```
C100    DATA >80C0,>C0C0,>C0C0,>C0E0
C101    DATA >FF7F,>0301,>0000,>0000
C102    DATA >0103,>0303,>0303,>0301
C103    DATA >0000,>0000,>0000,>7EFF
C104    DATA >F0F8,>FCFE,>FF7F,>3F1F
C105    DATA >0000,>0000,>0080,>C0E0
C106    DATA >0F07,>0301,>0000,>0000
C107    DATA >F0F8,>FCFE,>FF7E,>3C18
C108    DATA >0000,>0000,>0103,>7FFF
C109    DATA >E0C0,>C0C0,>C0C0,>C080
C110    DATA >0F1F,>3F7F,>FEFC,>F8F0
C111    DATA >E0C0,>8000,>0000,>0000
C112    DATA >0000,>0000,>0001,>0307
C113    DATA >0018,>3C7E,>FEFC,>F8F0
C114    DATA >FF7E,>0000,>0000,>0000
C115    DATA >80FF,>FFFF,>FFFF,>FF80
C116    DATA >00FF,>FFFF,>FFFF,>FF00
C117    DATA >FFFF,>FFFF,>FFFF,>FFFF
*
DNDAT   BYTE 103,102,117,100,101,104,105,106,107
        EVEN

MDDAT   BYTE 103,102,117,115,116,114
        EVEN

UPDAT   BYTE 112,113,108,110,111,102,117,109,114
        EVEN

UPADD   DATA -31,-30,0,1,2,31,32,33,64,>FFFF

MDADD   DATA 0,31,32,33,34,64,>FFFF

DNADD   DATA 0,31,32,33,64,65,66,97,98,>FFFF

SCREEN  DATA 34,28
        TEXT '***************************** '
        DATA 66,2
        TEXT '* '
        DATA 92,2
        TEXT '* '
        DATA 98,28
        TEXT '* GRAM KRACKER PRODUCTION * '
        DATA 130,2
        TEXT '* '
        DATA 156,2
        TEXT '* '
        DATA 162,28
        TEXT '***************************** '
        DATA 354,28
        TEXT '1 - INITIAL TEST (PHASE ONE)'
        DATA 418,28
        TEXT '2 - BATTERY TEST (PHASE TWO)'
        DATA 482,12
        TEXT '3 - BURN IN '
        DATA 0

DONE    TEXT 'TEST FINISHED '
HOLD    TEXT 'HOLD SPACE BAR TO ABORT '
BUSY    TEXT '          BUSY....          '
```

This is at least a partial printout of the GK production I wrote for Craig & Sue.

√ diskette

Doug

```
H6K     EQU  >6000
HDSKP   EQU  H6K+6

HD      DATA >AA01,0,0,>6010,0,0,0,0

        DATA HDA-HD+H6K,>0020
        BYTE 17
        TEXT 'OPTIONAL GRAMS OK'

HDA     DATA HD1-HD+H6K,>0020
        BYTE 9
        TEXT 'RAMS   OK'

HD1     DATA HD2-HD+H6K,>0020
        BYTE 9
        TEXT 'PROM   OK'

HD2     DATA HD3-HD+H6K,>0020
        BYTE 9
        TEXT 'GRAM 7 OK'

HD3     DATA HD4-HD+H6K,>0020
        BYTE 9
        TEXT 'GRAM 6 OK'

HD4     DATA HD5-HD+H6K,>0020
        BYTE 9
        TEXT 'GRAM 5 OK'

HD5     DATA HD6-HD+H6K,>0020
        BYTE 9
        TEXT 'GRAM 4 OK'

HD6     DATA >0000,>0020
        BYTE 9
        TEXT 'GRAM 3 OK'
HDEND   EQU  $
*
*
*
GRMESS  TEXT 'GRAM FAILURE'
RM2     TEXT 'RAM  FAILURE'
WTPRT   TEXT 'WRITE PROTECT FAILURE'
BNKSWP  TEXT 'BANK SWAPPING FAILURE'
PRM     TEXT 'PROM FAILURE'
GKOFF   TEXT 'GK OFF FAILURE'
BATT    TEXT 'BATTERY BACKUP FAILURE'
```

```
*
* Equate table
*
WSR     EQU  >8300
WSR1    EQU  >8320
WSR2    EQU  >8340
DSRWS   EQU  >8380
GPLWS   EQU  >83E0
R3LB    EQU  GPLWS+7
R12LB   EQU  GPLWS+25
STATUS  EQU  >837C
KEYBRD  EQU  >8375
VDPWA   EQU  >8C02
VDPWD   EQU  >8C00
VDPRD   EQU  >8800
YFLAG   EQU  >837A
XFLAG   EQU  >837E

FRSTSW  EQU  163
OFFSET  EQU  6
```

```
*
* Grom chech routine
*  ENTER: R0 start address
*         R1 finish address
*         R2 MSB compare byte
*
*  LEAVE: Normal return on no error
*         Skips a word on error return
*
GCOMP   DATA WSR1,GCOMP1

GCOMP1  MOV  R13,R3             Fetch caller's wrkspc
        MOV  *R3+,R0            Fetch start address
        BL   @GWADD             Setup GROM address

        MOV  *R3+,R1            Fetch finish address
        MOV  *R3,R2             Fetch fill byte
        MOV  @>83FA,R3          Fetch GROM base
        DEC  R0                 Adjust R0

        CI   R2,>AA00           Do we have the ROM flag?
        JEQ  GCOMP5             YES! Compare GRAM with ROM

GCOMP2  CB   *R3,R2             Fill byte compare?
        JNE  GCOMP3             NO! Error
        INC  R0                 Increment start address
        C    R0,R1              Done?
        JL   GCOMP2             NO!
        JMP  GCOMP4             Don't skip a word on return

GCOMP3  INCT R14                Skip a word
GCOMP4  RTWP                    Return
*
* ROM compare
*  8k maximum compare
*
GCOMP5  CLR  R2
GCOMP6  CB   *R3,*R2+           ROM byte compare?
        JNE  GCOMP7             NO! Error
        INC  R0                 Increment start address
        C    R0,R1              Done?
        JL   GCOMP6             NO!
        JMP  GCOMP8             Don't skip a word on return

GCOMP7  INCT R14                Skip a word
GCOMP8  RTWP                    Return
```

```
*
* Grom fill routine
*  ENTER: R0 with start GROM address
*         R1 with finish GROM address
*         R2 MSB with fill value
*
GFILL   DATA WSR1,GFILL1        Routine vectors

GFILL1  MOV  R13,R3             Fetch caller's wrkspc
        MOV  *R3+,R0            Fetch start address
        BL   @GWADD             Setup GROM address

        MOV  *R3+,R1            Fetch finish address
        MOV  *R3,R2             Fetch fill byte
        MOV  @>83FA,R3          Fetch GROM base
        DEC  R0                 Adjust R0

        CI   R2,>AA00           Do we have a fill ROM flag?
        JEQ  GFILL3             YES! Fill GRAM with ROM code

GFILL2  MOVB R2,@>400(R3)       Move fill byte into GRAM
        INC  R0                 Increment start address
        C    R0,R1              Are we done?
        JL   GFILL2             NO!

        RTWP                    Return to caller
*
* ROM fill
*  8k maximum fill
*
GFILL3  CLR  R2
GFILL4  MOVB *R2+,@>400(R3)     Move ROM byte into GRAM
        INC  R0                 Increment start address
        C    R0,R1              Are we done?
        JL   GFILL4             NO!

        RTWP                    Return to caller
```

```
*
* Print routine
*  ENTER: String length byte
*         Text string following BLWP
*
PRINT   DATA WSR1,PRINT1

PAB     EQU  >F80
PABBUF  EQU  >1000
PNTR    EQU  >8356

PABDAT  DATA >0012,PABBUF,>5050,>0000,>0004
NAME    TEXT 'PIO.'
WRITE   DATA >0312
CLOSE   DATA >0112

PRINT1  LI   R0,PAB             Load the PAB
        LI   R1,PABDAT             .
        LI   R2,14                 .
        BL   @VMBW                 .

        LI   R3,PAB+9           Open the printer
        MOV  R3,@PNTR              .
        BLWP @DSRLNK               .

        MOV  @YFLAG,@YFLAG      Is this the first line of print?
        JEQ  PRNTHD             YES! Print header first

PRNRET  MOVB *R14+,R4           Get string length

        LI   R0,PAB+5           Update the PAB
        MOVB R4,R1               with new length
        BL   @VSBW                 .

        LI   R0,PABBUF          Set up VDP buffer address
        BL   @VWDADD               .
        SRL  R4,8               Right justify counter

PRINT2  MOVB *R14+,@VDPWD       Move string into VDP buffer
        DEC  R4                 Done?
        JNE  PRINT2             NO!

        LI   R0,PAB             Write string out to printer
        LI   R1,WRITE              .
        LI   R2,2                  .
        BL   @VMBW                 .
        MOV  R3,@PNTR              .
        BLWP @DSRLNK               .

        LI   R0,PAB             Close printer
        LI   R1,CLOSE              .
        LI   R2,2                  .
        BL   @VMBW                 .
        MOV  R3,@PNTR              .
        BLWP @DSRLNK               .

        BLWP @CHKKEY            Do a dummy keyscan in case or FCTN 4

        RTWP
```

```
HEADER BYTE >0E                  Set up double width elongated
       TEXT '    Gram Kracker Production Report'
HEADES BYTE >0E
       TEXT '    ------------------------------------'
       BYTE >0F    >14 THIS IS FOR PANASONIC
       BYTE >0A,>0A,>0A,>0A

PRNTHD SETO @YFLAG

       LI   R0,PABBUF
       LI   R1,HEADER
       LI   R2,35
       BL   @VMBW

       LI   R0,PAB+5             Update the PAB
       LI   R1,35*>100            with new length
       BL   @VSBW                    .

       LI   R0,PAB               Write string out to printer
       LI   R1,WRITE                 .
       LI   R2,2                     .
       BL   @VMBW                    .

       MOV  R3,@PNTR                 .
       BLWP @DSRLNK                  .

       LI   R0,PABBUF
       LI   R1,HEADES
       LI   R2,40
       BL   @VMBW

       LI   R0,PAB+5             Update the PAB
       LI   R1,40*>100            with new length
       BL   @VSBW                    .

       MOV  R3,@PNTR                 .
       BLWP @DSRLNK                  .

       B    @PRNRET
```

```
*
* Rom check routine
*  ENTER: R2 MSB compare byte
*
*  LEAVE: Normal return on no error
*         Skips a word on error return
*
RCOMP   DATA WSR1,RCOMP1

RCOMP1  LI   R0,>6000          Set start address
        MOV  R13,R3            Fetch caller's wrkspc
        C    *R3+,*R3+         Increment R3 by four
        MOVB *R3,R2            Fetch fill byte

        CI   R2,>AA00          Do we have a ROM compare flag?
        JEQ  RCOMP5            YES! Compare RAM with ROM

RCOMP2  CB   *R0+,R2           Fill byte compare?
        JNE  RCOMP3            NO! Error
        CI   R0,>7FFF          Done?
        JLE  RCOMP2            NO!
        JMP  RCOMP4            Don't skip a word on return

RCOMP3  INCT R14               Skip a word
RCOMP4  RTWP                   Return
*
* RAM compare with ROM section
*  8k compare maximum
*
RCOMP5  CLR  R2
RCOMP6  CB   *R0+,*R2+         Fill byte compare?
        JNE  RCOMP7            NO! Error
        CI   R0,>7FFF          Done?
        JLE  RCOMP6            NO!
        JMP  RCOMP8            Don't skip a word on return

RCOMP7  INCT R14               Skip a word
RCOMP8  RTWP                   Return
```

```
        DEF  START
        DEF  SFIRST,SLOAD,SLAST
        AORG >E100

SFIRST  EQU  $
SLOAD   EQU  $
TOP     EQU  355

        COPY "WDS1.GKTST.EQUATES"
        COPY "WDS1.GKTST.DATA"
        COPY "WDS1.GKTST.UTIL"
        COPY "WDS1.GKTST.SETSW"
        COPY "WDS1.GKTST.GFILL"
        COPY "WDS1.GKTST.RFILL"
        COPY "WDS1.GKTST.GCOMP"
        COPY "WDS1.GKTST.RCOMP"
        COPY "WDS1.GKTST.PRINT"
        COPY "WDS1.GKTST.WAIT"

START   LWPI WSR                Load workspace
STARTA ~~TEST~~ BL   @CLRSCN    Clear screen

        LI   R0,>0380           Load color table
        LI   R1,>F400              .
        LI   R2,>1F                .
        BL   @RPTBLK               .

        LI   R0,>87F4           Load background color
        BL   @VWDADD               .

        BL   @FORMAT            Put production screen up
        DATA SCREEN                .

        LI   R0,>B20            Load character table address
        LI   R1,C100             and load switch chars
        LI   R2,144                .
        BL   @VMBW                 .

        CLR  @YFLAG             Clear page header flag
        CLR  @XFLAG             Clear expanded GK flag
TEST1   BLWP @CHKKEY            Check for a selection
        MOVB R0,R0                 .
        JEQ  TEST1                 .
        CI   R1,'1'*>100           .
        JEQ  SELON                 .
        CI   R1,'2'*>100           .
        JEQ  SELVEC                .
        CI   R1,'3'*>100           .
        JEQ  BURVEC                .
        CI   R1,>0500              .
        JNE  TEST1              Loop if none of above
        BLWP @0                 Reset

BURVEC  B    @BURNIN
SELVEC  B    @SELTW
*
* Begin main test
*
SELON   BL   @CLRSCN            Clear screen
        BLWP @SETSW             Set switches
```

```
        TEXT 'MUUUD '                .
        BLWP @WAIT                   Wait for user response
*
* Determine if we are testing an expanded GK
*
        CLR  R0                      GRAM address of zero
        BL   @GWADD                  Set up GRAM address
        BL   @GRDAT                  Grab first byte
        CB   R1,@HAA                 Is it an 'AA'?
        JEQ  SELON1                  YES!
        SETO @XFLAG                  NO! Set flag

SELON1  LI   R0,>2000                Set GRAM address to TI BASIC
        BL   @GWADD                  Set up address
        BL   @GRDAT                  Get first byte
        CB   R1,@HAA                 Is it an 'AA'?
        JEQ  SELON2                  YES!
        SETO @XFLAG                  NO! Set expanded GK flag
*
* Verify GRAMS
*
SELON2  LI   R6,2                    Initialize R6 as loop counter
        SETO R7                      Initialize R7
LOOP    CLR  R0                      Set start address
        SETO R1                      Set finish address
        INV  R7                      Invert R7 fill data
        MOV  R7,R2                   Set fill byte
        BLWP @GFILL                  Fill all of GRAM

        MOV  @XFLAG,@XFLAG           Expanded GK?
        JEQ  SEL3                    NO!

        LI   R1,>1FFF                Set finish address (Start and Fill already set)
        BLWP @GCOMP                  Check GRAM 0
        JMP  SEL1                    Jump on no error
        LI   R0,TOP                  PUT ERROR
        LI   R1,GRMESS                MESSAGE
        LI   R2,12                    ON
        BL   @VMBW                    SCREEN
        LI   R0,TOP+13                  .
        LI   R1,'0'*>100                .
        BL   @VSBW
*       BLWP @PRINT                  Report an error
*       BYTE 21
*       TEXT '  Gram 0 failed test'
*       BYTE 10

SEL1    LI   R0,>2000                Load start address
        LI   R1,>3FFF                Load finish test
        MOV  R7,R2                   Load fill byte
        BLWP @GCOMP                  Do compare
        JMP  SEL2                    Jump if good
        LI   R0,TOP                  PUT ERROR
        LI   R1,GRMESS                MESSAGE
        LI   R2,12                    ON
        BL   @VMBW                    SCREEN
        LI   R0,TOP+15                  .
        LI   R1,'1'*>100                .
        BL   @VSBW
*       BLWP @PRINT                  Report error to printer
```

```
*       BYTE 21
*       TEXT '  Gram 1 failed test'
*       BYTE 10

SEL2    LI   R0,>4000          Load start address
        LI   R1,>5FFF          Load finish test
        MOV  R7,R2             Load fill byte
        BLWP @GCOMP            Do compare
        JMP  SEL3              Jump if good
        LI   R0,TOP            PUT ERROR
        LI   R1,GRMESS          MESSAGE
        LI   R2,12              ON
        BL   @VMBW              SCREEN
        LI   R0,TOP+17            .
        LI   R1,'2'*>100          .
        BL   @VSBW
*       BLWP @PRINT            Report error to printer
*       BYTE 21
*       TEXT '  Gram 2 failed test'
*       BYTE 10

SEL3    LI   R0,>6000          Load start address
        LI   R1,>7FFF          Load finish test
        MOV  R7,R2             Load fill byte
        BLWP @GCOMP            Do compare
        JMP  SEL4              Jump if good
        LI   R0,TOP            PUT ERROR
        LI   R1,GRMESS          MESSAGE
        LI   R2,12              ON
        BL   @VMBW              SCREEN
        LI   R0,TOP+19            .
        LI   R1,'3'*>100          .
        BL   @VSBW
*       BLWP @PRINT            Report error to printer
*       BYTE 21
*       TEXT '  Gram 3 failed test'
*       BYTE 10

SEL4    LI   R0,>8000          Load start address
        LI   R1,>9FFF          Load finish test
        MOV  R7,R2             Load fill byte
        BLWP @GCOMP            Do compare
        JMP  SEL5              Jump if good
        LI   R0,TOP            PUT ERROR
        LI   R1,GRMESS          MESSAGE
        LI   R2,12              ON
        BL   @VMBW              SCREEN
        LI   R0,TOP+21            .
        LI   R1,'4'*>100          .
        BL   @VSBW
*       BLWP @PRINT            Report error to printer
*       BYTE 21
*       TEXT '  Gram 4 failed test'
*       BYTE 10

SEL5    LI   R0,>A000          Load start address
        LI   R1,>BFFF          Load finish test
        MOV  R7,R2             Load fill byte
        BLWP @GCOMP            Do compare
        JMP  SEL6              Jump if good
```

```
        LI   R0,TOP             PUT ERROR
        LI   R1,GRMESS           MESSAGE
        LI   R2,12               ON
        BL   @VMBW               SCREEN
        LI   R0,TOP+23             .
        LI   R1,'5'*>100           .
        BL   @VSBW
*       BLWP @PRINT             Report error to printer
*       BYTE 21
*       TEXT '  Gram 5 failed test'
*       BYTE 10

SEL6    LI   R0,>C000           Load start address
        LI   R1,>DFFF           Load finish test
        MOV  R7,R2              Load fill byte
        BLWP @GCOMP             Do compare
        JMP  SEL7               Jump if good
        LI   R0,TOP             PUT ERROR
        LI   R1,GRMESS           MESSAGE
        LI   R2,12               ON
        BL   @VMBW               SCREEN
        LI   R0,TOP+25             .
        LI   R1,'6'*>100           .
        BL   @VSBW
*       BLWP @PRINT             Report error to printer
*       BYTE 21
*       TEXT '  Gram 6 failed test'
*       BYTE 10

SEL7    LI   R0,>E000           Load start address
        LI   R1,>FFFF           Load finish test
        MOV  R7,R2              Load fill byte
        BLWP @GCOMP             Do compare
        JMP  SELON3             Jump if good
        LI   R0,TOP             PUT ERROR
        LI   R1,GRMESS           MESSAGE
        LI   R2,12               ON
        BL   @VMBW               SCREEN
        LI   R0,TOP+27             .
        LI   R1,'7'*>100           .
        BL   @VSBW
*       BLWP @PRINT             Report error to printer
*       BYTE 21
*       TEXT '  Gram 7 failed test'
*       BYTE 10
*
* Check the RAMs
*
SELON3  MOV  R7,R2              Fill byte
        BLWP @RFILL             Fill RAM 2
        BLWP @RCOMP             Check RAM 2
        JMP  SEL8               No error
        LI   R0,TOP+32          PUT ERROR
        LI   R1,RM2              MESSAGE
        LI   R2,12               ON
        BL   @VMBW               SCREEN
        LI   R0,TOP+49             .
        LI   R1,'1'*>100           .
        BL   @VSBW
*       BLWP @PRINT             Print error
```

```
*       BYTE 21
*       TEXT '  Ram  1 failed test'
*       BYTE 10

SEL8    BLWP @SETSW             Set WP switch to bank 1
        TEXT '   D  '             .
        BLWP @WAIT              Wait for user

        MOV  R7,R2              Fill byte
        BLWP @RFILL             Fill RAM 1
        BLWP @RCOMP             Check RAM 1
        JMP  ENDLP              No error
        LI   R0,TOP+32          PUT ERROR
        LI   R1,RM2              MESSAGE
        LI   R2,12               ON
        BL   @VMBW               SCREEN
        LI   R0,TOP+47             .
        LI   R1,'2'*>100           .
        BL   @VSBW
*       BLWP @PRINT             Print error
*       BYTE 21
*       TEXT '  Ram  2 failed test'
*       BYTE 10

ENDLP   BLWP @SETSW             Set switche
        TEXT '   U  '             .
        BLWP @WAIT              Wait for user

        DEC  R6
        JEQ  RELON2

        B    @LOOP
*
* Verify GRAMS with a ROM fill
*
RELON2
        MOV  @XFLAG,@XFLAG      Expanded GK?
        JEQ  REL3               NO!

        CLR  R0                 Set start address
        LI   R1,>1FFF           Set finish address (Start and Fill already set)
        LI   R2,>AA00           Fill flag for ROM fill
        BLWP @GFILL             Fill all of GRAM
        BLWP @GCOMP             Check GRAM 0
        JMP  REL1               Jump on no error
        LI   R0,TOP             PUT ERROR
        LI   R1,GRMESS           MESSAGE
        LI   R2,12               ON
        BL   @VMBW               SCREEN
        LI   R0,TOP+13             .
        LI   R1,'0'*>100           .
        BL   @VSBW
*       BLWP @PRINT             Report an error
*       BYTE 21
*       TEXT '  Gram 0 failed test'
*       BYTE 10

REL1    LI   R0,>2000           Load start address
        LI   R1,>3FFF           Load finish test
        LI   R2,>AA00              Load fill byte
```

```
        BLWP @GFILL             Fill all of GRAM
        BLWP @GCOMP             Do compare
        JMP  REL2               Jump if good
        LI   R0,TOP             PUT ERROR
        LI   R1,GRMESS           MESSAGE
        LI   R2,12               ON
        BL   @VMBW               SCREEN
        LI   R0,TOP+15             .
        LI   R1,'1'*>100           .
        BL   @VSBW
*       BLWP @PRINT             Report error to printer
*       BYTE 21
*       TEXT '  Gram 1 failed test'
*       BYTE 10

REL2    LI   R0,>4000           Load start address
        LI   R1,>5FFF           Load finish test
        LI   R2,>AA00              Load fill byte
        BLWP @GFILL             Fill all of GRAM
        BLWP @GCOMP             Do compare
        JMP  REL3               Jump if good
        LI   R0,TOP             PUT ERROR
        LI   R1,GRMESS           MESSAGE
        LI   R2,12               ON
        BL   @VMBW               SCREEN
        LI   R0,TOP+17             .
        LI   R1,'2'*>100           .
        BL   @VSBW
*       BLWP @PRINT             Report error to printer
*       BYTE 21
*       TEXT '  Gram 2 failed test'
*       BYTE 10

REL3    LI   R0,>6000           Load start address
        LI   R1,>7FFF           Load finish test
        LI   R2,>AA00              Load fill byte
        BLWP @GFILL             Fill all of GRAM
        BLWP @GCOMP             Do compare
        JMP  REL4               Jump if good
        LI   R0,TOP             PUT ERROR
        LI   R1,GRMESS           MESSAGE
        LI   R2,12               ON
        BL   @VMBW               SCREEN
        LI   R0,TOP+19             .
        LI   R1,'3'*>100           .
        BL   @VSBW
*       BLWP @PRINT             Report error to printer
*       BYTE 21
*       TEXT '  Gram 3 failed test'
*       BYTE 10

REL4    LI   R0,>8000           Load start address
        LI   R1,>9FFF           Load finish test
        LI   R2,>AA00              Load fill byte
        BLWP @GFILL             Fill all of GRAM
        BLWP @GCOMP             Do compare
        JMP  REL5               Jump if good
        LI   R0,TOP             PUT ERROR
        LI   R1,GRMESS           MESSAGE
        LI   R2,12               ON
```

```
        BL   @VMBW                SCREEN
        LI   R0,TOP+21              .
        LI   R1,'4'*>100            .
        BL   @VSBW
*       BLWP @PRINT               Report error to printer
*       BYTE 21
*       TEXT '  Gram 4 failed test'
*       BYTE 10

REL5    LI   R0,>A000             Load start address
        LI   R1,>BFFF             Load finish test
        LI   R2,>AA00                Load fill byte
        BLWP @GFILL               Fill all of GRAM
        BLWP @GCOMP               Do compare
        JMP  REL6                 Jump if good
        LI   R0,TOP               PUT ERROR
        LI   R1,GRMESS             MESSAGE
        LI   R2,12                 ON
        BL   @VMBW                 SCREEN
        LI   R0,TOP+23              .
        LI   R1,'5'*>100            .
        BL   @VSBW
*       BLWP @PRINT               Report error to printer
*       BYTE 21
*       TEXT '  Gram 5 failed test'
*       BYTE 10

REL6    LI   R0,>C000             Load start address
        LI   R1,>DFFF             Load finish test
        LI   R2,>AA00                Load fill byte
        BLWP @GFILL               Fill all of GRAM
        BLWP @GCOMP               Do compare
        JMP  REL7                 Jump if good
        LI   R0,TOP               PUT ERROR
        LI   R1,GRMESS             MESSAGE
        LI   R2,12                 ON
        BL   @VMBW                 SCREEN
        LI   R0,TOP+25              .
        LI   R1,'6'*>100            .
        BL   @VSBW
*       BLWP @PRINT               Report error to printer
*       BYTE 21
*       TEXT '  Gram 6 failed test'
*       BYTE 10

REL7    LI   R0,>E000             Load start address
        LI   R1,>FFFF             Load finish test
        LI   R2,>AA00                Load fill byte
        BLWP @GFILL               Fill all of GRAM
        BLWP @GCOMP               Do compare
        JMP  RELON3               Jump if good
        LI   R0,TOP               PUT ERROR
        LI   R1,GRMESS             MESSAGE
        LI   R2,12                 ON
        BL   @VMBW                 SCREEN
        LI   R0,TOP+27              .
        LI   R1,'7'*>100            .
        BL   @VSBW
*       BLWP @PRINT               Report error to printer
*       BYTE 21
```

```
*       TEXT '  Gram 7 failed test'
*       BYTE 10
*
* Check the RAMs with a ROM fill
*
RELON3 LI    R2,>AA00              Fill byte
       BLWP @RFILL               Fill RAM 2
       BLWP @RCOMP               Check RAM 2
       JMP  REL8                 No error
       LI   R0,TOP+32            PUT ERROR
       LI   R1,RM2                MESSAGE
       LI   R2,12                 ON
       BL   @VMBW                 SCREEN
       LI   R0,TOP+49               .
       LI   R1,'1'*>100             .
       BL   @VSBW
*       BLWP @PRINT               Print error
*       BYTE 21
*       TEXT '  Ram  1 failed test'
*       BYTE 10

REL8   BLWP @SETSW               Set WP switch to bank 1
       TEXT '   D  '                .
       BLWP @WAIT                Wait for user

       LI   R2,>AA00               Fill byte
       BLWP @RFILL               Fill RAM 1
       BLWP @RCOMP               Check RAM 1
       JMP  SELON4               No error
       LI   R0,TOP+32            PUT ERROR
       LI   R1,RM2                MESSAGE
       LI   R2,12                 ON
       BL   @VMBW                 SCREEN
       LI   R0,TOP+47               .
       LI   R1,'2'*>100             .
       BL   @VSBW
*       BLWP @PRINT               Print error
*       BYTE 21
*       TEXT '  Ram  2 failed test'
*       BYTE 10
*
* Test Write Protect
*  First intitialize GK
*
SELON4 CLR  R0
       SETO R1
       LI   R2,>E500
       BLWP @GFILL

       LI   R2,>2200
       BLWP @RFILL

       BLWP @SETSW
       TEXT '   U  '
       BLWP @WAIT

       LI   R2,>1100
       BLWP @RFILL
*
*  On with the test
```

```
*
        BLWP @SETSW             Set write protect switch
        TEXT '   M  '             .
        BLWP @WAIT              Wait for user response

        MOV  @XFLAG,@XFLAG      Do we have an expanded GK
        JEQ  SELON5             NO!

        CLR  R0                 Try clearing GRAM 0
        LI   R1,>1FFF              .
        SETO R2                    .
        BLWP @GFILL                .
        BLWP @GCOMP             See if write protect failed
        JMP  SELON6             YES!
        NOP                     NO!

SELON5  LI   R0,>6000           Try clearing module space
        SETO R1                    .
        SETO R2                    .
        BLWP @GFILL                .
        BLWP @GCOMP             See if write protect failed
        JMP  SELON6             YES!
        NOP                     NO!

        BLWP @RFILL             Try clearing RAM space
        BLWP @RCOMP             See if write protect failed
        JMP  SELON6             YES!
        JMP  SELON7             NO!

SELON6  EQU  $
        LI   R0,TOP+64          PUT ERROR
        LI   R1,WTPRT            MESSAGE
        LI   R2,21               ON
        BL   @VMBW               SCREEN
*       BLWP @PRINT             Print error message
*       BYTE 25
*       TEXT '  Write Protect failure '
*       BYTE 10
*
* Check bank swapping
*
SELON7  LI   R1,>1100
        LI   R2,>2200
        LI   R7,>5FFE

LOOP1   INCT R7
        MOVB R2,*R7             Swap to bank one
        CB   *R7,R1             See if it is full of >11s
        JEQ  SEL9               YES!
        JMP  SELA               NO! Indicate error

SEL9    INCT R7                 Increment address by two
        MOVB R2,*R7             Swap to bank two
        CB   *R7,R2             See if it is full of >22s
        JEQ  ENDLP1             YES!
SELA    EQU  $
        LI   R0,TOP+96          PUT ERROR
        LI   R1,BNKSWP           MESSAGE
        LI   R2,21               ON
        BL   @VMBW               SCREEN
```

```
*       BLWP @PRINT             NO! Print error
*       BYTE  25
*       TEXT '  Bank swapping failure '
*       BYTE 10
        JMP  SEL07A             On to next test

ENDLP1  CI   R7,>7FFE
        JNE  LOOP1
*
* Check PROM
*
SEL07A  BLWP @SETSW             Page loader in
        TEXT '    U '             .
        BLWP @WAIT              Wait for user response

        LI   R0,>2000           Set up address for loader
        BL   @GWADD               .
        LI   R1,>C000           Load address of master
        MOV  @>83FA,R2          Grab GROM library base
SELON8  CB   *R2,*R1+           Compare loader with master
        JNE  SELON9             Indicate error
        DEC  R0                 Finished?
        JNE  SELON8             NO!
        JMP  SELONA             YES!

SELON9
        LI   R0,TOP+128         PUT ERROR
        LI   R1,PRM              MESSAGE
        LI   R2,12               ON
        BL   @VMBW               SCREEN
*       BLWP @PRINT             Indicate loader error
*       BYTE 15
*       TEXT '  PROM failure'
*       BYTE 10
*
* Check GK OFF switch
*
SELONA  BLWP @SETSW             Set GK OFF switch
        TEXT 'U     '             .
        BLWP @WAIT              Wait for user response

        LI   R0,>6000           See if module GRAM space is off
        SETO R1                   .
        CLR  R2                   .
        BLWP @GCOMP               .
        JMP  SEL01A             YES! Go check RAM
        NOP                     NO! Check >FFs

        SETO R2                   .
        BLWP @GCOMP               .
        JMP  SEL01A             YES! Go check RAM
        JMP  SELONB             NO! Indicate error

SEL01A  MOVB R2,@>6000          Flip to bank one
        BLWP @RCOMP             See if RAM one is off
        JMP  SEL02A             YES! Check second bank
        NOP                     NO! Go check >00s

        CLR  R2
        BLWP @RCOMP             See if RAM one is off
```

```
        JMP   SELO2A              YES! Check second bank
        JMP   SELONB              NO! Indicate error

SELO2A  MOVB  R2,@>6002           Flip to bank two
        BLWP  @RCOMP              See if RAM two if off
        JMP   SELONC              YES! Continue with test
        NOP                       NO! Try >FFs

        SETO  R2
        BLWP  @RCOMP              See if RAM one is off
        JMP   SELONC              YES! Continue with test
        NOP                       NO! Indicate error

SELONB  EQU   $
        LI    R0,TOP+160          PUT ERROR
        LI    R1,GKOFF             MESSAGE
        LI    R2,14                ON
        BL    @VMBW                SCREEN
*       BLWP  @PRINT              Print GK OFF message
*       BYTE  17
*       TEXT  '  GK OFF failure'
*       BYTE  10

SELONC  MOV   @YFLAG,@YFLAG       Did we print anything?
        JEQ   SELO1C              NO!
        BLWP  @PRINT              YES! Do a form feed
        BYTE  1                      ..
        BYTE  12

SELO1C  LI    R0,648              Put finished message up
        LI    R1,DONE                .
        LI    R2,13                  .
        BL    @VMBW                  .

        BLWP  @SETSW              Reset GK
        TEXT  'MDDMU '                .
        BLWP  @WAIT               Wait for user

        B     @~~TEST~~STARTA     Return to main screen
*******************************************************************************
*******************************************************************************
~~LIST~~
SELTW   BL    @CLRSCN
        BLWP  @SETSW
        TEXT  'MUUUD '
        BLWP  @WAIT
*
* Determine if we are testing an expanded GK
*
        CLR   R0                  GRAM address of zero
        BL    @GWADD              Set up GRAM address
        BL    @GRDAT              Grab first byte
        CB    R1,@HAA             Is it an 'AA'?
        JEQ   SELTW1              YES!
        SETO  @XFLAG              NO! Set flag

SELTW1  LI    R0,>2000            Set GRAM address to TI BASIC
        BL    @GWADD              Set up address
        BL    @GRDAT              Get first byte
        CB    R1,@HAA             Is it an 'AA'?
```

```
        JEQ   SELTW2            YES!
        SETO  @XFLAG            NO! Set expanded GK flag

SELTW2  LI    R0,>6000          Set GRAM address to module space
        MOV   @XFLAG,@XFLAG     Do we have an expanded GK?
        JEQ   SELTW3            YES!
        CLR   R0                NO! Reset address to zero

SELTW3  SETO  R1                Finish address
        LI    R2,>E500          Byte that should be in each GRAM
        BLWP  @GCOMP            Go check it out
        JMP   SELTW4            Good!
        JMP   SELTW6            Battery backup error

SELTW4  LI    R2,>1100          Byte that should be in bank one
        BLWP  @RCOMP            Go check
        JMP   SELTW5            Good!
        JMP   SELTW6            Battery backup error

SELTW5  BLWP  @SETSW            Switch to bank two
        TEXT  '   D   '           .
        BLWP  @WAIT             Wait for user response

        LI    R2,>2200          Byte that should be in bank two
        BLWP  @RCOMP            Go check
        JMP   SELTW7            Good!
        NOP                     Battery backup error

SELTW6  EQU   $
        LI    R0,TOP+32         PUT ERROR
        LI    R1,BATT            MESSAGE
        LI    R2,22              ON
        BL    @VMBW              SCREEN
*       BLWP  @PRINT            Report error
*       BYTE  29
*       TEXT  '  Failed battery backup test'
*       BYTE  12
        BLWP  @WAIT
        B     @SELT7C           Return to title screen
*
* Passed all tests so put in header
*  for happy new owner
*
SELTW7  LI    R0,>6000          Set GRAM 3 address
        BL    @GWADD              .
        MOV   @>83FA,R2         Get GROM base
        LI    R1,HDEND-HD       Number of bytes to load into GRAM 3
        LI    R0,HD             Address of bytes
SELT7A  MOVB  *R0+,@>400(R2)     Move header into GRAM 3
        DEC   R1                Decrement counter
        JNE   SELT7A            Loop until done

        MOV   @XFLAG,@XFLAG     Do we have an expanded GK?
        JNE   SELT7B            NO!

        LI    R0,HDSKP          Don't let optional GRAMs
        BL    @GWADD             appear on the screen
        LI    R1,HDA-HD+H6K     Develop new vector
        MOVB  R1,@>400(R2)      Put it into header
        SWPB  R1                  .
```

```
        MOVB R1,@>400(R2)           .

SELT7B  BLWP @PRINT
        BYTE 21
        TEXT '  * Read/Write Test '
        BYTE 10
        BLWP @PRINT
        BYTE 23
        TEXT '       All Grams Passed'
        BYTE 10
        BLWP @PRINT
        BYTE 21
        TEXT '       GK Prom Passed'
        BYTE 10
        BLWP @PRINT
        BYTE 37
        TEXT '       RAM Banks One and Two Passed'
        BYTE 10,10,10
        BLWP @PRINT
        BYTE 25
        TEXT '  * Module Deselect Mode'
        BYTE 10
        BLWP @PRINT
        BYTE 27
        TEXT '       Module Grams Passed '
        BYTE 10
        BLWP @PRINT
        BYTE 37
        TEXT '       RAM Banks One and Two Passed'
        BYTE 10,10,10
        BLWP @PRINT
        BYTE 35
        TEXT '  * RAM Bank-Swap Control Passed'
        BYTE 10,10,10
        BLWP @PRINT
        BYTE 39
        TEXT '  * Write-Protection Control Passed '
        BYTE 10,10,10
        BLWP @PRINT
        BYTE 39
        TEXT '  * GK Prom-Override Control Passed '
        BYTE 10,10,10
        BLWP @PRINT
        BYTE 37
        TEXT '  * Battery-Backup Circuit Passed '
        BYTE 10,10,10
        BLWP @PRINT
        BYTE 43
        TEXT '         GRAM KRACKER UNIT PASSED ALL TESTS'
        BYTE 12

        LI   R0,648             Put finished message up
        LI   R1,DONE               .
        LI   R2,13                 .
        BL   @VMBW                 .

SELT7C  BLWP @SETSW             Reset GK for shipping
        TEXT ' DDMU '              .
        BLWP @WAIT              Wait for user for the last time
```

```
SELTW8  B     @~~TEST~~ STARTA      Return to main screen
*****************************************************************
*****************************************************************
BURNIN  BL    @CLRSCN               Clear screen

        BLWP  @SETSW                Set switches
        TEXT  'MUUUD '                 .
        BLWP  @WAIT                 Wait for user

        BL    @CLRSCN               Clear screen
        LI    R0,388                Put message up on screen
        LI    R1,HOLD                   .
        LI    R2,23                     .
        BL    @VMBW                     .

BURN    LI    R0,>81E0              Turn screen on
        BL    @VWDADD                   .
        LI    R3,>200               Load delay
BURN1   BLWP  @CHKKEY               Check for space bar
        MOVB  R0,R0                     .
        JEQ   BURN2                     .
        CB    R1,@H2000                 .
        JNE   BURN2                     .
        JMP   BURN3                 Abort if space bar
BURN2   DEC   R3                    Decrement counter
        JNE   BURN1                 Loop if not done
        LI    R0,>8180              Turn screen off
        BL    @VWDADD                   .

        CLR   R0                    Exercise GRAMs and RAMs
        SETO  R1                        .
        CLR   R2                        .
        BLWP  @GFILL                    .

        BLWP  @GCOMP                    .
        NOP                             .
        NOP                             .

        BLWP  @RFILL                    .

        BLWP  @RCOMP                    .
        NOP                             .
        NOP                             .

        SETO  R2                        .
        BLWP  @GFILL                    .

        BLWP  @GCOMP                    .
        NOP                             .
        NOP                             .

        BLWP  @RFILL                    .

        BLWP  @RCOMP                    .
        NOP                             .
        NOP                             .

        JMP   BURN                  Do it all again

BURN3   LI    R0,>81E0              Turn screen on
```

```
        BL    @VWDADD
        B     @~~TEST~~START          Return to title screen

        AORG  >FFFC                   Load vectors
        DATA  WSR,~~TEST~~START

SLAST   END
```

```
*
* Routine to put swithces up on the screen
*  Call with a BLWP folloded by a 5+byte string
*  for the position of each switch.  Up="U"
*  Down="D" and Middle="M"
*
SETSW  DATA WSR1,SW          Routine vectors

SW     LI   R5,FRSTSW        Init 1st screen location
       LI   R6,5             Init string counter
       CLR  R0               Clear comparator register

SW1    MOV  R6,R6            Are we done?
       JEQ  SW4              YES!
       MOVB *R14+,R0         Grab a char from the string
       LI   R3,UPADD         Init data
       LI   R4,UPDAT          registers
       CI   R0,'U'*>100      Is it an upper pos.?
       JEQ  SW2              YES!

       LI   R3,MDADD         Init data
       LI   R4,MDDAT          registers
       CI   R0,'M'*>100      Is it a middle pos.?
       JEQ  SW2              YES!

       CI   R0,' '*>100      Is it a space?
       JEQ  SW3              YES! Ignore
       LI   R3,DNADD         Init data
       LI   R4,DNDAT          registers

SW2    BLWP @WRTSW           Put the switch on the screen

SW3    AI   R5,OFFSET        Add next switch screen pos. of next switch
       DEC  R6               Decrement loop counter
       JMP  SW1              Loop

SW4    INC  R14              Correct return address to an even boundary
       RTWP                  Retunr to caller
*
* Subroutine that writes one of the switches to the screen
*  Called from SETSW
*
WRTSW  DATA WSR2,WSW         Routine vectors

WSW    MOV  R13,R6           Grab caller workspace base
       AI   R6,6             Add offset to point to R3
       MOV  *R6+,R3          Get address stack pointer
       MOV  *R6+,R4          Get char. stack pointer
       MOV  *R6,R5           Get screen offset

       LI   R1,>2000         Load a space into R1
       MOV  R5,R6            Fetch screen address
       AI   R6,-31           Back up a line minus a space
       LI   R7,5             Five lines to erase
WSW1   MOV  R6,R0            Move start address into R0
       LI   R2,3             Three spaces per line to blank
       BL   @RPTBLK          Blank a line
       AI   R6,32            Increment to next line
       DEC  R7               Are we done?
       JNE  WSW1             NO!
```

```
WSW2    MOV   *R3+,R0          Get screen address
        INV   R0               End of the stack?
        JEQ   WSW3             YES!
        INV   R0               NO! Restore address
        A     R5,R0            Add screen offset
        MOVB  *R4+,R1          Get char
        BL    @VSBW            Put char on screen

        JMP   WSW2             Loop

WSW3    RTWP                   Return to caller
```

```
*
*--------------------------------------------------------------------------------
* KEY OPERATIONS BLOCK
*
CHKKEY DATA WSR2,CHKKEZ        Routine vectors
CHKKEZ LWPI GPLWS
       BL   @KSCAN             Scan keyboard
       LWPI WSR2
       MOV  R13,R2
       MOVB @STATUS,*R2        Move STATUS into R0
       INCT R2
       CLR  *R2                Clear R1
       MOVB @KEYBRD,*R2        Move KEY into R1
       RTWP                    Return to caller
*
*--------------------------------------------------------------------------------
* VDP OPERATIONS BLOCK
*
VSBW   MOV  R11,R8             Save return address
       BL   @VWDADD            Set VDP address
       MOVB R1,@VDPWD          Move byte into VDP
       JMP  VDPRT              Return
VSBR   MOV  R11,R8             Save return address
       BL   @VRDADD            Set VDP address
       MOVB @VDPRD,R1          Read byte from VDP
       JMP  VDPRT              Return
VMBW   MOV  R11,R8             Save return address
       BL   @VWDADD            Set VDP address
VMBW1  MOVB *R1+,@VDPWD        Move byte into VDP
       DEC  R2                 Decrement loop counter
       JNE  VMBW1              Jump if not finished
       JMP  VDPRT              Return
VMBR   MOV  R11,R8             Save return address
       BL   @VRDADD            Set VDP address
VMBR1  MOVB @VDPRD,*R1+        Read data from VDP
       DEC  R2                 Decrement loop counter
       JNE  VMBR1              Loop if not finished
VDPRT  B    *R8                Return
VWDADD ORI  R0,>4000           Set write data bit
VRDADD SWPB R0                 Swap LSB into place
       MOVB R0,@VDPWA          Write LSB of address
       SWPB R0                 Swap MSB into place
       MOVB R0,@VDPWA          Write MSB of address
       ANDI R0,>3FFF
       RT                      Return
*
*--------------------------------------------------------------------------------
* Clear screen and repeat block routines
*
RPTBLK MOV  R11,R10            Save return address
       JMP  CLRSC1
CLRSCN MOV  R11,R10            Save return address
       CLR  R0                 Load screen address
       LI   R1,>2000           Load an EDGE character into R1
       LI   R2,767             Load R2 with number of spaces left to clear
CLRSC1 BL   @VSBW              Write first space to set VDP address
CLRSC2 MOVB R1,@VDPWD          Move spaces onto the screen
       DEC  R2                 Are we done?
       JNE  CLRSC2             NO!
       B    *R10               Return to caller
```

```
*
*===============================================================================
*       GROM OPERATIONS BLOCK
*
GWADD   MOV   @>83FA,R2     Set up for GROM address
        MOVB  R0,@>402(R2)  Set GROM address
        SWPB  R0
        MOVB  R0,@>402(R2)
        SWPB  R0
        RT                  Return to caller
*
GRADD   MOV   @>83FA,R2     Set up for GROM address
        MOVB  @2(R2),R0     Set GROM address
        SWPB  R0
        MOVB  @2(R2),R0
        SWPB  R0
        DEC   R0
        RT                  Return to caller
*
GRDAT   MOV   @>83FA,R2     Set up for GROM address
        MOVB  *R2,R1        Read GROM byte
        RT                  Return to caller
*
GWDAT   MOV   @>83FA,R2     Set up for GROM address
        MOVB  R1,@>400(R2)  Write GROM byte
        RT                  Return to caller
*
*===============================================================================
* KEYBOARD SCAN
*
ROWBAS  EQU   >24
COLBAS  EQU   >06
OLDMOD  EQU   >83C7
DBNCE   EQU   >83C8
*
* Perform some initializations
*
KSCAN   LI    R1,5                Set row counter
        CLR   R2                  Assume no key down at present
        CLR   R6                     .
        CLR   R7                  Assume no modifiers (CTRL, FNTN and SHIFT)
        CLR   R12
        SBO   21                  Turn alpha lock line off
*
* Scan the keyboard for a key
*
ROWLP   LI    R12,ROWBAS          CRU base for row selection
        SWPB  R1                  Move row number into place
        LDCR  R1,3                Turn specified row on
        SWPB  R1                  Restore row counter
        LI    R12,COLBAS          CRU base for column selection
        SETO  R4                  Clean column register
        STCR  R4,8                Fetch column data
        INV   R4                  Make one's represent keys down
        MOV   R1,R1               Are we on row 0?
        JNE   NOTONE              NO!
        MOVB  R4,R7               Save CTRL and/or FCTN keys
        ANDI  R4,>0F00            Mask out CTRL and/or FCTN bits

NOTONE  MOVB  R4,R4               Do we have a key?
```

```
        JEQ   NXTROW            NO! Try next line
*
* Determine which key is depressed
*
GOTKEY  MOV   R2,R2             Is this the second key being held down?
        JNE   NXTROW            YES! Just ignore it
        SETO  R2                Indicate that the first key press is found
        MOV   R1,R3             Get the row number
        SLA   R3,3              Multiply it by 8
        DEC   R3                Adjust it for the INC that follows
CNTLP   INC   R3                Add in a column
        SLA   R4,1              Shift the key bit into the carry
        JNC   CNTLP             Loop if we have'nt hit the key bit yet
        MOV   R1,R1             Are we working with line 0?
        JEQ   NXTROW            YES!
        LI    R1,1              NO! Force next line to be zero
*
* Select next line to be polled
*
NXTROW  DEC   R1                Decrement row number
        JOC   ROWLP             If not finished then go scan next line
        MOV   R2,R2             Was a key pressed this time through?
        JNE   DEBOUN            YES!
*
* No keys are down so wrap things up
*
NOKEY   CLR   R6                No key so reset flag register
        MOVB  R6,@OLDMOD        Clear any old modifiers
H700    EQU   $
        SETO  R0                Load R0 with >FFFF
        CB    R0,@DBNCE         Was a key just released?
        JEQ   NOKEY2            NO!
        LI    R12,1250          YES! Do some debounce. Load 10mS
KIL1    DEC   R12               Loop and kill some time
        JNE   KIL1                 .

NOKEY2  MOVB  R0,@DBNCE         Clear keyboard debounce register
        JMP   OLDCHR            Go indicate keyboard state to user
*
* A key is down so do a debounce if necessary
*
DEBOUN  CB    @R3LB,@DBNCE      Is the same key station depressed as last time?
        JEQ   MODIFY            YES!
H2000   EQU   $+2
HB20    EQU   H2000
        LI    R6,>2000          Load new key flag
        LI    R12,1250          Load 10mS of debounce time
KIL2    DEC   R12               Loop for awhile
        JNE   KIL2                 .
        MOVB  @R3LB,@DBNCE      Indicate which keyboard station is depressed

NEWMOD  MOVB  R7,@OLDMOD        Move modifiers into modifier register
*
* We have the key station now get the key code from tables
*
MODIFY  MOVB  @OLDMOD,R7        Fetch modifiers
        LI    R1,KFNCTN         Load FCTN character table address
        SLA   R7,2              Is the FCTN key down?
        JOC   MAPIT             YES!
        LI    R1,KFNCTN         Load FCTN character table address
```

```
        SRL   R7,15             Is the FCTN key down?
        JOC   MAPIT             YES!
        LI    R1,KSHIFT         Load the SHIFT character table address
        DEC   R7                Is the SHIFT key(s) down?
        JEQ   MAPIT             YES!
        LI    R1,KEYTAB         Must be unmodified keyboard!

MAPIT   A     R3,R1             Copy table offset into R1
        MOVB  *R1,R0            Get key value
*
* Here's where we check for the alpha lock
*
        LI    R12,'az'          Load lower character set range
        CB    R0,R12            Is key value less than an 'a'?
        JL    RSTP5             YES!
        CB    R0,@R12LB         Is key value greater than a 'z'?
        JH    RSTP5             YES!
        CLR   R12               Reset CRU base
        SBZ   21                Turn alpha lock line on
        SRC   R12,14            Waste some time
        TB    7                 Is alpha lock down?
        JEQ   RSTP5             NO!
        SB    @H2000,R0         Map lower case into upper case
RSTP5   SBO   21                Turn alpha lock line off
*
* Move key value out and set/reset STATUS

OLDCHR  MOVB  R0,@KEYBRD        Move key value (or >FF) into key register
        MOVB  R6,@STATUS        Load new key flag (if any) into GPL STATUS byte
        RT                      Return to caller
*
*===============================================================================
*
* Keyboard tables
*
KEYTAB  DATA  >FFFF,>FFFF,>FF0D,>203D  '****** ='
        DATA  >7877,>7332,>396F,>6C2E  'xws29ol.'
        DATA  >6365,>6433,>3869,>6B2C  'ced38ik,'
        DATA  >7672,>6634,>3775,>6A6D  'vrf47ujm'
        DATA  >6274,>6735,>3679,>686E  'btg56yhn'
        DATA  >7A71,>6131,>3070,>3B2F  'zqa10p;/'
KSHIFT  DATA  >FFFF,>FFFF,>FF0D,>202B  '****** +'
        DATA  >5857,>5340,>284F,>4C3E  'XWS@(OL>'
        DATA  >4345,>4423,>2A49,>4B3C  'CED#*IK<'
        DATA  >5652,>4624,>2655,>4A4D  'VRF$&UJM'
        DATA  >4254,>4725,>5E59,>484E  'BTG%^YHN'
        DATA  >5A51,>4121,>2950,>3A2D  'ZQA!)P:-'
KFNCTN  DATA  >FFFF,>FFFF,>FF0D,>2005  '****** *'
        DATA  >0A7E,>0804,>0F27,>C2B9  '*~***'**'
        DATA  >600B,>0907,>063F,>C1B8  ''****?**'
        DATA  >7F5B,>7B02,>015F,>C0C3  '*[{**_**'
        DATA  >BE5D,>7D0E,>0CC6,>BFC4  '*]}*****'
        DATA  >5CC5,>7C03,>BC22,>BDBA  '\*|**"**'
*
* DSR link subroutine
*
SCLEN   EQU   >8355
SCNAME  EQU   >8356
CRULST  EQU   >83D0
SADDR   EQU   >83D2
```

```
*
***UTILITY BLWP VECTORS
*
DSRLNK DATA DSRWS,DLENTR      Link to device service routine
*
***DATA
*
DECMAL TEXT '.'
HAA    BYTE >AA
*
***LINK TO DEVICE SERVICE ROUTINE
*
DLENTR SZCB @H2000,R15        Reset equal bit
       MOV  @SCNAME,R0        Fetch pointer into PAB
       MOV  R0,R9             Save pointer
       AI   R9,-8             Adjust pointer to flag byte
       BL   @VSBR             Read device name length
       MOVB R1,R3             Store it elsewhere
       SRL  R3,8              Make it a word value
       SETO R4                initialize a counter
       LI   R2,NAME           Point to NAME
LNK$LP INC  R0                Point to next char of name
       INC  R4                Increment character counter
       C    R4,R3             End of name?
       JEQ  LNK$LN            YES
       CB   *R2+,@DECMAL      Have we hit a decimal point?
       JNE  LNK$LP            NO
LNK$LN CI   R4,7              Is name length >7
       JGT  LNKERR            YES! Error
       CLR  @CRULST
       MOV  R4,@SCLEN-1       Store name length for search
       INC  R4                Adjust it
       A    R4,@SCNAME        Point to position after name
*
***SEARCH ROM FOR DSR
*
SROM   LWPI GPLWS             Use GPL workspace to search
       CLR  R1                Version found of DSR etc.
       LI   R12,>0F00         start over again
NOROM  MOV  R12,R12           Anything to turn off
       JEQ  NOOFF             NO
HB1E   EQU  $
       SBZ  0                 YES! Turn it off
H100   EQU  $+2
NOOFF  AI   R12,>0100         Next ROM's turn on
       CLR  @CRULST           Clear in case we're finished
       CI   R12,>2000         At the end
       JEQ  NODSR             No more ROMs to turn on
       MOV  R12,@CRULST       Save address of next CRU
HEDGE  EQU  $
       SBO  0                 Turn on ROM
       LI   R2,>4000          Start at beginning
       CB   *R2,@HAA          Is it a valid ROM?
       JNE  NOROM             NO
       AI   R2,8              Go to first pointer
       JMP  SGO2
SGO    MOV  @SADDR,R2         Continue where we left off
       SBO  0                 Turn ROM back on
SGO2   MOV  *R2,R2            Is address a zero
       JEQ  NOROM             YES! No program to look at
```

```
        MOV  R2,@SADDR          Remember where we go next
        INCT R2                 Go to entry point
        MOV  *R2+,R9            Get entry address
*
***SEE IF NAME MATCHES
*
        MOVB @SCLEN,R5          Get length as counter
        JEQ  NAME2              Zero length, don't do match
        CB   R5,*R2+            Does length match?
        JNE  SGO                NO
        SRL  R5,8               Move to right place
        LI   R6,NAME            Point to NAME
NAME1   CB   *R6+,*R2+          Is character correct?
        JNE  SGO                NO
        DEC  R5                 More to look at?
        JNE  NAME1              YES
NAME2   INC  R1                 Next version found
        BL   *R9                Match, call subroutine
        JMP  SGO                Not right version
        SBZ  0                  Turn off ROM
        LWPI DSRWS              Select DSRLNK workspace
        MOV  R9,R0              Point to flag byte in PAB
        BL   @VSBR              Read flag byte
        SRL  R1,13              Just want the error flags
        JNE  IOERR              ERROR!
        RTWP
*
***ERROR HANDLING
*
NODSR   LWPI DSRWS              Select DSRLNK workspace
LNKERR  CLR  R1                 Clear the error flags
IOERR   SWPB R1
        MOVB R1,*R13            Store error flags in calling R0
        SOCB @H2000,R15         Indicate an error occured
        RTWP                    Return to caller
*
*===============================================================================
*
*       FORMAT
*
*       This routine loads data to the screen from a formatted stack
*
*       ENTER: Start of stack in DATA after BL
*
*       FORMAT OF STACK:
*                               Word 1 -- Screen position. Zero if done
*                               Word 2 -- String length
*                               Rest   -- String
*
*+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
*
FORMAT  MOV  *R11+,R1           Grab start address of stack
        MOV  R11,R10            Save return address
FORMA1  MOV  *R1+,R0            Load screen location
        JEQ  FORMA2             If zero the return
        MOV  *R1,R3             Move string length into R3
        MOV  *R1+,R2            Load string length
        A    R1,R3              Add address to R3. R3 now points to next entry
        BL   @VMBW              Put string on the screen
        MOV  R3,R1              Restore link
```

```
        JMP   FORMA1          Loop until entire stack is written
FORMA2  B     *R10            Return
```

```
*
* Wait routine
*
WAIT    DATA WSR1,WAIT1

CONT    TEXT 'SET SWITCHES/PRESS ANY KEY'
SND     EQU  >8400
BEEP    BYTE >80,>05,>92
KSND    BYTE >9F

WAIT1   LI   R1,BEEP
        MOVB *R1+,@SND
        MOVB *R1+,@SND
        MOVB *R1,@SND

        LI   R0,706
        LI   R1,CONT
        LI   R2,26
        BL   @VMBW

        LI   R8,150
WAIT2   DEC  R8
        JNE  WAIT3
        MOVB @KSND,@SND
WAIT3   BLWP @CHKKEY
        MOVB R0,R0
        JEQ  WAIT2

        MOVB @KSND,@SND
        LI   R0,706
        LI   R1,BUSY
        LI   R2,26
        BL   @VMBW

        RTWP
```