look Up Table.

D56C INIT: 3E0 RTN ; End of KEYDB & Start of keyTable

D ; Newel:
E φ37
F φ36
Fφ φ76
1 φ66
2 φ25
3 φ75
4 φ05
5 φ34
6 φφ φ4
7 φ94
8 φ10
9 φ3φ
A φ70
B φ8φ
C φCφ
D φ11
E φC3
F φ18

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
    1 φC6 ; USER
    2 φC5 ; PRGM
    3 φ13 ; ENTER
    4 D584 φφ0 NOP ; End of Table Marker
    5
    6
SUBROUTINE - GET HEX KEY WITH TIMEOUT - TKEYH.

DEPENDING ON:
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 PT TO φ
C [KEY], [S&X], [MX] E.E. ALL
B [S&X]
A [S&X]

TKEYH:
CLRKEY
PT=φ
LID
HEX 3FF
CLR
C=C-1 S&X
JCB+4 L2
?KEY
JNC-01 L1
JNC+04 PARSEH
L2:
POP
C=CH+1 M
GOTO ADR
PARSEH:
C=KEY
LID C S&X
A=C S&X
B=Φ S&X
GOVB D56C (WIT)
GXL1:
C=CH+1 M
RDMROM
?C+Φ S&X
JNC-0E L2
?A=Φ S&X
JNC+Φ6 FOUND
B=C S&X
C=CH+1 PT
JCC-13 L2
B=CC S&X
JNC-0A GXL1
FOUND:
C=B S&X
CLRKEY
?KEY
JCC-Φ2
RTN

; Clear the keydown flag
; C S&X holds time-out constant
; Decrement time-out constant
; If carry then timed-out, goto L2
; Check for a key down
; If no key down carry on writing
; otherwise try to parse key
; Time-out (Invalid key exit
; Skip line after all to TKEYH
; Get key code
; Put it in C[S&X]
; and A [S&X]
; Clear Hex value counter
; Get slot of key table in C[ADDR]
; Point to next word in table
; Get the word
; If zero then we have run of the end
; so Invalid key exit
; If keydown matches table then
; we have found the right one
; If not then increment Hex value counter
; Otherwise try next table entry
; Duplicate Hex value in C[S&X]
; Wait for key to come up

; END
SUBROUTINE - MAIN STATUS DISPLAY - MAIND.

DEPENDENCIES:
OUTHEX @
MESSL @
CLCDE @

ROUTINES USED:

INPUT:
N REGISTER HOLDS STATUS INFORMATION (BUFQ FORMAT)
FLAG  7 INDICATES DISPLAY PT OR _
FLAG  9 " P OR \
FLAG 11 " Q OR \\n
OUTPUT:
DISPLAY SHOWS : PT = r  P = p  Q = q
N REGISTER UNCHANGED

x58# MAIND:
NC XRQ CLCDE ; Clear & enable LCD
NC XRQ MESSL ; Output message PT=... right of LCD
  @10 P
  @14 T
  23D =

?FS 7 ; If flag 7 is set, display an underscore
JC+@8 US1 ; other than P or Q
C=N ; Get status register
RCR 4 ; Put the P/Q nybble in C[XRQ]
?C+@0 XR ; Set carry if non zero i.e. if 'Q'
JNC+@4 ACP ; If not 'Q', display a P
LDI ; otherwise display Q
HEX @11 'Q'
JNC+@6 CONT1 ; Go & write digit to LCD

ACP:
LDI ; Display P
HEX @10 'P'
JNC+@3 CONT1

US1:
LDI ; Display 'r' underscore
HEX @1F 'r'
JNC+@5 CONT1

CONT1:
WRABCIR ; Output digit
NC XRQ MESSL ; Output message UP=... right of LCD
  @20 W
  @10 P
  23D =

?FS 9 ; If flag 9 is set, display an underscore
JC+@B US2 ; otherwise get the value of P
C=N ; from the status register and indicate
RCR 4 ; to 'other' native that there is
LC 1 ; 1 digit
LC 0 ; 0 underscores
M=C
NC XRQ GOSUB
DEF OUTHEX ; Output P & continue below

US2:
LDI ; Output underscore 'r'
HEX @1C 'r'
WRABCIR

CONT2:
NC XRQ MESSL ; Output message UQ=... to RHS of LCD
  @20 U
  @11 Q
  23D =

?FS 11 ; If flag 11 set, display underscore
JC+@B US3

PT=φ ; otherwise set value of φ
C=N ; out of status information and
RCR 3 ; indicate to OUTHEX
LC 1 ; 1 digit
LC φ ; 0 underscores
M=C
NC XR GOSUB
DEF OUTHEX ; Display value of φ
RTN ; End

; Display final '-'

RTN ; and End

SUBROUTINE - ALTERNATIVE STATUS DISPLAY - ALT D.

DEPENDENCIES: OUTHEX @ MESSL @ CLCDE @ GOSUB
ROUTINES USED
INPUT:
N REGISTER holds status information (comma format)
FLAG 6 indicates display M, ST, OR ' - '
FLAG 5 " " LSN ST OR ' - '
FLAG 4 " " MSN G OR ' - '
FLAG 3 " " LSN G OR ' - '

OUTPUT:
DISPLAY shows: S T = . H D G = . . L
N REGISTER unchanged
FLAG 10 (ALT DISPLAY) set

ALSEF ALT D:
NC XR CLCDE ; Clear & enable LCD
NC XR MESSL ; Output ST= to RHS of LCD
φ13 S
φ14 T
23D =
?FS 6 ; If flag 6 is set, display underscore
JO?φA US4 ; Other than the most significant nibble
C=N
PT=φ ; of ST
LC 1 ; 1 digit
LC φ ; 0 underscores
M=C
NC XR GOSUB
DEF OUTHEX ; Displaying MSN of ST
JNC+φ4 CONT3

US4:
LDI
HEX φ1F ; ' - '
WRABC1R

CONT3:
?FS 5 ; If flag 5 is set, display underscore
JO?φB US5 ; Other than the least significant nibble
C=N
RCR 13 ; of ST
PT= φ
LC 1 ; 1 digit
LC φ ; 0 underscores
M=C
NC XR GOSUB
DEF OUTHEX ; Display LSN of ST
JNC+φ4 CONT4
US5:
LDI
HEX 01F 1.
WR ABC1R

CONT4:
NC XQ MESSL
02B =
087 G
23D =
FFS 4
JC+ΦB US6
C = N
RCR 2
PT = Φ
LC 1
LC Φ
M = C
NC XQ 60SUB
DEF OUTH EX
JNC+Φ4 CONT5

US6:
LDI 01F 1.
WR ABC1R
FFS 3
JC+ΦB US7
C = N
RCR 1
PT = Φ
LC 1
LC Φ
M = C
NC XQ 60SUB
DEF OUTH EX
JNC+Φ4 CONT6

US7:
LDI
HEX 01F 1.
WR ABC1R

CONT6:
SF 10
C = N
PT = 7
?C#Φ 0R
JC+Φ6 ALD
NC XQ MESSL
02B =
2Φ8 H
RTN

ALD:
NC XQ MESSL
02B =
2Φ4 D
RTN

x644
SUBROUTINE - GET KEY WITH TIMEOUT - TKEY.

DEPENDENCIES: NONE
Routines used: NONE
Input: NONE
Output: RETURNS DIRECTLY WITH KEYPAD IN A AND C \([S\&X]\)
        SKIPS A LINE & RETURNS IF TIMED-OUT
Uses: ACTIVE PT \(\neq\) SETS IT TO \(\phi\)
        C \([K\&Y], [M\&], C[S\&X]\) i.e. ALL \(A[S\&X]\)

x645
TKEY:
CLRKEY\(\text{; Clear the keydown flag}\)
R\(\neq\)\(\phi\)
LDI
HEX 3FF \(\text{; C[S\&X] hold the time-out constant}\)
L4:
CCE-1 S\&X \(\text{; Decrement the-out constant}\)
JNC+\(\phi\)4 SKPNXT \(\text{; If any then time-out, goto SKPNXT}\)
?KEY \(\text{; Check for a key down}\)
JNC-\(\phi\)3 L4 \(\text{; If no key down carry on waiting}\)
JNC+\(\phi\)4 G0TKEY \(\text{; Otherwise, got a key so exit}\)
SKPNXT:
P0P \(\text{; Time-out return, skip this skiv call TKEY}\)
C=C+1 M
G0T0 ADDR
G0TKEY:
C=KEY \(\text{; Key pressed or get its value}\)
RCR 3 \(\text{; cd set up keycode in c[S\&X]}\)
C=\(\phi\) X5 \(\text{; cd set up keycode in c[S\&X]}\)
A=C S\&X
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 keycodes \((M5N=\neq\phi)\)

x656
KEYDBU:
?KEY \(\text{; If no key down, skip this list}\)
JNC+\(\phi\)4 \(\text{; Otherwise, wait for key to come up}\)
CLRKEY?
?KEY
JCE-\(\phi\)2
?KEY
JNC-\(\phi\)
C=KEY
RCR 3 \(\text{; Key down - get its value in}\)
C=\(\phi\) X5 \(\text{; cd set up keycode in c[S\&X]}\)
A=C S\&X \(\text{; cd set up keycode in A[S\&X]}\)
CLRKEY?
?KEY
JCE-\(\phi\)2
RTN \(\text{; When it is up then exit}\)

PATCH TO ALLOW STED S 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 \(\text{; GET TOP REGISTER (STATUS)}\)
PT=\(\phi\) \(\text{; CLEAR [P\&] ONLY}\)
C=\(\phi\) \(\text{WPT}\)
JNC+39 JLOOP \(\text{; RETURN & RESTORE STATUS}\)
SUBROUTINE - ASK USER IF HE IS SURE.

DEPENDENCIES: KEYDBU
Routines Used: MESSL
CLCDE

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 [call] c [sx] returned containing #16
A [sx] returns with key code of key pressed
LCD enabled & contains message on exit

X665 SURE: NC XQ CLCDE ; clear & enable LCD
NC XA MESSL ; output prompt

Alpha: SURE(_Y/N)?
GOSUB KEYDBU ; get a new key & wait for it to come up
LDI #16 ; compare it with y key
PHA 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 RB&..RB4 and B&..B4
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 B&..B4 and the code overwritten in the table RB&..RB4 (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 ; set rom page number in C[67]
PT=5
LC F
LC 1
LC 2

DO: RDSM ; set high address byte (from low word)
?C=PH S&X ; if zero then breakpoint is not active
JNCNHF DNEXT ; so check next
N=C ; put address and word in N
PT=PH
G=C ; put word in G
C=C+1 M ; increment address to get (high word)
RDSM ; low address byte
PT=2 ;POSITION POINTER TO PUT BACK HIGH ADDR
C=G ;NOW C[3:0] HOLDS BREAKPOINTS ADDR
ROR 11 ;NOW IN C[ADD]
B<>C ALL ;NOW IN G[ADD]
C=N ;AND IS READY TO PUT BACK OUR1 & OUR2
A=C ALL ;A[6:3] HOLDS LOOP VALUE DUPLICATE
ROR 3 ;LOOP IN C[3:0]
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 PT ;ADD 10 HEX
ROR 11 ;AND PUT INTO C[ADD]. THIS IS RBa
RDM ;GET WORD OUR1 FROM RBa
B<>C M ;GET BKPT ADDRESS
WRM ;SAVE WORD OUR1 TO BKPTa
C=C+1 M ;INCREMENT BKPT ADDRESS
B<>C M
C=C+1 M ;INCREMENT RBa ADDRESS
RDM ;GET WORD OUR2 FROM RBa+1
B<>C M
WRM ;WRITE WORD OUR2 TO BKPTa+1
NEXT:
PT=Q
PT=4
PT=0
PT=3
C=C-1 PQ ;DECREMENT LOW BYTE OF LOOP
C RNT ;IF THERE IS A CARRY THEN ALL DONE: END
C=C-1 PQ ;ELSE DECREMENT AGAIN TO GIVE NEXT
X6AD JNC-28 DLOOP ;Ba 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
B0, B9 ARE ENABLED i.e. "NC XQ BKPT" OVERWRITES
THE CODE THAT WAS THERE ORIGINALLY.
USES: A, C, M, N, G, P, Q FLAGS 0 & 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: RDM ;GET high ADDRESS byte
PC<>0 S6X ;IF ZERO THEN BREAKPOINT IS NOT
JNC+18 Next ;ACTIVE SO CHECK NEXT
N=C
PT=0
G=C
C=C+1 IM
RDROM
; MSBYTE IN G
LSBYTE IN C[2:0]
Pt=2
C=G
; C[3:0] HOLDS BKPT ADDRESS
Rcr 11
; A[ADDR] HOLDS BKPT ADDRESS
A<>C ALL
; A[ADDR] HOLDS BKPT ADDRESS FOR WRC1
NC XQ PTRDC
; PUT PAGE NUMBER INTO C[6]
Pt=5
LC B
LC F
LC 7
; ADDRESS OF 'BKPT' IS xBF7
Rcr 3
; PUT INTO C[3:0]
M=C
; M[3:0]
CF φ
; XQ
CF 1
; NC
GO SUB WRC1
; WRITES NC XQ 'BKPT' @ BKPT
C=N
; C[6:3] HOLDS LOOP VALUE
ENEXT:
PT=0
PT=4
PT=5
PT=3
C=C-1 P-Q
; DECREMENT LOW BYTE OF LOOP
C RIN
; THERE IS A CAUSE THEN ALL DONE: END
C=C-1 P-Q
; ELSE DECREMENT AGAIN TO SUE NEXT
x6D5
JNC -21 ELOOP
; An ADDRESS AND REPEAT PROCESS

CONTINUE AFTER BREAKPOINT — CONTBK.

DEPENDENCIES:
FINDY WRCL KEYDBU ENTRY
MENU OUTHEX CR9

ROUTINES USED:
PCRDC MESSL MSG105

INPUT:
BUFFER A 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 RBA' INTO ENTRY
AND JUMPS TO CR9 (B9-CPU) AND ENTERS CODE THROUGH
BREAKPOINT. CONFIRMATION OF ADDRESS (B9-R6) IS
REQUIRED BY USER BEFORE EXECUTION COMMENCES.

USES:
ALL REGISTERS, BUFFER A, BREAKPOINT TABLES

x6D6 CONTBK:
GO SUB FINDY
A=C ALL
NC XQ PCRD
Pt=5
LC B
LC 1
LC 2
; C[ADDR] = xF12 = 'B9'
CLOOP: RDROM
C R
JNC+X C NEXT ; TRY THE NEXT ONE
N C ; N HOLDS LOOP IN [6:3]
P C=H
RDROM
P C=2
C=G
P C=3
?A=C WPT ; IF NOT SAME SET CARRY
JNC+D CFOUND ; IF SAME THEN FOUND RIGHT BREAKPOINT
C=N

NEXT: P C=Q
P C=S
P C=P
C=C-1 PQ ; DECREMENT LOOP LOWER BYTE
JC+D CEND ; IF CARRY THEN JUMP TO CEND
C=C-1 PQ ; ELSE MOVE LOOP TO NEXT G
JNC-15 CLOOP ; AND SEE IF IT IS THE RIGHT ONE
GOOD MENU ; NO CORRESPONDING BREAKPOINT FOUND
CFOUND: C<>N ; PUT ENTRY ADDRESS IN N
RCR 3 ; PUT LOOP IN C[3:0]
P C=CH 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]
P C=5
LC 18
LC F
LC 3 ; XBF3 IS ADDRESS OF 'ENTRY'
A=FC ALL
SF C
CF 1 ; NC
GOSUB WRC1 ; WRITE 'NC GO RBn' AT 'ENTRY'
C=N ; GET BACK ADDRESS FOR USER
RCR 13 ; TO CONFIRM
P C=0
LC 4 ; 4 DIGITS
LC 0 ; 0 UNDERSCORES
M=C
NC XQ CLLCDE
NC XQ MESSL

ALPHA::
CONT w@+w
GOSUB OUTHEX ; OUTPUT ADDRESS
NC XQ LEFT
GOSUB LSTRU ; GET A NEW KEY & WAIT FOR IT
LDI 087 ; TO COME UP, COMPARE WITH R/S
?A=L C S&X ; IF NOT R/S SET THE CARRY FLAG
JNC+04 CSETUP ; R/S PRESSED SO CONTINUE SET UP
COTD MENU ; ELSE RETURN TO MENU
SF &8
NC XQ MSF&5
CF 5
GOSUB FIND9
NC XQ LTOC
PT=S
LC B
LC F
LC 5
PUSH ; FOR SECURITY PUT 'RTH' ADDRESS ON STACK
LDI 006 ; POINT TO TOP REGISTER OF BUFFER
C=C+1A S&X
RAM SLCT
READ DATA
RCR 5 ; RETRIEVE STL2 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 STL1 AND
PUSH ; PUT IT ON THE STACK
GOTO CR9 ; ENTERS B9>CPU HERE TO
; PRESERVE STACK AND AS BUFFER IS ALREADY FOUND AND SELECTED

DISPLAY BREAKPOINT INFORMATION (ADDRESS) - DBI

DEPENDENCIES: DBN BAT OUTHEX
ROUTINES USED: NONE
INPUT: BREAKPOINT NUMBER IN A[S&X]
OUTPUT: ADDRESS OF BREAKPOINT APPENDED TO DISPLAY
; i.e. 'BKPTA viet
USES: ALL REGISTERS CORRUPTED EXCEPT N

GOSUB DBN ; DISPLAY BREAKPOINT NUMBER
GOSUB BAT ; GET START ADDRESS OF BREAKPOINT
C=C+1 M ; TABLE, i.e. DP#0 (IF IN FLE D.B.)
A<<C ALL ; PUT ADDRESS IN A ADDRESS
RCR 1 ; AND SET BREAKPOINT NUMBER
C=0 S&X
RCR 10
C=C+C M ; MULTIPLY BY TWO TO GIVE OFFSET
C=A+T C M ; COMBINE OFFSET & BKPT TABLE ADDRESS
RDROM ; GET HE WORD AT BA
RC#0 S&X ; IF ZER0 THEN NO BREAKPOINT
JNC+04C NOBK ; SO JUMP & DISPLAY JUST 4 UNDERSCORES
PT=\phi
G=C
C=C+1 \text{ M}
\text{ ; SET LOW ADDRESS IN C[J:J]}
PT=2
C=G
\text{ ; COMBINE ADDRESSES AND ROTATE}
RCR 13
\text{ ; READY FOR OUTHEX FORMATT}

LC 4
\text{ ; 4 DIGITS}
LC \phi
\text{ ; \phi UNDERSCORES}
JNCI+\phi 4 DBIE

NOBK:
PT=\phi
\text{ ; IF NO BREAKPOINT THEN}
LC \phi
\text{ ; \phi UNDERSCORES}
LC 4
\text{ ; 4 UNDERSCORES}

DBIE:
M+C
SOTO OUTHEX

\text{ x76D }

LOW LEVEL BREAKPOINT EDITOR - EDGN

DEPENDENCIES:
GETHXA \text{ B KD } DAFN OUTHEX ADDDI G WRC1
DBI ENBKPNT DBN DEDIG FINDGN

ROUTINES USED:
MSGI\phi5 \text{ LEFTJ}

INPUT:
HIGH LEVEL MSGW COMES IN VIA EDGN WITH
BREAKPOINT TO EDIT IN B[L8\phiX] AND A[L8\phiX]

OUTPUT:
EXITs THROUGH BKED (HIGH LEVEL EDITOR)
ALLOWS ENTRY OF NEW ADDRESS FOR BKPTA, PRINTING
OF BREAKPOINT ADDRESS & NORMAL LOCAL EDITING
FUNCTIONS (ASSORT, DELETE, FINISH)

USES:
ALL REGISTERS, DISPLAY, BREAKPOINT TABLE, ROM UNDER TEST

\text{ x76E EDGN:}
A=C S\phiX
\text{ ; GET BKPT#}
N=C \text{ ; SAVE IN N}
EDBL:
C=N \text{ ; RETRIEVE BKPT#}
A=C S\phiX
GOSUB DBI \text{ ; DISPLAY BKPT INFORMATION}
EDBK:
GOSUB GETHXA \text{ ; GET HEXADECIMAL KEYPRESS}
?FS 2 \text{ ; IF SPECIAL KEY SET CARRY}
JNCI INPUT \text{ ; IF \phi..F GO TO LOW LEVEL INPUT}
LOE \phi 46 \text{ ; ENTER}
?AHC S\phiX
JCR NXK1 \text{ ; IF NOT ENTER TRY NEXT KEY}
C=N \text{ ; ELSE SAVE N}
M=C
SF 8
NC xQA MSGI\phi5 \text{ ; PRINT DISPLAY}
CF 6
C=M
N=C \text{ ; RETRIEVE N}
JEDBK:
JNC- EDBK
NXK1:
C=\phi S\phiX
?AHC S\phiX
JCR NXK2 \text{ ; IF NOT DELETE TRY NEXT KEY}
EDEL:  GOSUB DAEN ; DISABLE & FIND BKPT
C=0 $X
; MAKE IT UNREADABLE BY MAKING
; $X = \phi_0\phi_0
VROM
JEDBL:
JNC- EDBL
NXK2:  C=C+1 $X
?#C $X
JCT+ NXK3
; IF NOT ON TRY NEXT KEY
EON:
SETHEX
GOSUB GNKPT
; REQUIRED FOR ENTRY FROM ERS
GOTO BKED
; ENABLE BREAKPOINTS AND
NXK3:
C=C+1 $X
?#C $X
; RETURN TO HIGH LEVEL MENU
JTC- JEDBK
; IF NOT RIS IGNORE KEY & SET ANOTHER
ERS:
PT=0
C=N
SETDEC
C=C+1 PT
; IF A CARRY IS CAUSED BY 9 \rightarrow 10 THEN
JTC- EON
; RETURN TO TOP LEVEL OF BKED
SUTHESX
; OTHERWISE GET NEXT BKPT INFO
N=C
JNC- JEDBL
; SET PAGE INTO C[13]
INPUT:  RCR 13
PT=0
; ADD ON DIGITS FOR OUTHESX
LC 1
; TO INDICATE 1 DIGIT
LC 3
; AND 3 REMAINING UNREASCRING
M=N
; SAVE IN M READY
RCR 9
; PUT PAGE # INTO C[ADDR]
PT=1
; GET A WORD TO TEST FOR RAM OR ROM
RDRM
; GET CONTENTS
A=C WPT
; STORE FOR COMPARISON LATER
C=C-1 WPT
; TAKE 1 COMPLEMENT
WROM
; SAVE WORD
RDRM
; READ WORD IF ROM WILL BE DIFFERENT
C=C-1 WPT
; RE-COMPLEMENT
WROM
; WRITE ORIGINAL BACK (FOR RAM)
=?#C WPT
; COMPARE ORIGINAL & RE-READ VALUE
JTC- JEDBL
; SET CARRY IF ROM & SO IGNORE KEY
LLDSP:
C=N
; SET BKPTH
A=C $X
; INTO A[$X] FOR DBN
GOSUB DGN
; DISPLAY BKPT,0
GOSUB OUTHESX
; OUTPUT PART FORMED ADDRESS
NC XG LEFT
; AND LEFT JUSTIFY DISPLAY
LLLOOP:
GOSUB SETHAX
; SETHEXDECIMAL KEYPRESS
FS? $2
; SET CC0M IF SPECIAL KEY
JNC+ LLIN
; IF 0..9 THEN ORDINARY INPUT
C=0 $X
?#C $X
JTC+ NXK4
; IF NOT \rightarrow TRY NEXT KEY
LLDEL:
C=M
PT=0
C=C-1 PT
?C#PH PT
JTC+ LLDEL
GOTO EDBL
; IF DELETE 1ST DIGIT GO UP A LEVEL
LDEL: GOSUB DELDIG ; ELSE DELETE DIGIT 
JNC- LLDSP ; RE-DISPLAY AT THIS LEVEL
JNZ- LLDSP ; ORDINARY INPUT
JNC- LLOOP ; CHECK THERE IS ROOM LEFT & IGNORE IF NOT
JNC- ADDDIG ; OTHERWISE ADD DIGIT AND
JNC- LLDSP ; RE-DISPLAY AT THIS LEVEL
NXT4: C=C+1 ; ON ; IF NOT ON THEN TRY ANOTHER KEY
?A#C S&X ; ON ; IF ON THEN 40 & SKIP A BIT
JNC+ NXT
C=C+1 S&X
JNC- LLOOP ; IF NOT R is THEN GET ANOTHER KEY
CF 3 ; INDICATE SHOULD RETURN TO EDLB
C=M S&X ; SET NUMBER OF DIGITS INPUT
A=C S&X ; IN A[S&X]
LDI D&4
PT=0
A=C S&X
JNC+ FOURIN ; IF 4 DIGITS SKIP NEXT BIT
JNC- LLOOP ; THIS HAS NO EFFECT UNLESS 4 DIGITS INPUT
NXT: SF 3 ; INDICATE SHOULD RETURN TO EON
C=M S&X ; GET NUMBER OF DIGITS INPUT
A=C S&X ; INTO A[S&X]
LDI D&4
PT=0
?A#C PT ; IF NOT 4 THEN SET CARRY
JNC+ FOURIN ; IF 4 DIGITS THEN SKIP NEXT BIT
GOTO EON ; ELSE ON ABORT ADDRESS ENTRY &
; RETURN TO TOP LEVEL OR EXIT
FOURIN: GOSUB DAFIN ; DISABLE KEPT AND SET ADDR OF KEPT
A=C M ; STORE IN A[ADDR]
C=M ; GET HIGH ADDRESS BYTE
RCR 3 ; INTO C[1:0]
C=0 XS ; INTO C[2:0]
A>0 C M ; GET ADDRESS OF KEPT
A=C M ; STORE FOR LATER USE
WROM ; WRITE HIGH ADDRESS BYTE INTO Bn
CM ; SET LOW ADDRESS BYTE
RCR 1 ; INTO C[1:0]
C=0 XS ; INTO C[2:0]
A>0 C M ; SET ADDRESS OF KEPT (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 uncompt and contain addd of BKPT
and BKPT# respectively

Now RBn to RBn+3 must be updated
It is not necserary to evolve the actual breakpoint as that
will be done automatically when we return to the top level remn
of BKED
GO SUB FINDBN

RCL 3

PT=1

C=C+1

WPT

C=C+1

WPT

C=C+1

WPT

C=C+1

WPT

C=C+1

PT

C[C3:0] = RBn

RCL 11

C [ADDR] = RBn

B=C

M

B [ADDR] = RBn

RCL 12

C [ADDR] = BKPTn

RDROM

GET OVR1

B=C

M

SWAP IN ADDRESS OF RBn

WRROM

WRITE OVR1 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 OVR2

B=C

M

SWAP IN ADDRESS OF RBn+1

WRROM

WRITE OVR2 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 WRCL

C=B

M

C [ADDR] = BKPTn+1

RCL 3

C[C3:0] = BKPTn+2

M=C

MC[C3:0] = BKPTn+2 READY FOR WRCL

SF φ

GO

CF 1

NC

GO SUB WRCL

WRITE 'NC GO BKPTn+2' TO RBn2 & 3

IFS 3

= R15 Pressed THEN JUST SET UP THIS

JCT+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

Routines Used: ENCPX

IAUNA - PRINTER ROM ADDRESS 6DB2
PBRTX - PRINTER ROM ADDRESS 6FF2
OUTPUT - PRINTER ROM ADDRESS 63DD


OUTPUT: IF PRINTER IN TRACE MODE GIVES:

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

X82F PRCPU:

LDE 013 ENTER ; Entry is from MENU so see if
PA #C S&X ; enter 'hay' in dump before printing
JNC+ PRCPU1 ; if it is enter PRCPU1
JMENU: GOTO MENU ; if it is not jump back to menu
PRCPU1: NC XQ ENCPX ; Enable chip 00
NC XQ IAUNA ; Try to initialize printer
JNC- JMENU ; No print so return immediately
LDE 043 'C' ; Print 'CPU':
NC XQ PBRTX
LDE 050 'P'
NC XQ PBRTX
LDE 055 'U'
NC XQ PBRTX
LDE 03A 'I'
NC XQ PBRTX
NC XQ OUTPUT ; Send ready
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
LDE 043 'C' ; Label output with 'C'
M=C
GOSUB PREG ; Print register & increment pointer to next
LDE 041 'A'
M=C
GOSUB PREG ; Register 'A'
LDE 042 'B'
M=C
GOSUB PREG ; Register 'B'
LDE 04D 'M'
M=C
GOSUB PREG ; Register 'M'
LDE 04E 'N'
M=C
GOSUB PREG ; Register 'N'
X82F JMENU2: GOTO MENU ; Return to main menu
x873  PRREG:  NC  XQ  EICP@0  ; Enable chip @0
NC  XQ  INUMA  ; Check printing is OK
JNC+  JMNUE  ; If not return to menu
C=M  ; Else get character label from M
NC  XQ  PBXTX  ; Send to printer
LDI  0@3D  '='  ; Send '=' to printer
NC  XQ  PBXTX  ; Enable chip @0
NC  XQ  EICP@0  ; Enable chip @0
C=B  SBLX  ; Get register address
RAM  SLCT  ; Select register
C=C+1  SBLX  ; Increment pointer for next time
C<>B  SBLX  ; and put it back in B[EsdxI]
READ  DATA  ; Get register
SLCT  Q  ; Pointer Q is used as a counter for
R=0  ; the number of digits output
PR  13  ; Put digit to be printed in C[0]
M=C  ; Save register for next time
SLCT  P  ;
R=0  ;
A=C  @R  ;
LDI  0@0A  ;
JNC+  GT9  ; If digit in A..F jump to GT9
LDD  0@3D  ; Prepare mask for number 0..9
JNC+  CPR2  ;
GT9:  C=C-1  SBLX  ; If A..F then subtract 9
A=A-C  @R  ;
LDD  0@4D  ; And prepare mask for letter A..F
CPR2:  A<>C  @R  ; Output character
NC  XQ  PBXTX  ;
C=M  ;
SLCT  Q  ; Printer Q in the digit counter
R=R+1  ; When it returns to 0 all the digits
R=0  ; have been output
JNC-  CPR1  ; Until then go around the loop again
x8 AP  ;
NC  GO  OUTPUT  ; Finally send end of line and return
DEPENDENCIES:

Routines used:
GOSUB
MSG, IF5
GOTO

INPUT:
BUFFER9 CONTAINING BUFFER ID, REGISTERS C,A,B,M,N AND STATUS
(IN BU99 FORMAT)

OUTPUT:
UPDATED BUFFER9 WITH NEW STATUS INFORMATION

x:03 STED:
ST = $0
; Clear all flags $0-$7 (used for prompting)
CF 11
; Clear A prompt
CF 9
; Clear P prompt
GOSUB MK9
; Find buffer 9, Add in A
LDI $0006
; Add 6 to point to top register containing
C = C + A $0006
; status information
RAM SLCT
; Get this info into C
READ DATA
N = C
; and keep it in N throughout edit session.
LOOP1:
CF $0F
; Indicates display main set of status
?FS $0F
; IF flag 10 is set, then display alternative
JCE $05 DALT
; set below
DMAIN:
GOSUB MAIN
JNC+$04 CONT
DALT:
GOSUB ALTD
GOSUB KEYDBU
LDI $088 $0F18
ON
?A+C $0006
; If key down was not $0F, then try next key
JCE+$0F CONTA
GOSUB FIND9
; On pressed: find the buffer and top register
LDI $0006
C = C + A $0006
RAM SLCT
C = N
; Get edited status,
C = $0 MS
; Ensure $131 is 1 so register cannot
C = C+1 MS
; be all zero
WRITE DATA
; Write edited values into buffer
GOTO MENU
; Each STEP: GOTO MAIN DEBUG MENU
CONTA:
LDI $088
?A+C $0006
JCE+$05 CONTB
?FS $0F
JCE-24 LOOP1
SF $0F
; If AIT set displayed, then change to MAIN
JNC-25 LOOP
CONTB:
LDI $078
?A+C $0006
JCE+$04 CONTC
JLOOP:
N = C
JNC-2C LOOP
CONT:
C = C+1 $0006
?A+C $0006
JCE+$06 CONTD
C = N
; Get status register
TLOOP3:
PT=7
C=Φ @R
SF 1Φ
JNC-Φ9 SLOOP
LDI φ9
PΛH C S8X
Jc+9 φ6 CONTE
C=N
PT=7
C=Φ @R
C=Φ1 OR
JNC-ΦA TLOOP3
LDI φ13
PΛH C S8X
Jc+9 φ7 CONTF
C=N
M=C
SF 8
GOTO PATCH
Q
CONTF:
C=Φ+1 S8X
PΛH C S8X
Jc+10 CONTQ
Q
QIN:
GOSUB MAIND
GOSUB TKEYH
JNC+9B QIN
SF 11
GOSUB MAIND
GOSUB TKEYH
JNC+9B QIN
CF 11
JNC-10 QIN
RCR 1Φ
B<>C ALL
C=N
PT=4
B<>C @R
NE=C
CF 11
CF 1Φ
GOTO LOOP
LDI φ83
PΛH C S8X
Jc+1A CONTM
PIN:
GOSUB MAIND
GOSUB TKEYH
JNC+9B PPIN
SF 9
GOSUB MAIND
GOSUB TKEYH
JNC+9B PPIN
CF 9
JNC-10 PIN
RCR 9
B<>C ALL
C=N
PT=5
B<>C @R
PPIN:
JNC 7 Indicates HEX φ R DECB 1
Set to HEX i.e. φ
Ensure the appropriate ALT set displayed
and continue - writing status to N
IF NOT 0 pressed then try next
Get status
Display ALT set & write status to N
IF NOT 0 pressed then try next key
Save status
Indicate printer is to be used if 15 set
Print values on screen and restore status.
Save 4 & B
Get status
Check nibble 4 back into status
Register
And sub status
Ensure that 0 is displayed
and that the main display set is on
Jump jump back to status of notice
IF NOT 0 then try next key

Show main display
Get a Hex key value within a second or so
RMM1: Key pressed so process it below
RMM2: No key or invalid key so change
value of φ to an underscore (Flag 9)
Display status for another key
RMM1: Key pressed so process it below
RMM2: No key so change φ back to value
and try again until a valid one is pressed
Put Hex value input into φφφ φφφ φφφ φφφ
Save 5 & B
Get status
Put nibble 5 back into status
Register
JLOOPS:
  N=C
  JNC-3φ JL0OPS
  LDDI φ34
  ?A#C CAJ
  JC+94 CONTENT

CONTIN:
  ?A#C S&X
  RIN:
  2SUB MAIN
  2SUB TKEY
  JNC+93 RAIN
  SF 7
  2SUB MAIN
  2SUB TKEY
  JNC+83 RRIN
  CF 7
  JNC-1φ RIN
  CF 7
  LDDI φ83
  ?A#C S&X
  JNC+φ4 PDOWN
  LDDI φ14
  ?A#C S&X
  JC+94 RIN

QDOWN:
  CN
  PT=6
  C=0 Ω R
  C=φ Ω R
  JNC-24 JLOOPS

PDOWN:
  CN
  PT=6
  C=0 Ω R
  JNC-28 JLOOPS

CONTIN:
  ?A#C S&X
  JC+33 CONTIN

G1:
  2SUB ALTD
  2SUB TKEYM
  JNC+φ1 G2
  SF 4
  2SUB ALTD
  2SUB TKEYM
  JNC+φ3 G2
  CF 4
  JNC-1φ G1
  CF 4
  RCR 11
  B<φC ALL
  PT=3
  C=N
  B<φC Ω R
  N=C

G2:
  2SUB ALTD
  2SUB TKEYM
  JNC+φ3 G4
  SF 3
  2SUB ALTD
  JNC+φ3 G4

G3:
  2SUB ALTD
  2SUB TKEYM
  JNC+φ3 G4

; Ensure IP displayed not underscore
; Some status register & jump back
; with main display set up
; If not 0 then try next key

; Show main set
; Get a key within a second or so
; RIN?1: Key pressed so process it below
; RIN?2: No key so change PT value
; For underscore (length 7) and display set
; Wait for another key for about a second
; RIN?1: Key pressed so process it below
; RIN?2: No key so change PT value
; From ' ' to a value P/4 add to again
; Key pressed to clear underscore flag
; Check 0 if 0 then P pressed

; If for ship down dirk
; Check 0 if 0 then Q pressed

; If neither P nor Q wait for another key
; Q pressed so set rightl6 6 of
; if status to 1 to indicate active
; pointer in Q

; P pressed so set rightl6 0 of
; status to 0 to indicate active
; pointer in P

; If not 0 then try next key

; Show alternative display set
; Get a hex key within a second or so
; RIN?1: Key pressed so process it below
; RIN?2: No valid hex key so underscore
; Get hex key if 0 value and try again
; RIN?1: Key pressed so process it below
; RIN?2: Reset ' ' to 1st rightl6 value
; and try again
; Turn of 1st rightl6 underscore
; and put hex value into status
; rightl6 3 corresponding to top
; rightl6 of G4

; Now get 2nd rightl6 of G
; Get a hex key within a second or so
; RIN?1: Key pressed so process it below
; RIN?2: No valid hex key so underscore
; 2nd rightl6 of G3 value and try again
; Get hex key
; RIN?1: Key pressed so process it below
MAIN DEBUG MENU & FAT ENTRY

DEPENDENCIES: ALL ROUTINES!

ROUTINES USED: CLUDE MESSL ENCPDP CLDSP

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

;X27 MENU:
CLRKEY
?KEY
JCE-02 MENZ
JNC+ MENU

;SOME FUNCTIONS MAY RETURN WITH A KEY
;STILL DOWN SO THEY COME IN HERE AND
;THE CALLING IS DONE BY THESE LINES
;WHEN THE KEY COMES UP 'MENU' IS ENTERED

FUNCTION: DEBUG

;FAT NAME

;X30 MENU:
NC XQ CLUDE
NC XQ MESSL
NC XQ ENCPDP

;DISPLAY PROMPT

;DEBUG CMD ?

;ENABLE CHIP 90

;FORCE OUT ANY KEYPAD FLAG

C=0 ALL
SET UP 2.5 MINUTE TIMEOUT VALUE

PT 4
IN THE C REGISTER (2:42)

LC 4

LOOP:
C=C-1 ALL
DECRAEMENT TIMEOUT VALUE
JCT+7 TIMEOUT
IF CARRY IS SET THEN TURN CALCULATOR OFF

;OTHERWISE LOOK TO SEE IF A KEY IS DOWN

JNC-03 LOOP
IF NO KEY THEN CONTINUE WAITING

C=KEY
GET THE KEY INTO C[68X]

RCR 3
C=0 X5
C=0 X5
A=C S8X
AND FORMATTED INTO A[58X] FOR TESTING

CLRKEY
WAIT FOR KEY TO COME UP BEFORE

?key
; DOING ANYTHING ELSE

JCE-02

LDI 013 016 ON

PA[0] S8X

JCT+3 CON1

;if it was not the on key try the next

EXIT:
C=REG 12(c)
OTHERWISE TIDY UP EVERYTHING AND EXIT

C=0 M
to NORMAL OPERATIONAL MODE

PT=3
LC 3
CF 10
SF 13
REG=C 12(b)
NC 40 CLDSP (HEX)

TIMEOUT:
C=REG 13(c)
IF TIMED OUT THEN TIDY EVERYTHING UP
C=0 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 10
IN FACT AND THE POWER DOWN APPL
SF 13
ROUTINE IS RUN IF IT IS PRESENT
REG=C 12(b)
PRESERVING BATTERIES IN ANY LOOP DEVICES
NC 40 OFF (ICB)
CON1: LDI $031
  ?A+C S&X
  JC+04 CON2
START1: GOTO START
CON2: LDI $0C0
  ?A+C S&X
  JNC-06 START1
  LDI $74
  ?A+C S&X
  JC+04 CON3
  GOTO STED
CON3: LDI $011
  ?A+C S&X
  JC+04 CON4
  GOTO Fлаг
CON4: LDI $070
  ?A+C S&X
  JC+04 CON5
  GOTO CNTBK
CON5: LDI $038
  ?A+C S&X
  JNC+ BKED
  LDI $34
  ?A+C S&X
  JC+04 CON6
  GOTO RED
CON6: GOTO PRECPU
    ; SEE IF ENTER? PRESSED & PRINT CPU REGISTERS
    ; IF IT WAS, ELSE JUMP BACK TO MENU
BREAKPOINT EDITOR - BKED

DEPEN DENCIES: EDN DAGKPT
              EDGN ENBKPT
              GETHKA MENU
              SURE DES

ROUN TES USED: GOSUB
                PCTDC
                MSGI"F5

INPU T: NOVE - Enter via MENU

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

xA95 EDN:
xA98 BKED:
GOTO EDN
GOSUB DES ; Enter low level edit of breakpoint N
GOSUB GETHKA ; Indicate status BKPTS ON/OFF/NONE
PFEET 2 ; Get hexadecimal heapers
JC+2D SPEC ; If special heapers pressed, jump
LDC HEX "$A
PA<=C S&X ; If "A" or " $ then set carry
PA+SC S&X
JC+17 BCL
GOSUB SURE ; Otherwise ask if user is sure. If he
JC- BKL BKED ; wants to erase BKPTS and ignore if not
GOSUB DAGKPT ; To delete, disable breakpoints, and
NC XQ PCTDC ; Clear every word in range xF00 - xF3B
PT = 5
LDER B
LDER 3
LD@R B
C=0 S&X
SLCT Q
R=4
SLCT P
R=3

CLRL: WRDM
      C=C-1 P-Q
JNC-02 CLRL ; Done when carry of xF00 - xFF

JBKED: JNC- BKL BKED ; Jump back to status display
BC1:  C=C+1 S&X ; D passed
      ?AhC S&X
      JC+05 BC2
      GOSUB DAGKPT ; disable breakpoints
      JNC-07 JBC1
BC2:  C=C+1 S&X ; E passed
      ?AhC S&X
      JC- $A JBC2
GOSUB ENBKPT ; enable breakpoints
JNC-DE JBKED

SPEC: LDI HEX 06 ENTER ; If ENTER pressed
?A#C S&X
JC+05 SPEC1
SETF 8
NC XO MSG105 ; then print the screen
JBKED2: JNC-16 JBKED

SPEC2: LDI HEX 01 ON ; If on pressed
?A#C S&X
JC-04 JBKED2
xRDB
GOTO MENU ; then jump back to main menu
SUBROUTINE - GET NYBBLE FROM BUFFER 9

DEPENDENCIES: NONE

ROUTES USED: NONE

INPUT: A[S & X] = HEX \( P \) \{ + offset \} \{- offset \}

N REGISTERS CONTENTS

MS = NYBBLE BEING EDITED

BBB IN [H] = BASE ADDRESS OF BUFFER 9

R = REGISTER BEING EDITED IN [RS]

BUFFER 9 IN CPU FORMAT

OUTPUT: N UNCHANGED

BUFFER UNCHANGED

NYBBLE IS IN C[MS]

APPROPRIATE REGISTER IN BUFFER 9 IS DELETED

C[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 14[M][MS] INTO A[MS]

LC E

A=E

C=N

; SET EMBR POINTERS INTO C

PT=1

; POSITION ACTIVE POINTER TO NYBBLE 1

LOOP1: ?A\#0 PT

; IF A[0] = 0 THEN DONE + EXT OFFSETS

INC+0B 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] \#H DEC THEN GOTO LOOP1

JNC-05 LOOP1

C=C+1 XS

; ELSE INCREMENT REGISTER NUMBER

C=0 MS

; AND POSITION NYBBLE POINTER TO

JNC-02 LOOP1

; POSITION OF REGISTER, SO TO LOOP1

PART2:

PT=0

; POSITION POINTER

LOOP2: ?A\#0 PT

; IF A[0] = 10 THEN DONE - EXT OFFSETS

JNC+0B PART3

; SO GO TO PART3

A=A-1 PT

; DECREMENT OFFSET

C=C-1 MS

; DECREMENT NYBBLE NUMBER

JNC-04 LOOP2

; IF CANDY CLEAR GO TO LOOP2

C=C-1 XS

; OTHERWISE POSITION POINTERS TO

PT=13

; PREVIOUS REGISTER AND TOP NYBBLE

LC D

JNC-09 PART2

; SO TO PART2 (AS POINTER WAS MOVED)

PART3:

A=ALL

; PUT NYBBLE & REGISTER IN A

B=A MS

; PUT NYBBLE NUMBER IN B FOR LATER USE

RCL 13

C=C+A M

; ADD BASE ADDRESS OF BUFFER & REGISTER

C=0 S\&X

; SELECT PERIPHERAL O

PAH SLCT

RCL 3

; SET ADDRESS OF REGISTER

RAM SLCT

; SELECT IT

READ DATA

; AND READ IT

LOOP 3:

RCL 1

; ROTATE CONTENTS [MS] TIMES

A=A-1 MS

; UNTIL Digit REQUIRED IS IN C[MS]

JNC+02 LOOP3

; REGISTER IS STILL SELECTED

RTN

; C[MS] CAN BE USED TO PUT NEW VALUE N
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, S8X
B S8X, M
C ALL
G, PT, M, ST

Output: M UNCHANGED
N UNCHANGED
PT = Φ
APPENDS HEX DIGITSlington DISPLAY

X814  OUTHEX: C, ST ; SAVE STATUS IN B[M]
RCR 3
C<→B M
C=M ; GET DISPLAY INFO
A=C S8X ; A[S8X] = 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
B=A S8X ; B[S8X] = NUMBER OF DIGITS NOT YET DISPLAYED
P ?A+Φ S8X ; IF NO MORE DIGITS TO BE DISPLAYED
JNC+24 UNS ; GO DOWN AND OUTPUT UNDERSCORES
PT=Φ ; OTHERWISE POINT POINTER AT NEXT DIGIT
OUL: ?A+Φ S8X ; TO BE OUTPUT, BY COUNTING DOWN A[S8X]
JNC+04 ; AND MOVE PT UP FROM Φ UNTIL
+PT ; ACC[S8X] = Φ
A=A+1 S8X
JNC-Φ4 OUL ; GET DIGITS AT PT AND PT+1
G=C
PT=Φ
C=Φ ALL
C=G ; WRITE THEM TO C[16Φ]
+PT
LC Φ ; CLEAR C[1] LEAVING PT EΦ & DIGIT [Φ]
ST=C ; PUT DIGIT IN STATUS REGISTER
?FS 3 ; IF Φ OR ABOVE SET THE CARRY FLAG
JNC+Φ8 NUM ; IF 0-7 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 DTU DIGIT SAVING 6-3
ST+C ; STORE IN STATUS
C=C+1 PT ; ADD 2 TO
C=C+1 PT ; RESTORE VALUE
?FS 3 ; IF Φ(A) OR ABOVE SET THE CARRY FLAG
JNC+Φ4 LET ; MUST BE A-F SO Φ & DO LETTER OUTPUT
NUM: +PT ; PT=1, WRITE
LC 3 ; IN 3 GIVING CORRECT DISPLAY
JNC+Φ5 OUT ; CODE 30-39 FOR NUMBERS & OUTPUT
LET:  A=R C  S8K  ; MUST BE A LETTER A-F SO
LDD  ; SUBTRACT A TO GIVE \Phi 1 - \Phi 6
\Phi 9  ; THE CORRECT DISPLAY CODE
C=A-C S8K  ; IN C[S8K]
OUT:  WRA8CLR  ; OUTPUT DIGIT TO RHS OF DISPLAY
C=B S8K  ; SET NUMBER OF DIGITS NOT YET DISPLAYED
C=C-1 S8K  ; SUBTRACT ONE FOR DIGIT JUST OUTPUTED
A=C S8K  ; MAKE THAT THE NEXT ONE TO BE SHOWN
C=M  ; PUT M INTO C READY FOR IT
JNC-25 Ouh  ; GO BACK AND DO IT AGAIN
UNS:  C=B M  ; SET ST BACK FROM B[M]
RCL 11  ; PUT IN C[1110]
ST=C  ; RESTORE STATUS
USL:  ?A #F MS  ; ANY MORE UNEESSED TO OUTPUT?
NC RTN  ; NO:
LDD  ; EXIT
\Phi F \ldquo'\rdquo  ; YES: LOAD CHARACTER READY
WRA8CLR  ; OUTPUT UNEESES
A=A-1 MS  ; DECREMENT UNEESED COUNTER
JNC-\Phi 7 USL  ; SO AND DO IT AGAIN
SUBROUTINE - ENTRY ROUTINE 'START'

DEPENDENCIES: ADDDIG, DELDIG, OUTHEX, GETHEX, WRC1, GETDRS, B9>CPU, ENTRY

Routines used: MESSL, CLCDE, GOSUB, PCTOC, LEFTJ

Input: No registers
Prompts for address of code to be entered, accepts next digits and deletes key. RIS 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.

; START:
C=Φ ALL
; LOAD M WITH 4 IN [M[S]
PT=13
; AND REST OF WORD CLEAR
LC 4
; READY FOR OUTHEX TO INDICATE
M=Φ
; 4 underscores & NO DIGITS YET
ENT:
NC XQ CLCDE
; CLEAR & ENABLE LCD
NC XQ MESSL
; OUTPUT MESSAGE
ALPHA:
ENTRY 0, B
; 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#Φ MS
; THEN FINISH WITH EITHER RIS
JNC+Φ 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#Φ SX
; 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 RIS TO BE PRESSED
J+C#Φ DEL
; IF FLAG 4 SET MUST HAVE BEEN 0
NC XQ MSG10S
; OTHERWISE: PRINT ENTRY ADDRESS ON
NC XQ CLCDE
; ANY PRINTING DEVICES, CLEAR & ENABLE
NC XQ ENCPD0
; THE DISPLAY, ENABLE CHIP ΦΦ AND
NC XQ RSTSQ
; RESET SYSTEM FLAGS ETC.
CF Φ
; INDICATE XQ
CF 1
; INDICATE NC
C=Φ ALL
NC XQ PCTOC
; PUT THIS ROM'S PAGE NUMBER IN C[6]
PT=5
; PUT IN C[ADDR]
LC 8
; THE ADDRESS OF
LC F
; ROUTINE 'ENTRY' WHERE
LC 3
; THE INSTRUCTION IS TO BE WRITTEN
A<Φ ALL
; SAVE IT IN A[ADDR]
C=M
; GET ADDRESS FROM M AND TIDY
C=Φ MS
; IT UP FOR WRC1 SO THAT
M = C ; M[8:0] 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 # & X S&X ; ABDORT START AND RETURN TO MAIN
JC-31 KEY ; MENU HAVING WAITED FOR ON KEY TO BE RELEASED
GOTO MENZ2

SUBROUTINE - GET DELETE OR R/S 'GETDRS'

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

X89A GETDRS: GOSUB KEYDBU ; SET KEY & WAIT FOR IT TO COME UP
LDI 8C3 ; LOAD [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 8B7 ; LOAD [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

XBA2 JNC- GETDRS ; ELSE GET ANOTHER KEY
SUBROUTINE - BUFFER TO CPU & CODE ENTRY & RTN

DEPENDENCIES: MLOC8 @
ROUTINES USED: GOSUB
INPUT: BUFFER & COMPARE CPU STATUS
OUTPUT: CPU REGISTERS SET UP, CODE ENTERED AT 'ENTRY'
USES: ALL REGISTERS AFFECTED

 olmad o

x868

097CPU:

* IC XR GOSUB
DEF MK9
(JMAKE &) LOCATE
BUFFER 9

CR9:
LDI
HEX $06
C=C+4 $8X
RAM SLCT
A=A+C $8X
READ DATA

RESTOR:
ST=C
PT= 2

RESTRO:
G = C
PT= Q

RST S
PT= 0

RESET

LPQ:
?C=0 MS
JNC+94 LTP
; YES &.do P NEXT

+PT
; NO & INC PT POSITION

C=C+1 MS
J NC-04 LPQ

LTP:
RCR 1
PT= P

PT= 0

RESET

LPP:
?C=0 MS
JNC-04 LPP

CR92:
RCL 1

PT= P

CR93:
CF
RCL 1

?C=0 MS
JNC+02 CR93

IF M CARRY ACTIVE PT IS P ANYWAY SO JUMP

CR94:
SF
A=A-1 $8X

A=A+C $8X

READ DATA

; SELECT
A=A+C $8X

; PUT ADDR BACK IN A[$8X]

; AND READ VALUE
RESTM:
N=C
A=A-1
M=M-1
RAM SELCT
A->C
READ DATA

RESTB:
C<>B
A=A-1
A<>C
RAM SELCT
A->C
READ DATA

RESTA:
A<>C
C=C-1
RAM SELCT

RESTC:
READ DATA
PUSH
RCR 3
C=0
PRPH SELCT
RAM SELCT
RCR 11
POP
IFS 10
ENTRY
ENTRY
ENTRY

XBF3 ENTER:
NOP
NOP
RTN:
CF 8
JNC+2 RB2
BPIF:
SF 8
RB2:
NOP

xC00 CPU->B9:

; put it in N
; put next in M
; put next in B
; put next in A
; last one in C
; using part of C[ADDR] but without
; destroying it enable peripheral and
; chip 10
; restore C
; restore hex/dec mode
; if no carry then hex mode - enter
; otherwise set decimal mode
; restore flag 10
; two words written to by WRCL TAP
; contain the two word IN X3 TO CODE UNDER TEST
; repeat IN THE CODE COME BACK HERE
; clear flag 8
; BPIF'S X3 BACK HERE & SET FLAG 8
; free space

; see next routine
SUBROUTINE CPU REGISTERS -> BUFFER 9

Dependences: MLOCB @ FPT@

Routines Used: PUSH

Input: FROM CODE UNDER TEST - ALL REGISTERS

Output: BUFFER 9 CONTAINS CPU REGISTERS IN DEIMED FORMAT

XCF0 CPU>99:

PUSH ;SAVE C[6:3] ON STACK

RCR 3 ;ROTATE THEM DOWN TO [3:0]

CFF SAK; USE [2:1] TO SELECT

PPHF SLCRT; PERIPHERAL AND

RAM SLCRT; CHIP @

RCR 11; RESOLVE TO [6:5]

POP ;RESTORE ORINAL VALUE

WRITE 11 (a); SAVE C IN (a)

AEC ALL; SAVE A

WRITE 12 (a); IN (b)

CF 10; CLEAR HEX/DECIMAL FLAG

C=qO SAK; DO q-1 IN CURRENT MODE

C=qC SAK

C=qST; PRESERVE STATUS W.C., ANSWER IN ST

PES 1; IF BIT 2 IS SET MUST BE IN HEX MODE

TThan CONT; SO CONTINUE (Aq RESULT MUST BE F NOT q)

SF 10; OTHERWISE SET FLAG TO INDICATE DEIMAL

CONT;

C=qST; RESTORE STATUS REGISTERS

SETDEC; FIND BUFFER 9 (DONE LIKE THIS TO

A=qO MS; PRESERVE THE STCK & ENSURE THAT

A=qA+1 MS; IF BUFFER 9 DOES NOT EXIST IT IS

SEPHEX; NOT AUTOMATICALLY CREATED - BUT THE

* USER IS INFORMED BY THE NO BUFFER

NC XR GsovB; ERROR)

DEF MLOCB

JNC0+14 CS9; FOUND: CONTINUE BELOW WITH ADDRE IN A[S]AK

* NF: GIVE "NO BUFFER" ERROR INSTEAD THIS DOES

NC XR GOTO; NOT EXIT IN THE BEST WAY SO A GOTO REQUIRED

DEF ERRNB; TO PUT EVERYTHING BACK IN ORDER

CS9:

C=qO SAK; SELECT CHIP @

RAM SLCRT

A=qA+1 SAK; POINT TO 1st REG. OF BUFFER

READ 11 (a); RESTORE CPU.C

AEC ALL

RAM SLCRT; SELECT REGISTER

AEC ALL

WRITE DATA; SAVE CPU.C IN BUFFER

C=qO SAK; SELECT CHIP @

RAM SLCRT

A=qA+1 SAK; POINT TO 2nd REG. OF BUFFER

READ 12 (b); RESTORE CPU.A

AEC ALL

RAM SLCRT; SELECT REGISTER

AEC ALL

WRITE DATA; SAVE CPU.A IN BUFFER

AEC ALL

C=qC+1 SAK

RAM SLCRT

C=qC ALL

WRITE DATA; SAVE CPU.B IN BUFFER

C=qC ALL
C = C + 1 SAK
RAM SLCT
C = M
WRITE DATA ; SAVE CPU.M IN BUFFER
C = M
C = C + 1 SAK
RAM SLCT
C = N
WRITE DATA ; SAVE CPU.N IN BUFFER
C = N
C = C + 1 SAK ; POINT TO FINAL REGISTER
C = B SAK ; SAVE IN B[S.BX]
CF 12 ; FLAG 12: C = 0 THEN S = P = Q
CF 9 ; FLAG 9: C = 0 THEN S = P = Q
?P = Q ; IF POS(P) = POS(Q) THEN SET CARRY
JNC + PH2 PNQ ; NOT THE SAME SO DON'T
+PT ; INCREMENT POINTER
SF 12 ; IF SAME INC PT & SET FLAG 12

PNQ:
NC KQ GOSUB
DEF FPT ; FIND POSITION OF ACTIVE (PT) ⇒ C[MS]
A = C MS ⇒ A[MS]
PT = P ; SELECT P

CLPT:
NC KQ GOSUB
DEF FPT ; FIND POSITION OF P ⇒ C[MS]
?A # C MS ; IF POS(P) # POS(PT) THEN SET CARRY
JNC + PH2 CLPT ; IF SAME CONTINUE BELOW
SF 9 ; OTHERWISE INDICATE PT = Q ACTIVE
SF 12 ; PUT POS(P) IN A[MS]
PT = Q ; SELECT Q

PTQ:
NC KQ GOSUB
DEF FPT ; FIND POSITION OF Q ⇒ C[MS]
?FS 12 ; IF P = Q ORIGINALLY THEN SET CARRY
JNC + PH5 DLPT ; IF THEY WERE TERN THEN END
?FS 9 ; IF P = Q ORIGINALLY THEN SET CARRY
JNC + PH2 PTQ ; IF PT = Q WAS P JUMP
PTP:
A = C MS ; SWAP POS(P) & POS(Q) (WHICH WAS INCREMENTED)
PTP 12 ; END PT PART
DLPT:
C = B SAK ; SELECT TOP REGISTER OR BUFFER
RAM SLCT
C = B SAK
CF 12 ; END PT PART
MAKE BUFFER 9 SUBROUTINE - MK9

DEPENDENCIES: MMK9F
Routines used: NONE
Input: NONE
Output: BUFFER 9
Uses: ALL REGISTERS

X9A MK9:
C=0
ALPHA: C
PT=13
LC 9
LC 9
LC 0
LC 7
N=C
GOSUB MMK9F

MAKE BUFFER 9 SUBROUTINE - MK9

FIND BUFFER 9 SUBROUTINE - FIND9

XCA4 FIND9:
SETDEC
A=0
A=A-1
MS
SETHEX
GOSUB MLO9B
RTN

XCA4 ERANG:
NC XQ ERRSUB	GIVES 'NO BUFFER' ERROR
NC XQ CLCDE	CLEAR & ENABLE LCD
NC XQ MESSL	OUTPUT ERROR MESSAGE

XCC0 ALPHA:
NC XQ LEFTJ	LEFT JUSTIFY DISPLAY
NC XQ MSG105	PRINT MESSAGE ON TRACE PRINTERS
**Find User Flag Subroutine - Fred**

**Dependencies:** None

**Routines Used:** INTINT

**Input:**
- BCD Flag Number in M[2:1] for Fred
- Binary Flag Number in A[S&X] for IF

**Output:**
- Leaves a containing A 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 0) to clear bit (SET → CLR)

**Uses:** A, B, C, M, D

xC75 Fred:
- C = M
- ; Get BCD flag number into C[2:1]
- RCR 2
- ; Put MSN in C[3] and LSN in NC XQ INTINT
- ; C[M]. Combine two INTO C[3:0] binary
- A = C
- S&X
- ; Binary flag number in A[S&X]
- xC7A IF:
- NC XQ ENCP +Φ
- ; Enable chip Φ (status registers)
- LDIF φ0φ8
- C<>B S&X
- ; B in B[S&X]
- C = φ ALL
- ; Clear C
- C = φ1 S&X
- ; Set bottom bit to 1
- RCR 2
- ; Move to next byte and subtract B
- A = A - B S&X
- ; From flag number each time
- JNC -Φ2
- ; If not there yet move to next byte
- JNC +Φ2
- ; Carry: Don't shift for first flag
- C = C + C ALL
- ; Shift bit one place left
- A = A + 1 S&X
- ; Add 1 to (-3B) value until there
- JNC -Φ2
- ; Is a carry back to zero
- A = C ALL
- ; Put mask into A
- READ λ
- ; Set flag result
- C = C AND A
- ; And the flags & mask
- C<>B ALL
- ; Put result in B
- READ λ
- ; Set flags & put them in A[ALL]
- A<>C ALL
- ; While bit mask goes into C[ALL]

XBE
- RTN

**Get Hexadecimal Key Press in A, B & C - Gethx**

**Dependencies:** Gethex

**Routines Used:** None

**Input:** None

**Output:** Decode value (as Gethex) in A, B, C[S&X]

**Uses:** All Registers & Flags as Gethex

xC0F Gethx:
- GSUB Gethex
- ; Get hex key press
- A = C S&X
- ; In addition to B & C put it in A[S&X]
- CLR KEY
- ; Wait for key to come up
- ?KEY
- JC -Φ2
- RTN
SUBROUTINE - WRITE CLASS 1 INSTRUCTION - WRC1

DEPENDENCIES: NONE

ROUTINES USED: NONE

INPUT:
M [3:0] CONTAINS ADDRESS FOR CLASS 1 INSTRUCTION
A [ADDR] CONTAINS ADDRESS THAT WILL HOLD THE INSTRUCTION
FLAG 0 CLEAR; XR SET; GO
FLAG 1 CLEAR, CARRY CLEAR; SET; CARRY SET

OUTPUT:
M UNCHANGED
A [ADDR] DESTROYED
C [ALL] 2ND WORD OF INSTRUCTION
FLAGS 0 & 1 UNCHANGED
ADDR SPECIFIED IN A[ADDR] CONTAINS 1ST WORD

ACC1 - WRC1:
C=M

SET LOWER BYTE
C=0
XS
C=C+C S\phi X
C=C+C S\phi X
; MULTIPY BY 4 TO SHIFT LEFT 2 BITS
C=C+1 S\phi X
; SET BIT 0 TO SET SAME CLASS 1
A<X C M
; GET ADDR
WRAM
; WRITE 1ST WORD
C=C+1 M
; INCREMENT
A>X C M
; AND SAVE ADDRESS
C=M
; SET
RCL 2
; UPPER BYTE
C=0
XS
C=C+C S\phi X
C=C+C S\phi X
; ARITHMETIC SHIFT LEFT 2 BITS
?FS \phi
; IF 'GO' THEN SET CARRY
JNC H\phi 3 CARRY
; IF NO CARRY - JUMP
C=C+1 S\phi X
; ADD 2 (SET BIT 1) TO GIVE 'GO'
C=C+1 S\phi X
; RATHER THAN 'XQ'
CARRY:
?FS 1
; IF 'C' THEN SET CARRY
JNC H\phi 2 WRITE
; IF NO CARRY WRITE WORD
C=C+1 S\phi X
; ELSE ADD 1 (SET BIT 0) TO GIVE 'C'
WRITE:
A>X C M
; RATHER THAN 'NC'
WRAM
; WRITE 2ND WORD OF INSTRUCTION

\textsc{xcd8}

RTN
; END
SUBROUTINE - DELETE DIGIT - DELDIS

DEPENDENCIES: NONE
ROUTINES USED: NONE
INPUT: M contains number of underscores in \([m_5]\)
NUMBER OF DIGITS IN \([6_7]\)
DIGITS IN \([1_7]\) UPWARDS
OUTPUT: AS INPUT BUT WITH DELETED DIGIT

XCEB DELDIS: C=M ; GET DIGIT ENTRY
PT=0
?C\#P PT ; IF NO DIGIT PUT IN YET
NC RTN ; THEN IGNORE DELETE REQUEST & EXIT
A=C MS ; SAVE NUMBER OF PROMPTS IN \(A[m_5]\)
C=P MS
A=C PT ; SAVE NUMBER OF DIGITS IN \(A[6_7]\)
RCR 1 ; REMOVE LAST DIGIT
A=C+1 MS ; RESTORE
A=C PT
C=C-1 PT ; ONE LESS DIGIT
C=C+1 MS ; ONE MORE PROMPT
M=C ; RESTORE M
XCFP
RTN

SUBROUTINE - ADD DIGIT - ADDDIG

DEPENDENCIES, ROUTINES USED, INPUT AS ABOVE
OUTPUT: AS INPUT BUT WITH DIGIT FROM \(B[6_7]\) ADDED

XCE9 ADDDIG: C=M ; GET DIGIT ENTRY
PT=0
A=C MS ; SAVE NUMBER OF PROMPTS
A=C PT ; AND NUMBER OF DIGITS
C+B PT ; PUT IN NEW DIGIT FROM \(B[6_7]\)
RCR 13 ; AND MOVE LEFT 1 DIGIT
A=C+1 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[X$S$] CLEAR
- C[MS] HEX digit representing position of active pointer PT. Active Pointer P and Q unchanged

**Registers Used:** C[X$S$], C[MS] only

**FPT:**

```
C = 0 MS ; CLEAR COUNTER IN C[MS]
C = 0 XS ; CLEAR DUPLICATE COUNTER IN C[X$S$]
PT = 0 ; SET CARRY IF PT = 0
JC + 05 DFPT ; PT @ 0, PUT IT BACK & END
- PT ; DEC PT POSITION
C = C + 1 MS ; INC COUNTERS
C = C + 1 XS
JNC - 05 LFPT
?C # 0 XS ; POSITION IN C[MS] & C[X$S$], USE NC RTN ; C[X$S$] TO PUT C BACK TO CORRECT + PT ; POSITION USING THIS LOOP AND
C = C - 1 XS ; ENDING WHEN C[X$S$] IS ZERO
```

**Flowchart Diagram:**

(Find Pointer)

CLEAR COUNTER.
CLEAR DUPLICATE COUNTER.

IS PT @ 0 ?

NO  DEC PT POSITION.
    INC BOTH COUNTERS

YES

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

(Answer is in C[MS])
LOCATE BUFFER SUBROUTINE - MLOCB

DEPENDENCIES: NONE
Routines used: NONE
Input: BUFFER ID IS IN A[MS]
Uses: C[ALL], A[ALL]
Assumes: ALL OTHER REGISTERS, FLAGS & POINTERS UNCHANGED
Assumes: HEXMODE

MLOCB: LDE 0BF ; BUFFER AREA STARTS AT 0CF
A=C S&X ; SAVE REGISTER ADDRESS IN A[S&X]
INCADR A=A+1 S&X ; INCREMENT RAM ADDRESS
CONL S&X ; ENABLE CHIP I/O TURN OFF PERIPHERALS
PR PN SLCT ; CANNOT CALL SKIP HERE AS STACK
RAM SLCT ; MUST BE LEFT UNCHANGED
READ 13(c) ; LOAD USER STATUS c INTO C
?ACC S&X ; IF RAM = END THEN NO SUCH BUFFER
JNC+11 NOBUF R ; 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=#0 ALL ; IF ZERO THEN NO MORE BUFFERS
JNC+0B NOBUF R ; (BUT AT LEAST 1 FREE REGISTER)
C=CH MS ; INCREMENT TOP NYBBLE TO GET A
JC=0E INCADR ; CARRY IF STILL IN KEY AREA
C=C-1 MS ; RESTORE IT TO CORRECT VALUE
RCL C=13 ; PUT LOWER NYBBLE OF ID BYTE INTO
?A [#C MS ; C[S&X] AS OS MAY HAVE SET TOP TO 0
JNC+0B LBEND ; END IF IT IS
RCL C=11 ; OTHERWISE SET SIZE INTO C[S&X]
C=0 X5 ; ENSURING C[X5] IS CLEAR
A=A+4 S&X ; ADD SIZE & RAM ADDR TO GIVE NEXT HEADER
JNC+15 CONL S&X ; AND CONTINUE AT NEW HEADER
NOBUF R ; POP IF NOT FOUND THEN RETURN TO
C=CH M ; RETURN ADDRESS + 1
GOTO ADR ;
LBEND RCR 1 ; FOUND BUFFER SO RESTORE IT AND
RTN ; RETURN TO NORMAL RETURN ADDRESS.
USER FUNCTION - LCBUF

DEPENDENCIES: MLOCGB
ROUNTRIES USED: BDDBIN, ERRNE, ENCPH0, APNDNW

USER INPUT: BUFFER ID VALUE 0-15 (INCLUSIVE) IN X REGISTER
USER OUTPUT: X - UNCHANGED

ALPHA - LAST TWO DIGITS ARE ADDRESS OF BUFFER IN THE FORM TO BE USED BY NCCLM, RAMED OUT.

ERRORS:
ALPHA DATA - IF X CONTAINS NON-NUMERIC DATA
NONEXISTENT - IF BUFFER DOES NOT EXIST OR VALUE IN X IS OUT OF RANGE 0-15

LD X 36 F
LD X 15 U
LD X 02 B
LD X 03 C
LD X 06 L

LCBUF:
READ 3(x) ; GET BUFFER NUMBER
NC XQ BDDBIN ; CONVERT TO BINARY (ERRNE IF X>999 OUT)
RCR 1 ; PUT LSN INTO C[MS]
C=0 X5 ; CLEAR X[5]
?C#0 S&X ; IF ID>15 THEN
JC+06 ELBC ; GIVE ERROR - NONEXISTENT
A=C MS ; PUT THE VALID ID INTO A[MS]
GOSUB MLOCB ; LOCATE THE BUFFER
JNC+03 UBC ; FOUND IT - SO CONTINUE BELOW

ELBC:
NC GO ERRNE ; NOT FOUND - GIVE ERROR - NONEXISTENT

UBC:
NC XQ ENCPH0 ; Deselect BUFFER HEADER ID
A>C S&X ; GET RAM ADDR IN C[S&X]
C=0 M ; ENSURE NO RUBBISH IN C[3]
Pt=2 ; GET TOP BYTE INTO G
N=C ; SAVE REST IN N
G=C
NC XQ APNDNW ; APPEND (WITH NO WARNINGS) TOP BYTE
C=N ; INTO ALPHA REGISTER
Pt=0
G=C ; SET BOTTOM BYTE INTO G

LD X 06 F
NC GO APNDNW ; APPEND (WITH NO WARNING) BOTTOM BYTE
MAKE & INITIALISE BUFFER - MKBUF USER FUNCTION
- MMKBUF M-CODE SUBROUTINE

MKBUF
INPUT FROM USER: M REGISTER CONTAINS BUFFER HEADER
OUTPUT TO USER: BUFFER CREATED & REGISTERS WITHIN
IT SET TO '00 00 00 00 00 00 00 00 00 00 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 HAS SPECIFIED AS 0.

MMKBUF
DEPENDENCIES: MLOCB
ROUTINES USED: ENCPKF 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 '00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00'
; EXCEPT FOR THE HEADERS WHICH IS AS SPECIFIED.

XD40 FUNCTION MKBUF
XD45 MKBF: READ 5(M) ; GET HEADER FROM ALPHA M REGISTER
XD47 MMKBF: N=C ; STORE IN N
AD48 C=N ; GET HEADER FROM N
XD51 RCR 10 ; CHECK BUFFER SIZE ≠ 0 AND
XD53 C<>B S<;X ; PUT IT INTO B[5<;X]
XD55 B=0 XS
XD58 P<;BF S<;X
EMM: NC GO ERRDE ; IF SIZE ≠ 0 OR INVALID SIZE 'DATA ERROR'
XD5E GOSUB MLOCB ; FIND BUFFER
XD61 JNC+7 CH=SZ ; IF IT EXISTS CHECK ITS SIZE
XD64 A=A+B S<;X ; ELSE CHECK THERE IS ENOUGH RAM
XD67 A=A-1 S<;X ; FOR A BUFFER THIS SIZE
XD6A NC XQ ENCPKF
XD6C READ 13(C)
XD6E R<;C S<;X ; IF (REQ+SIZE-) BE END. THEN THERE
XD70 NC GO PACKE ; IS NOT ENOUGH ROOM SO 'PACK'
XD72 A=A+1 S<;X ; AND 'TRY AGAIN'
XD74 A=A-B S<;X ; ELSE PUT START OF BUFFER
XD77 A<>C S<;X ; INTO C[5<;X] AND A[5<;X]
XD7A A=C S<;X
XD7C RAM SLTI ; SELECT (WRITE) REGISTER TO PUT
XD7E C=N ; HEADER IN
XD80 WRITE DATA
XD82 JNC+7 CLRBF1 ; CLEAR BUFFER
CHKSIZ
RCA 10 ; C[1:0]=PRESENT SIZE (REQUIRED SIZE
C=0 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+05 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#0 S&X ; IF ZERO THEN ALL DONE
EXIT: NC GO ENCRO0 ; SO END WITH CHIP 00 ENABLED
A=A+1 S&X ; OTHERWISE INCREMENT ADDRESS
C=0 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 & SET 'BLANK'
WRITE DATA ; WRITE 'BLANK' BUFFER REGISTER
×DP6 JNC-0D CLRBF1 ; GO BACK & CONTINUE UNTIL ALL DONE
USER FUNCTION - BUF?

DEPENDING ON : MLOCB @ xD6F
ROUTINES USED : NOSKP, SKP, BCDBIN, GOSUB
INPUT : BUFFER ID (SINGLE) DIGITS 0-15 IN X
OUTPUT : ID IS UNAFFECTED IN X

IF THE BUFFER EXISTS - DISPLAYS 'YES' FROM KEYPAD
OR DOES NOT SKIP NEXT LINE IN PROGRAM.
IF THE BUFFER DOES NOT EXIST - DISPLAYS 'NO' OR SKIPS NEXT LINE.

xD77  0BF  ?;
      066  F
      0815  U
      082  B

xD78  BUF? READ 3 (X) ; GET BUFFER NUMBER FROM X
      NC XR BCDBIN ; CONVERT TO BINARY
      RCR 1 ; PUT INTO [MS]
      C=0 XS ; ENSURE LESS
      $?C#0 $A X ; THAN 16
      JC+06 SK ; BUFFER CAN NOT EXIST IF X > 15
      00 A=C MS ; ENTER PARAMETER FOR MLOCB
      GOSUB MLOCB ; LOCATE BUFFER
      06 JNC+03 NSk ; DON'T SKIP IF IT EXISTS
      SK ; NC GO SKP ; SKIP IF IT DOESN'T

xD8A  NSk: NC GO NOSKP ; NO SKIP

USER FUNCTION - CLRBUF

DEPENDING ON : MLOCB @ xD6F, CLRBUF2 @ xD6A {Part of MMRKRF}
ROUTINES USED : BCDBIN, GOSUB, ERANE
INPUT : BUFFER ID (SINGLE) DIGITS 0-15 IN X
OUTPUT : ID IS UNAFFECTED IN X

IF THE BUFFER SPECIFIED DOES NOT EXIST - GIVES ERROR 'NONEXISTENT'
OTHERWISE - WRITES '00 00 00 00 00 00 00 00' INTO EACH REGISTER IN
THE BUFFER EXCEPT THE HEADER WHICH IS UNCHANGED.

xD88  0BF  F;
      015  U
      082  B
      0812  R
      081C  L
      083  C

1  CLRBUF: READ 3 (X) ; GET BUFFER NUMBER FROM X
      NC XR BCDBIN ; CONVERT TO BINARY
      RCR 1 ; PUT INTO [MS]
      C=0 XS ; ENSURE LESS
      $?C#0 $A X ; THAN 16
      JC+06 ERCLB ; BUFFER CAN NOT EXIST IF X > 15
      00 A=C MS ; ENTER PARAMETER FOR MLOCB
      GOSUB MLOCB ; LOCATE BUFFER
      06 JNC+03 CLBC ; FOUND SO CLEAR IT
      ERANE: NC GO ERANE ; NOT FOUND GIVE 'NONEXISTENT'
      00 CLBC: RCR 16 ; ADDR IS IN A$ #X
      C=0 XS ; SIZE IS IN C$ #X

xD81  JNC-37 CLRBUF2 ; CLEAR BUFFER
USER FUNCTION - DELBUF

DEPENDENCIES: MLOCB @ XDB0, ERCLB @ XDBD

ROUTINES USED: BCOSIB, GOSUB, ENCPX, PKIAS

INPUT: BUFFER ID (SINGLE) 0-15 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.

XDB2 0 96 F: 1
3 15 U
4 82 B
5 0C L
6 05 E
7 04 D

XDB8 DELBUF: READ 3 (X) ; SET BUFFER NUMBER OUT OF X
9 NC XQ BCOSIB ; CONVERT TO BINARY
8 ECR 1 ; PUT INTO MS
6 C=0 XS ; ENSURE LESS
3 ?C=0 S=K ; THAN 16
E TC-11 ERCLB ; BUFFER CANNOT EXIST IF X>15
F A=K MS ; SET UP ID

80 NC XQ GOSUB ; LOCATE BUFFER AND RETURN
2 DEF MLOCB ; WITH HEADER REGISTERS SELECTED
3 JNC+02 DLBC ; FOUND SO CLEAR ITS MSNIBLE FOR DELETION
4 JNC+17 ERCLB ; GIVE ERROR AS IT'S NOT FOUND

5 DLBC: C=0 MS ; CLEAR MSNIBLE
6 WRITE DATA ; WRITE IT BACK
7 NC XQ ENCPX ; ENABLE CHIP 0X

XDB9A NC GO PKIAS ; PACK ID TO ASW AREA

HEADER FOR DEVELOPMENT FUNCTIONS.

DEPENDENCIES: NONE

ROUTINES USED: NONE

INPUT: NONE

OUTPUT: NONE

X(08) 0A3 S:
6 N
F E
4 V
2 U
1 D
0

X(9S) HEADER2: RTN ; TOTALLY NOTHINGNESS!
SUBROUTINE - USER FLAG EDITOR.

DEPENDENCIES: OUTHEX GETHEX ADDING
FINDF MENZ DELDIG
ROUTINES USED: CLLCDE LEFTS ANNT+4
MESSL MSG1GS
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 FLAG-d:
c=0 ALL ; M holds information for
L: c=ch+1 MS ; OUTHEX - start off with
L: c=ch+1 MS ; two underscores
L: M = C

FLAGS:
NC XQ CLLCDE ; Clear & enable LCD
NC XQ MESSL ; Push message into right

ALPHA::
FLAG-d:
gosub OUTHEX ; Output flag number and/or underscores
C=M ; Set number of digits displayed
PT=0 ;
C=C-1 PT ; Subtract two
C=C-1 PT
?CF#? PT ; If result ≠ 0 then don't indicate flag
JC+15 FDSR ; SET/CLR
Gosub FINDF ; Find the flag bit we're looking at
NC XQ LENLCD ; Turn on the display ready for SET/CLR
?B#? ALL ; If any bit is set then the flag
JC+8B FSET ; Is set so jump off and say so
NC XQ MESSL ; otherwise indicate flag is clear

ALPHA::
NC+97 FDSR ; Indicate flag is set

FSET:
NC XQ MESSL ; Indicate flag is set

FD3:
NC XQ LEFTS ; Enable
NC XQ LENLCD ; and left justify display

FLOOP:
gosub GETHEX ; Get hexadecimal keystrokes
A=C S&K ; Store in A[X][Y] as well as C&C
PF5 2 ; Set carry if special key pressed
JC+17 FNORM ; If not set then just an ordinary key
?E#? S&K ; If delete key pressed then jump
JC+31 FDEL ; Its delete routine
LDI 30#2 RS ; If &s length pressed the toggle
?A#L S&K ; Instant flag if there is one
JC+2E FRS ;
C=C-1 S&K
?#C S&K ; If not on pressed then set
JC-0E FLOOP ; another key
C=M ; If on pressed then check for
PT=0 ; final printing of the flag status
C=C-1 PT ; and only print it if these are
C=C-1 PT ; two digits on the screen
?C#? PT ; If there are any then print
JC+05 WAYOUT ; If only one then don't bother
; Print what is on the screen
SF & MSG105
NC XR MS4105
CF 5

; Jump back into DEBUG menu
WAYOUT: GOTO MENU12

; If a normal key is pressed then
FNORM: LDI φ0A
?A<5 S$X
JNC+1F FLO0P
C=M
PT=0
C=C-1 PT
?C#0 PT
JCL+8E FISTD
PT=1
2<digit input: if first
C=M
digit is less than 5 then
A=C ALL
φ.9 are valid as second
LDI φ5φ
they press so go to φ.9 bit

; Accept φ.5 only
F10UP: NDI φ06
A<5 S$X
?A<5 S&X

; anything else won't do
FLP2: JNC-3φ FLO0P
A<5 S$X

; Accept φ.9 & update display register
F09: OSUBV ADDDV5
FCON: SOTO FLAGS

; Go back and update display itself
FDEC: JNC+12 FDL

; If not reached so this one does
FRS: C=M
PT=0
C=C-1 PT
C=C-1 PT
?C#0 PT
JCL-8E FLP2

; If not ignore 8< and get another key
FINDW: A<5 ALL
?A<5 ALL
JNC+φ3
C=A-C ALL
JNC+φ2
C=A+C ALL

; else subtract paddle (i.e. clear bit)
EDL: C=M
PT=0
?C#0 PT
JNC-1D FLP2

; else delete the last digit
OSUBV DELDIS
JNC-1C FCON

; and re-join the loop
FISTD: C=M
PT=0
C=C-1 PT
C=C-1 PT
?C#0 PT
JCL+φ5 FIST2

; else clear all display
SF 8

; so don't print 'FLAGS'...
NC XR MS4105
CF 5
FAST2:  
c = 0  ALL ; Then clear the last flags data
=c+1  MS ; and allow input of 0..5 in
=c+1  MS ; the same way as for a second
M=c  ; digit following a 5 first digit

x E49

JNC=34  R1 DUP
RETURN FROM BREAKPOINT - CE9, RBKPT

DEPENDENCIES: MENU FIND9 ERRNB OUTHEX MLOCB KEYDIB
ROUTINES USED: MESSL MSG105 ENLCD

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

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

RETA CE9:
PFSET B
JCTF+14 RBKPT
GOTO MENU

RBKPT:
SEFDEC
A=0 MS
R=A-1 MS
SETHEX
GOSUB MLOCB
JNC+14 RBKPT1
GOTO ERRNB

RBKPT1:
RCR 11
POP
C=C-1 M
C=C-1 M
RCR 4
POP
RCR 13
WRITE DATA
LDE @B6
C=C+A S&X
RAM SLCT
READ DATA
RCR 5
POP
RCR 9
WRITE DATA

DETFK:
NC XQ CLLCDE
NC XQ MESSL

ALPHA:
STK2+U
GOSUB FIND9
LDE @B6
C=C+A S&X
RAM SLCT
READ DATA
RCR 7
R=Φ
LDO R 4
M=C

; Print 'No Buffer' error if not found
; Put buffer header ready for CKPT address
; Pop breakpoint address off stack
; Subtract 1 to give correct word
; Rotate header for STK 1
; Pop STK 1 off stack
; Put buffer header
; Write new header containing RBKPT and STK 1
; Get top buffer register ready for STK 2
; Prepare to display STK 2

; Message will read STK2+U @ way
; Find buffer with STK 2
; Prepare value for output with
; 4 digits
; Φ underscores
NC XQ ENLCD ; Enable LCD for output
GOSUB OUTHEX ; Output address
NC XQ LEFTJ ; Left justify the screen
SETF B
NC XQ MSG105 ; Print full message on TRACE printer
CRLF 5
GOSUB KEYDBU ; Wait for a key to be pressed
DSTK1: NC XQ CLELDE ; Save in DSTK2 but for STK1
ALPHA: STK1=0
GOSUB FIND9
RCR 3
R=0
LD@ R 4
LD@ R 0
M=C
NC XQ ENLCD
GOSUB OUTHEX
NC XQ LEFTJ
SETF B
NC XQ MSG105
CRLF 5
GOSUB KEYDBU ; Wait for a key to be pressed
LDE @C3 ; If delete then return to display
?A# C SXK ; of STK2
JC+84 DBKPT ; Else program for display of BKPT
GOTO DSTK2
DBKPT: NC XQ CLELDE ; Display BKPT (same as DSTK2)
ALPHA: BKPT=0
GOSUB FIND9
RCR 13
R=0
LD@ R 4
LD@ R 0
M=C
NC XQ ENLCD
GOSUB OUTHEX
NC XQ LEFTJ
SETF B
NC XQ MSG105
CRLF 5
GOSUB KEYDBU ; Wait for a key to be pressed
LDE @C3 ; If delete then return to display
?A# C SXK ; of STK1
JC+84 DGEND ; Else return to main menu
GOTO DSTK1
DGEND: GOTO MENU
## Display Breakpoint Number - DBN

**Dependencies:** None
**Routines Used:** CLCCDE, MESSL

**Input:**
- BKPT number in A[ES]X value 00 - 99 inclusive

**Output:**
- Display right justified showing 'BKPTn.m@l' & Enabled

<table>
<thead>
<tr>
<th>xEE4</th>
<th>DBN</th>
</tr>
</thead>
<tbody>
<tr>
<td>NC</td>
<td>XQ</td>
</tr>
<tr>
<td>NC</td>
<td>XQ</td>
</tr>
</tbody>
</table>

**ALPHA:**
- BKPT
- A > C $\delta X$
- A = C $\delta X$
- R = 2
- LDOR $\phi$
- LDOR $3$
- WRABCIR
- NC   | XQ  |
|       | MESSL |

**ALPHA:**
- m@l

<table>
<thead>
<tr>
<th>xEFE</th>
<th>RTN</th>
</tr>
</thead>
</table>
### BREAKPOINT TABLES - BAT, Bn, RBn

**Dependencies:** none

**Routines Used:** none

**Use:**
- BAT - is used to return the table address
- B0-B9 - used to hold high/low breakpoint address pairs
- RB0-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.

<table>
<thead>
<tr>
<th>xEFF</th>
<th>BAT:</th>
<th>RTN</th>
</tr>
</thead>
<tbody>
<tr>
<td>xF00</td>
<td>B0:</td>
<td>high</td>
</tr>
<tr>
<td></td>
<td></td>
<td>low</td>
</tr>
<tr>
<td>xF02</td>
<td>B1:</td>
<td>high</td>
</tr>
<tr>
<td></td>
<td></td>
<td>low</td>
</tr>
<tr>
<td>xF04</td>
<td>B2:</td>
<td>high</td>
</tr>
<tr>
<td></td>
<td></td>
<td>low</td>
</tr>
<tr>
<td>xF06</td>
<td>B3:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xF08</td>
<td>B4:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xF0A</td>
<td>B5:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xF0C</td>
<td>B6:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xF0E</td>
<td>B7:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xF10</td>
<td>B8:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xF12</td>
<td>B9:</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>xF14</td>
<td>RB0:</td>
<td>ovr1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ovr2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>xF18</td>
<td>RB1:</td>
<td>ovr1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ovr2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>09</td>
<td></td>
<td></td>
</tr>
<tr>
<td>xF1C</td>
<td>RB2:</td>
<td>ovr1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ovr2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>xF20</td>
<td>RB3:</td>
<td>NC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>06</td>
<td></td>
<td></td>
</tr>
<tr>
<td>xF28</td>
<td>RB9:</td>
<td>ovr1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>ovr2</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>09</td>
<td></td>
<td></td>
</tr>
<tr>
<td>xF38</td>
<td>RB10:</td>
<td>NC</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NC</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>06</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes:
- Called using GOSUB to return address of table
- High/low byte pair giving address of breakpoint
- As above for breakpoint 1 etc.

Note: If high byte of a breakpoint in zero, BKED assumes the breakpoint is invalid or does not exist and addresses P0x are in the operating system.

For words from where the breakpoint instruction is written
Jump back to word after breakpoint instruction to continue execution (for each breakpoint)

Note: It is assumed that ovr-2 instructions do not set the carry.
Any otherwise the NC go will be ignored. Also it is assumed that breakpoints will not be placed over 3 word jumps and go to subroutines, or message output, etc.
DISPLAY BREAKPOINT STATUS - DBS

DEPENDENCIES: NONE
Routines Used: CLLCDE
PCOTC
MESSL
LEFTJ

INPUT: BREAKPOINT TABLE 2F - 2F9 CORRECTLY FORMATTED
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 XR CLLCDE ; Clear and enable display
 NC XR PCOTC ; Get address of 2F9 word 0
 R=5 ; i.e. XF2
 LD@R F ; where x in the current page
 LD@R 1
 LD@R 2

DBSL:
 MOCOM ; Get high address word
 ?C#0 S&X ; If word is not zero then set carry
 JNC+1 DBSN ; ZERO: Try next breakpoint
 PT=0
 G=0 ; Save high addr in G
 C=PT+1 M ; Get low address
 RDRM
 PT=2
 C=G ; Combine high and low
 RCR 11
 RDRM ; Get word at BKPTn
 A=PT S&X ; Put word in A [S&X]
 LDE 3DD ; If word is not first part of NC XR BKPT
 ?A#C S&X ; Then set the carry flag
 JC+2 DIS ; and indicate BKPTS are disabled
 C=PT+1 M ; Else get next word and
 RDRM ; shift to give PAGE/SUBPAGE in
 C=PT+1 S&X
 C=PT+1 S&X
 A=PT S&X
 NC XR PCOTC ; Get high page in
 RCR 4 ; c[2]
 PT=1
 LC 0 ; Subpage 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+24 EN ; Else enabled

DBSN:
 PT=Q ; Point to next word
 PT=4
 PT=3
 C=PT S&Q A ; If carry xEFF -> xEFF then done - no bkpt
 JC+03 BOFF ; else try next entry
BOFF:  NC  XQ MESSL  ; Indicate 'NOBKPTS'
ALPHA:  NO=0BKPTS
JNC+17 DBSE
DIS:  NC  XQ MESSL  ; Indicate 'BKPTS = OFF'
ALPHA:  BKPTS=OFF
JNC+16 DBSE
E.N.:  NC  XQ MESSL
ALPHA:  BKPTS=ON
DBSE:  C->B S8X  ; Put last character in B[S8X]
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, REG ST
         FINDBN - LEAVES M & N UNCHANGED, A CORRUPT
                    [ADDR] CONTAINS N ADDRESS

xF8E  DAFN:  C=N  ; Save N in ST
            ST=C
            GOSUB DABKPT  ; Disables Breakpoints
            C=ST
            N=C  ; Restore N

xF95  FINDBN:  GOSUB BAT  ; Get 1n table
            C=C+1 M  ; std addrel
            A=C M  ; into A [ADDR]
            C=N  ; Get breakpoint number from N
            RCR 1  ; Mask off unecessary digits
            C=0 S8X
            RCR 10
            C=C+1 M  ; Double breakpoint number and
            C=C+A M  ; Add to table address

xF90  RTN
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-1B.

**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
SETDEC
A=0 MS
A=A-1 MS
SETHEX
GOSUB MLDCB
JNC+02 PONT
JNC+06 PEXIT
PONT:
SETDEC
C=0 MS
C=C-1 MS
SETHEX
WRITE DATA
PEXIT:
C=0 S&X
RAM SECT
C=N
JNC+ ONMSG
PWRDN:
C=ST
N=C
READ 14(d)
RCR 6
ST=C
SETF 6 ; SET error ignore flag
C=ST
RCR 8
WRITE 14(d)
C=0 M
LD@R 7 ; GET XROM number of ROM in Page 7
FETCH
```
A=C S6X
LNE 01C ; Check that the ROM is HP-IL ROM
?A%C S6X
JC+05 DFD6 ; If not then don't call it
NC XQ 77CD ; Power Up All Loop Devices
NC XQ 77CA ; Power Down All Loop Devices
C=N ; Restore Poll Data
ST= C
JNC+1A EXIT ; Exit

LINK: JNC-2B PRSU9 ; Link back from ON to PRSU9
ONMSG: DSPTOG ; Tim 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 ; Tim on display to show message
EXIT: NC GO ROMCK10 ; Exit back to operating system poll routine

XFFF PSE: NOP
5 RUN: NOP
6 WPNK: NOP
7 OFF JNC-36 PWRDN
8 I/O: NOP
9 ON: JNC-2A LINK
A COLDS: NOP
B TRAILER: 'B'
C 'L'
D 'P'
E 'M'
XFFF CHKSUM: checksum word