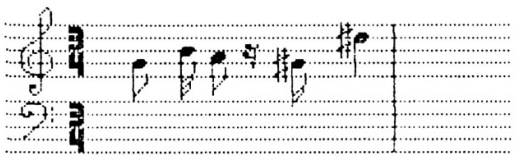
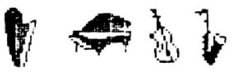
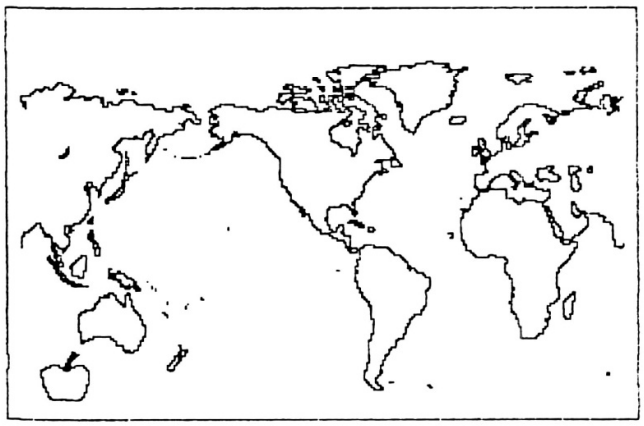
Visible Memory Printer Dump



Dr. Frank Covitz





The MTU visible memory is 8K bytes of dynamic RAM which, during refresh (transparent to the 6502), generates a video image of itself. The display signal is standard composite video, and can be seen on a conventional monitor or converted TV set. With the MTU/PET interface, the PET screen itself can be used as the display. The 320 (horizontal) by 200 (vertical) pixel matrix allows you to generate moderately high resolution graphics. (64,000 individual pixels can be set on or off - obviously a job for 6502 machine language or routines callable by BASIC).

The following 6502 program allows you to get a hard copy of this on the CBM 2022 (tractor feed) PRINTER. The first part is fairly self-documenting, and is used to open the special character channel to the printer and set the vertical spacing (not available on the CBM 2020 pressure feed printer). The VMDUMP machine code scans each 320H by 7V line to form the special character matrix. Skipping the proper number of spaces, the main program then prints this character.

This is a SLOW process, since the 2022 printer can handle only one special character per line, so as many as 53 prints to the same line may be required before the line is complete. Since 30 lines may be needed to complete the 8K scan, the whole process can take up to 30 minutes to finish!! In practice, however, sinces spaces are "weeded out", 5-10 minutes is usually sufficient to get a moderately dense print-out, and less for line-type graphs. Horizontal and vertical

```
VM PRNTR DUMP
2
                        ;
                        ;*****************
                        ;*               *
                        ;* VM PRNTR DUMP *
                        ;*               *
                        ;* BY F.COVITZ   *
                        ;*               *
                        ;*****************
                        ;
                        ;THIS PROGRAM DUMPS AN 8K BLOCK
                        ;TO THE CBM 2022 PRINTER AS A
                        ;BIT MAPPED IMAGE.
                        ;MAINLY INTENDED FOR USE WITH THE
                        ;M.T.U. 8K VISIBLE MEMORY.
                        ;FORMAT IS 320H*200V PIXELS
                        ;
260  0280                       *=   $0280          ;ORIGIN (EXAMPLE ONLY)
                        ;
                        ;*** UPGRADE ROM Z-PAGE LOC'NS ***
                        ;
300  0280          LENGTH   =   $D1            ;LENGTH OF FILE NAME
310  0280          LFN      =   $D2            ;LOGICAL FILE NO.
320  0280          SA       =   $D3            ;SECONDARY ADDRESS
330  0280          DN       =   $D4            ;DEVICE NO.
                        ;
                        ;*** CONSTANTS ***
360  0280          LF1      =   1              ;NORMAL PRINT CHANNEL
370  0280          SA1      =   0              ;NORMAL PRINT SEC.ADDR.
380  0280          LF2      =   2              ;VERT. SPACING CHANNEL
390  0280          SA6      =   6              ;VERT. SPACING SEC.ADDR.
400  0280          LF3      =   3              ;SPEC. CHAR. CHANNEL
410  0280          SA5      =   5              ;SPEC. CHAR. SEC.ADDR.
420  0280          PRNTR    =   4              ;PHYSICAL DEVICE # OF PRINTER
430  0280          SPACE    =   " "
440  0280          CR       =   $8D            ;CARRIAGE RETURN (WITHOUT LF)
450  0280          CRLF     =   $0D            ;CARRIAGE RETURN (WITH LF)
460  0280          MAX      =   54             ;MAX # OF SPACES (54*6=324)
470  0280          REV      =   $FE            ;'REVERSE' KEY
480  0280          SCHAR    =   $FE            ;SPEC.CHARACTER
```

registration are not perfect, given the limitations of the 2022's mechanism, and there is some distortion (vertical or horizontal, depending on your outlook). The accompanying examples should give you some idea of the results you can expect. Given the slow speed and imperfect registration, the fact still remains that the capability is there.

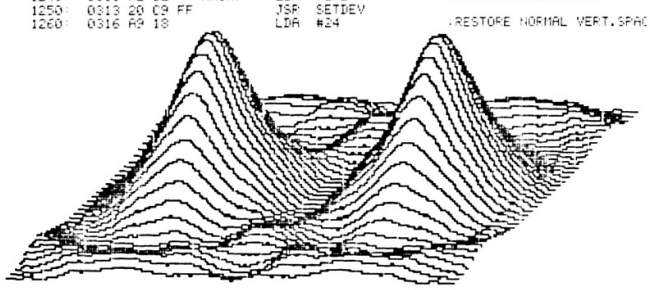
As written it is configured to go into both tape buffers (starts at $0280, 640dec.). It is then easily accessed by DISK-based systems. When you see something you like on the VM, just key in SYS640 and take a five minute break. If you have a tape-based system, you will need to relocate the code elsewhere. The assembly source should make this relatively easy to do.

The only routine that is specific to upgrade ROM is the OPEN subroutine. I believe this is at $F52D in original ROM. Zero-page locations $D1, $D2, $D3 and $D4 are ROM dependent, and correspond to locations $EE, $EF, $F0, and $F1, respectively for original ROM. Zero page locations 1 and 2 are used as an indirect pointer. The last point to be aware of is the setting of VMORG, the origin page of the 8K block of memory. There is a single LDA #VMORG in the source code, so you must change this single byte if you want a dump of a different 8k block.

To be even more benign to the calling program, the original contents of zero page locations 1 and 2 (VM and VM + 1), as well as the registers, could be saved. Use the following code sequence:

```
MAIN SEI ;prevent interrupts
   PHP ;save status
   TXA ;save registers
   PHA
   TYA
   PHA
   LDA VM ;save loc'ns 1 and 2
   PHA
   LDA VM + 1
   PHA
   .
   .
   .
```

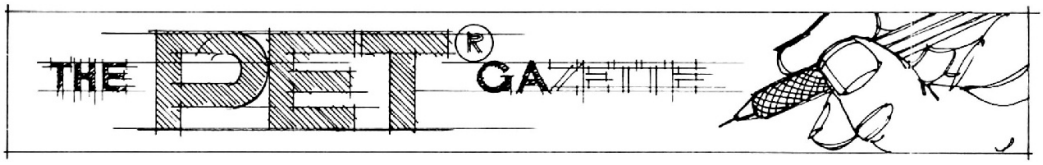
```
                        ;
                        ;*** SYSTEM LOCATIONS ***
                        ;***                  ***
                        ;*** ! = UPGRADE ROM  ***
                        ;
540:  0280              PIAK    =   $E812       ;KEYBOARD PORT
550:  0280              CLOSE   =   $FFE7       ;CLOSE IEEE DEVICES
560:  0280              OPEN    =   $F524       ;! OPEN IEEE DEVICE
570:  0280              SETDEV  =   $FFC9       ;SET OUTPUT DEVICE
580:  0280              OUTCHAR =   $FFD2       ;SEND 1 CHAR.

600:  0280 78           MAIN    SEI             ;PREVENT INTERRUPTS
610:  0281 A9 00                LDA  #0         ;SET NULL NAME
620:  0283 85 D1                STA  LENGTH
630:  0285 85 D3                STA  SA         ;OPEN NORMAL PRINT CHANNEL
640:  0287 A9 01                LDA  #LF1
650:  0289 85 D2                STA  LFN
660:  028B A9 04                LDA  #PRNTR
670:  028D 85 D4                STA  DN
680:  028F 20 24 F5             JSR  OPEN
690:  0292 A9 02                LDA  #LF2       ;OPEN VERT.SPACING CHANNEL
700:  0294 85 D2                STA  LFN
710:  0296 A9 06                LDA  #SA6
720:  0298 85 D3                STA  SA
730:  029A 20 24 F5             JSR  OPEN
740:  029D A9 03                LDA  #LF3       ;OPEN SPEC.CHAR.CHANNEL
750:  029F 85 D2                STA  LFN
760:  02A1 A9 05                LDA  #SA5
770:  02A3 85 D3                STA  SA
780:  02A5 20 24 F5             JSR  OPEN
790:  02A8 A2 01                LDX  #LF1       ;SEND 'HOME'
800:  02AA 20 C9 FF             JSR  SETDEV
810:  02AD A9 13                LDA  #"S"
820:  02AF 20 D2 FF             JSR  OUTCHAR
830:  02B2 A2 02                LDX  #LF2       ;SEND VERTICAL SPACING
840:  02B4 20 C9 FF             JSR  SETDEV
850:  02B7 A9 10                LDA  #16
860:  02B9 20 D2 FF             JSR  OUTCHAR
870:  02BC 20 20 03             JSR  VMDUMP     ;INITIAL CALL
880:  02BF AD 12 E8 MAIN1       LDA  PIAK       ;CHECK 'REVERSE' KEY
890:  02C2 C9 FE                CMP  #REV
900:  02C4 F0 4B                BEQ  MAIN7      ;ABORT ON REVERSE KEY
910:  02C6 AD FE AF             LDA  SPC
920:  02C9 C9 36                CMP  #MAX       ;CHECK IF LINE FEED NEEDED
930:  02CB D0 04                BNE  MAIN2
940:  02CD A9 0D                LDA  #CRLF      ;SEND A CRLF
950:  02CF D0 35                BNE  MAIN5
960:  02D1 A2 03        MAIN2   LDX  #LF3       ;SPEC.CHAR.CHANNEL
970:  02D3 20 C9 FF             JSR  SETDEV
980:  02D6 A0 06                LDY  #6         ;6 BYTES TO SEND
990:  02D8 B9 F5 AF MAIN3       LDA  MATRIX-1,Y ;TRANSMIT TO PRINTER
1000: 02DB 20 D2 FF             JSR  OUTCHAR
1010: 02DE 88                   DEY
1020: 02DF D0 F7                BNE  MAIN3
1030: 02E1 38                   SEC             ;HOW MANY SPACES TO TRANSMIT
1040: 02E2 A9 38                LDA  #MAX+2     ;LEAVE A LITTLE ROOM
1050: 02E4 ED FE AF             SBC  SPC
1060: 02E7 A8                   TAY             ;USE AS INDEX
1070: 02E8 A2 01                LDX  #LF1       ;SET FOR NORMAL PRINT CHANNEL
1080: 02EA 20 C9 FF             JSR  SETDEV
1090: 02ED A9 20        MAIN4   LDA  #SPACE
1100: 02EF 20 D2 FF             JSR  OUTCHAR    ;SEND SPACES TO PRINTER
1110: 02F2 88                   DEY
1120: 02F3 D0 F8                BNE  MAIN4
1130: 02F5 A2 01                LDX  #LF1
1140: 02F7 20 C9 FF             JSR  SETDEV
1150: 02FA A9 FE                LDA  #SCHAR     ;TRANSMIT SPECIAL CHAR.
1160: 02FC 20 D2 FF             JSR  OUTCHAR
1170: 02FF A2 01                LDX  #LF1
1180: 0301 20 C9 FF             JSR  SETDEV
1190: 0304 A9 8D                LDA  #CR        ;SEND CR (WITHOUT LF)
1200: 0306 20 D2 FF MAIN5       JSR  OUTCHAR
1210: 0309 20 77 03 MAIN6       JSR  VMNEXT     ;RE-ENTER VMDUMP
1220: 030C AD FF AF             LDA  LINES      ;CHECK FOR DONE
1230: 030F D0 AE                BNE  MAIN1
1240: 0311 A2 02        MAIN7   LDX  #LF2       ;CLOSE ALL CHANNELS
1250: 0313 20 C9 FF             JSR  SETDEV
1260: 0316 A9 18                LDA  #24        ;RESTORE NORMAL VERT.SPACING
```



```
PLA restore loc'ns 1 and 2
STA VM + 1
PLA
STA VM
PLA ;restore registers
TAY
PLA
TAX
PLP ;restore status
CLI ;interrupts
RTS
```

```
1270:  0318 20 D2 FF          JSR  OUTCHAR
1280:  031B 20 E7 FF          JSR  CLOSE          ;CLOSE ALL CHANNELS
1290:  031E 58                CLI                 ;ALLOW INTERRUPTS
1300:  031F 60                RTS
                              ;
                              ;*********************
                              ;* VMDUMP SUBROUTINE *
                              ;*********************
                              ;
                              ; DOES THE TOUGH JOB OF FORMING
                              ;THE SPECIAL CHARACTER MATRIX.
                              ;IT SKIPS OVER SPACES (BUT KEEPS
                              ;TRACK OF THEM). THE ROUTINE
                              ;CLOBBERS A,X,AND Y
                              ;
                              ;!! VMORG SET = $90 !!
                              ;SO SET YOUR OWN IF NECESSARY.
                              ;
1450:  0320                   VMORG    =   $90            ;VIS.MEM.
1460:  0320                   VMEND    =   256*VMORG+$1FFF  ;LAST VM LOC'NS USED AS RAM
                              ;
                              ;ZERO PAGE LOC'NS 1 AND 2 USED
1490:  0320                   VM       =   1              ;INDIRECT POINTER
                              ;
1510:  0320                   MATRIX   =   VMEND-9        ;THE CHARACTER MATRIX
1520:  0320                   BTPT     =   VMEND-3        ;BIT LOCATION
1530:  0320                   CNTR     =   VMEND-2        ;COUNTS 7 BITS PER BYTE
1540:  0320                   SPC      =   VMEND-1        ;COUNTS NO.OF SPACES
1550:  0320                   LINES    =   VMEND          ;COUNTS NO.OF LINES
                              ;
1570:  0320 A9 00             VMDUMP   LDA  #0            ;ENTRY POINT
1580:  0322 85 01                      STA  VM            ;INITIALIZE THE POINTER
1590:  0324 A9 90                      LDA  #VMORG
1600:  0326 85 02                      STA  VM+1
1610:  0328 A2 09                      LDX  #9            ;FETCH INITIAL DATA
1620:  032A BD 9E 03 VMD0              LDA  DATA,X
1630:  032D 9D F6 AF                   STA  MATRIX,X      ;AND STORE IT
1640:  0330 CA                         DEX
1650:  0331 10 F7                      BPL  VMD0          ;ALL 10 BYTES
1660:  0333 A2 05             VMD1     LDX  #5            ;6 BYTES TO FORM
1670:  0335 AC FD AF VMD2              LDY  CNTR          ;7 BITS TO ASSIGN
1680:  0338 B9 97 03                   LDA  OFSTAB,Y      ;OFFSET TO 8K BLOCK
1690:  033B A8                         TAY
1700:  033C B1 01                      LDA  (VM),Y        ;FETCH A BYTE
1710:  033E 2C FC AF                   BIT  BTPT          ;RETURNS Z=0 IF BIT OFF
1720:  0341 18                         CLC
1730:  0342 F0 01                      BEQ  VMD3
1740:  0344 38                         SEC
1750:  0345 3E F6 AF VMD3              ROL  MATRIX,X      ;THE BYTE IS FORMED
1760:  0348 CE FD AF                   DEC  CNTR          ;DO 7 BITS
1770:  034B 10 E8                      BPL  VMD2
1780:  034D A9 06                      LDA  #6            ;RESTORE BIT COUNTER
1790:  034F 8D FD AF                   STA  CNTR
1800:  0352 18                         CLC
1810:  0353 6E FC AF                   ROR  BTPT          ;ADVANCE 1 BIT RIGHT
1820:  0356 D0 09                      BNE  VMD4          ;BTPT NOT DONE
1830:  0358 6E FC AF                   ROR  BTPT          ;RESTORE BTPT =#80
1840:  035B E6 01                      INC  VM            ;NEXT ADDRESS
1850:  035D D0 02                      BNE  VMD4          ;BTPT NOT DONE
1860:  035F E6 02                      INC  VM+1
```

```
1870: 0361 CA       VMD4    DEX
1880: 0362 10 D1            BPL  VMD2          ;DO THE 6 BYTES
1890: 0364 A2 05            LDX  #5            ;NOW CHECK FOR ALL ZERO
1900: 0366 18               CLC
1910: 0367 BD F6 AF VMD5    LDA  MATRIX,X
1920: 036A 29 7F            AND  #$7F          ;MASK BIT 7
1930: 036C 9D F6 AF         STA  MATRIX,X
1940: 036F F0 01            BEQ  VMD6          ;SKIP ON ZERO
1950: 0371 38               SEC                ;A NON-ZERO WAS FOUND
1960: 0372 CA       VMD6    DEX                ;SKIP ON ZERO
1970: 0373 10 F2            BPL  VMD5          ;A NON-ZERO WAS FOUND
1980: 0375 B0 1F            BCS  VMD8          ;RETURN ON A NON-ZERO BYTE
                    ;
                    ;RE-ENTRY POINT HERE
                    ;
2020: 0377 CE FE AF VMDNEXT DEC  SPC           ;OTHERWISE IT IS A SPACE
2030: 037A D0 B7            BNE  VMD1          ;SO KEEP GOING
2040: 037C A9 36            LDA  #MAX          ;BUT NOT PAST 54 SPACES
2050: 037E 8D FE AF         STA  SPC           ;RESTORE SPACE COUNTER
2060: 0381 CE FF AF         DEC  LINES         ;DECREMENT LINES COUNTER
2070: 0384 F0 0B            BEQ  VMD7          ;RETURN IF LINES=0
2080: 0386 18               CLC
2090: 0387 A5 01            LDA  VM            ;OTHERWISE GO TO NEXT
2100: 0389 69 F1            ADC  #$F1          ;LINE OF 8K BLOCK
2110: 038B 85 01            STA  VM            ;(6*40+1) LOCATIONS AHEAD
2120: 038D 90 02            BCC  VMD7
2130: 038F E6 02            INC  VM+1
2140: 0391 A9 80    VMD7    LDA  #$80          ;RESTORE BTPT
2150: 0393 8D FC AF         STA  BTPT
2160: 0396 60       VMD8    RTS
2170: 0397 F0 C8 A0 OFSTAB  .BYTE $F0,$C8,$A0,$78  ;OFFSETS TO NEXT LINE
2180: 039B 50 28 00         .BYTE $50,$28,0
2190: 039E 00 00 00 DATA    .BYTE 0,0,0,0,0,0      ;INITIAL DATA
2200: 03A4 80 06 35         .BYTE $80,6,MAX-1,29
                    ;END
```



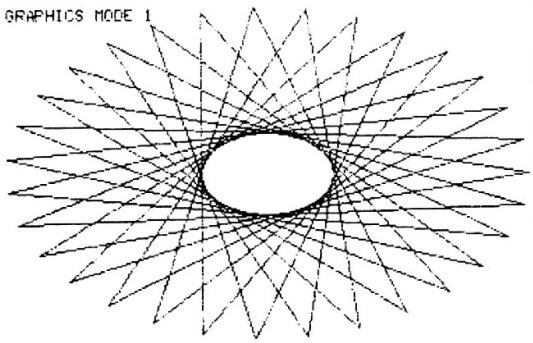
A Fast Visible Memory Dump

Martin J. Cohen, Ph.D.
Los Angeles, CA

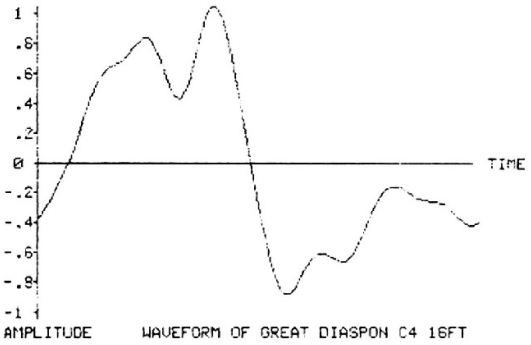
"The MTU Visible Memory is 8K bytes of dynamic RAM which, during refresh (transparent to the 6502), generates a video image of itself. The 320 (horizontal) by 200 (vertical) pixel display allows you to generate moderately high resolution graphics. (64,000 individual pixels can be set on or off — obviously a job for 6502 machine language or routines callable by BASIC.)"

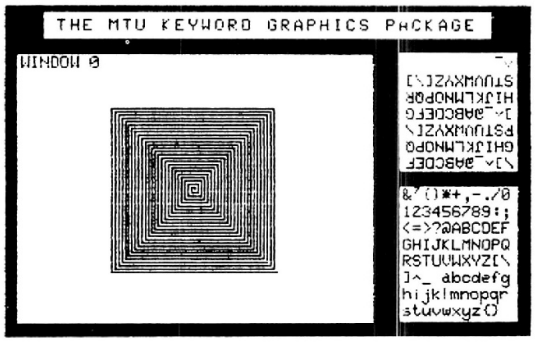
This description (on page 104 of **COMPUTE!**, issue 7, Nov./Dec., 1980) begins Dr. Frank Covitz's article in which he gives a truly ingenious method of using Commodore's 2022 tractor-feed printer to produce a hard copy of the MTU Visible memory. The primary disadvantage of this method is that, because the 2022 was not designed for graphics output, the process can take 10 to 30 minutes.

The 6502 machine language program described here, called SDUMP, produces a hard copy of the Visible Memory on Integral Data Systems' "Paper Tiger" printers with DotPlot graphics. Because these printers have graphics built in, the Visible Memory can be dumped in 90 seconds on any Paper Tiger and in only 45 seconds on the Paper Tiger 460 run at 9600 baud. These times apply to any contents of the Visible Memory, no matter how complicated or dense. The routine SDUMP does not even take advantage of clear areas of the Visible Memory, and could presumably be speeded up if this were done.



To see some of the capabilities of the Visible Memory/Paper Tiger combination, examine figures 1 through 3. Figure 1 shows four of the





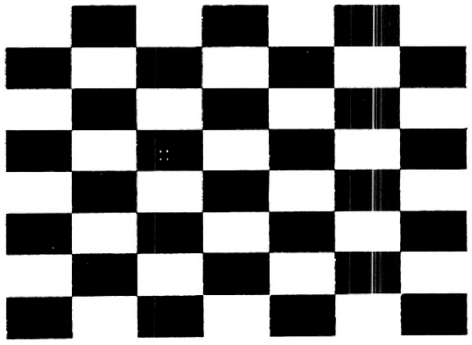


Figure 1

P1 (DENSITY) FOR K = 4, N = 2

Q	ALPHA = 0.	.125	.25	.5	1	2	4
	RHO = 0.	.5	1	2	4	8	16
.05	.285	.181686	.115778	.046960	.007693	.000203	0.
.1	.540000	.361678	.241868	.107708	.021050	.000768	0.
.15	.765000	.538112	.377244	.183775	.042380	.002078	.000004
.2	.960000	.708930	.520502	.276517	.074611	.004827	.000015
.25	1.125	.871864	.669801	.386991	.121355	.010224	.000052
.3	1.26	1.024422	.822792	.515782	.186954	.020314	.000166
.35	1.365	1.163877	.976548	.662790	.276438	.038453	.000491
.4	1.44	1.287245	1.127478	.826958	.395385	.069998	.001387
.45	1.485	1.391273	1.271233	1.005928	.549572	.123247	.003769
.5	1.5	1.472420	1.402602	1.195608	.744344	.210629	.009896
.55	1.485	1.526836	1.515399	1.389644	.983511	.350033	.025174
.6	1.44	1.550348	1.602333	1.578762	1.267587	.565844	.062113
.65	1.365	1.538434	1.654867	1.749954	1.591018	.888714	.148533
.7	1.26	1.486205	1.663062	1.885497	1.938004	1.351842	.343298
.75	1.125	1.388381	1.615403	1.961747	2.276254	1.979278	.762388
.8	.960000	1.239265	1.498604	1.947681	2.547874	2.757421	1.608574
.85	.765	1.032722	1.297404	1.803133	2.656193	3.572946	3.152575
.9	.54	.762148	.994325	1.476662	2.446936	4.086017	5.446845
.95	.285	.420448	.569423	.903005	1.681591	3.482044	7.005667

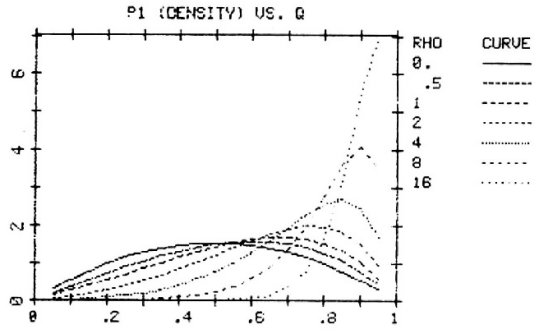


Figure 2

screens produced by the demonstration program supplied with the visible Memory. Figures 2 and 3 show some intermixed text and graphics produced using the MTU Keyword Graphics Package, of which I am the principal author. This package interfaces with BASIC to allow graphics commands to be entered as part of your BASIC program. Listing 1 shows the code used to produce the plots in figures 2 and 3.

The principal problem in dumping the Visible Memory to the Paper Tiger is that a byte of the Visible Memory is displayed as 8 pixels lined up horizontally, while a byte output to the Paper Tiger in graphics mode produces, depending on the model, 6 or 7 dots lined up vertically. The main task of SDUMP is therefore to take 6 or 7 bytes in the Visible Memory which are lined up vertically and convert them to 8 bytes of 6 or 7 bits which will then be output to the Paper Tiger.

My first attempt at this was done in BASIC, and is in listing 2. I knew it would execute extremely slowly, but it would be much easier to debug. Once the code was working, it was a fairly straightforward matter to translate the BASIC into assembly language — since I knew the logic was correct, I only had to make sure the translation was correct. Another advantage of this method is that if I want to program the routine in some other language, such as PASCAL, FORTH, or FORTRAN, it will be much easier to do it with BASIC as the basis instead of assembly language.

The current version of SDUMP is in listing 3. It is a modularized form of the BASIC code in listing 2, and is designed to be easily modifiable. It is assembled starting at $6000 (hex), so that it can reside in memory with the MTU Keyword Graphics Package, and be called with a SYS (96*256).

The initial part of SDUMP contains a transfer vector and a data area. The transfer vector has jumps to the three main routines in SDUMP: OUTVM, which dumps the whole Visible Memory; OUTROW, which dumps 6 or 7 rows of the Visible Memory starting at the location set in VM (at $6013); and OUTCOL, which outputs a column of

```
P1 (CUMULATIVE) FOR K = 4, N = 2

Q    ALPHA = 0.          .125        .25         .5          1           2           4
        RHO = 0.          .5          1           2           4           8           16
 .05          .01425      .009084     .005788     .002348     .000384     .000010     0.
 .1           .04125      .027168     .017882     .007733     .001437     .000048     0.
 .15          .0795       .054073     .036744     .016922     .003556     .000152     0.
 .2           .1275       .089520     .062769     .030748     .007286     .000393     .000001
 .25          .18375      .133113     .096259     .050097     .013354     .000905     .000003
 .3           .24675      .184334     .137399     .075886     .022702     .001920     .000012
 .35          .315        .242528     .186226     .109026     .036524     .003843     .000036
 .4           .387        .306890     .242600     .150374     .056293     .007343     .000105
 .45          .46125      .376454     .306162     .200670     .083772     .013505     .000294
 .5           .53625      .450075     .376292     .260451     .120989     .024037     .000789
 .55          .6105       .526417     .452062     .329933     .170164     .041539     .002047
 .6           .6825       .603934     .532179     .408871     .233544     .069831     .005153
 .65          .75075      .680856     .614922     .496369     .313095     .114267     .012580
 .7           .81375      .755166     .698075     .590643     .409995     .181859     .029745
 .75          .870000     .824586     .778845     .688731     .523808     .280823     .067864
 .8           .918000     .886549     .853775     .786115     .651201     .418694     .148293
 .85          .956250     .938185     .918646     .876272     .784011     .597341     .305922
 .9           .983250     .976292     .968362     .950105     .906358     .801642     .578264
 .95          .997500     .997315     .996833     .995255     .990437     .975744     .928547
```

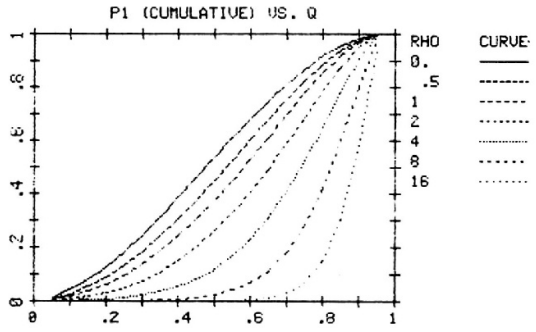


Figure 3

```
PE FOR K = 4, N = 2

M    ALPHA = 0.          .125        .25         .5          1           2           4
        RHO = 0.          .5          1           2           4           8           16
2            .509553     .454439     .408615     .338910     .259034     .218670     .208887
4            .759892     .708395     .661886     .582550     .469299     .362902     .359455
```

8 bytes, 6 or 7 bits high. These routines are made available in this manner in case you would like to mix text and graphics in a more sophisticated manner than a simple dump.

Following the transfer vector is the data area. The values here specify how the Visible Memory is to be dumped and where it is. SDUMP is assembled to work with the 460 Paper Tiger, but by making the changes described in lines 25-27, the code will work on the Paper Tiger 440. Presumably, with similarly minor changes, SDUMP will also work on the newest Paper Tiger, the 445.

The following should be noted about SDUMP and its use: The only code in SDUMP that is specific to a particular version of BASIC is that in OUTCH, lines 235-280. This code was given to me by Greg Yob — thanks Greg. It outputs the character in the ACC directly to the device whose number is in RDEV, at location $600E in the data area. Because this code bypasses the PET's file system and directly accesses the IEEE-488 routines, the device does not even have to be opened.

Each routine in SDUMP checks to see if the stop key is pressed, using the routine STOPTS at lines 281-292. If so, the routine quits and returns to the routine which called it. Because of the way the Paper Tigers enter and exit graphics mode, it is possible for them to be left in graphics mode when the stop key is pressed. If this happens, you will know it when it does, the easiest method of recovering is to turn the printer off, then on.

You should not have a CMD operation open

```
4000 REM PLOT X IN (XA,XB) NX WIDE, Y IN (YA,YB) NY HIGH, KEY PK
4005 IF PF=0 THEN RETURN
4010 CLEAR
4060 CX=NX/(XB-XA):REM CONVERSION CONSTANTS
4070 CY=NY/(YB-YA)
4090 FOR JR=1 TO NR:REM GET THE DATA POINTS
4100 DOTL DO(JR,1),DO(JR,2)
4110 FOR JQ=1 TO NQ
4120 X=QS(JQ)
4130 IX=INT(.5+(X-XA)*CX)
4140 IF IX<0 THEN IX=0:REM MAKE SURE X OK
4150 IF IX>NX THEN IX=NX
4160 IX=IX+OX
4210 IF PK=1 THEN Y=DD(JQ,JR):REM PK=1 FOR P1 DENS DIST
4220 IF PK=2 THEN Y=CD(JQ,JR):REM PK=2 FOR P1 CUM DIST
4300 REM CONVERT Y LIKE X
4310 IY=INT(.5+(Y-YA)*CY)
4320 IF IY<0 THEN IY=0:REM FORCE ON PLOT
4330 IF IY>NY THEN IY=NY
4335 IY=IY+OY
4340 IF JQ=1 THEN MOVE IX,IY
4350 DRAW IX,IY
4480 NEXT JQ
4490 NEXT JR
4495 DOTL 1,0
4500 REM PRODUCE THE PLOT
4510 MOVE OX,OY:REM BORDER
4520 DRAW OX+NX,OY:DRAW OX+NX,OY+NY
4530 DRAW OX,OY+NY:DRAW OX,OY
4590 MOVE OX+NX/2-3*(LEN(PL$)+6),OY+NY+10
4592 CHAR PL$;" VS. Q"
4594 PL$=""
4600 REM DISPLAY RHO AND DOTS
4610 IX=OX+NX+10:IY=OY+NY-7
4620 MOVE IX,IY:CHAR "RHO     CURVE"
4630 FOR I=1 TO NR
4640 V=RS(I):GOSUB2002
4650 IY=IY-12:MOVE IX,IY
4660 CHAR V$
4670 DOTL DO(I,1),DO(I,2):LINE IX+42,IY+3,319,IY+3
4680 NEXT I
4690 DOTL 1,0
4700 REM DRAW A GRID
4702 TL=3:REM TIC LENGTH
4705 DX=.1:REM X GRID SPACING (ALWAYS)
4710 DY=10:REM Y SPACING - HAVE TO SEARCH
4715 IF YB/5<DY THEN DY=DY/10:GOTO4715
4720 EX=INT(XB/DX+.01):EY=INT(YB/DY+.01):REM POINTS ON GRID
4725 FX=1:IF EX>5 THEN FX=2:IF EX>10 THEN FX=5:IF EX>20 THEN FX=10
4730 FY=1:IF EY>5 THEN FY=2:IF EY>10 THEN FY=5:IF EY>20 THEN FY=10
4735 FOR I=0 TO EY:OZ=OY+I*DY*CY:LINE OX-TL,OZ,OX+TL,OZ:REM Y AXIS
4737 LINE OX+NX-TL,OZ,OX+NX+TL,OZ
4740 IF I=FY*INT(I/FY) THEN CHROT 1:MOVE OX-TL-5,OZ-3:CHAR MID$(STR$(I*DY),2)
4745 NEXT I
4750 FOR I=0 TO EX:OZ=OX+I*DX*CX:LINE OZ,OY-TL,OZ,OY+TL:REM X AXIS
4752 LINE OZ,OY+NY-TL,OZ,OY+NY+TL
4755 IF I=FX*INT(I/FX) THEN CHROT 0:MOVE OZ-3,OY-TL-10:CHAR MID$(STR$(I*DX),2)
4760 NEXT I
4900 REM PRODUCE THE PLOT
4910 PRINT#PRINT
4920 CMD3:REM REGULAR OUT TO THE SCREEN
4930 SYS(LP):REM THERE IT GOES
4940 CMD1:REM BACK TO THE PRINTER
4950 RETURN
```

Listing 1

to the Paper Tiger when SDUMP is called, because of the way this command is interpreted in the IEEE-488 system. To avoid this, open a unit to the screen (device 3) and switch to this unit before invoking SDUMP. For example:

```
OPEN 1,4:REM PRINTER FILE
OPEN 2,3:REM SCREEN FILE
CMD 1:REM OUTPUT TO PRINTER
.....
CMD 2:REM DIVERT OUTPUT
SYS(96*256):REM DUMP VISIBLE MEMORY
CMD 1:REM RESUME PRINTER OUTPUT
```

The byte in the data area called EORVAL (at $6011) is exclusive-ored with each Visible Memory byte when it is accessed for dumping. This gives a visible indication of the progress of the dump which I find entertaining. It is actually an instance

of Cohen's first law of interactive computing — "Always let the operator know that something is going on." However, this leaves the screen reversed when the dump finishes. If you do not like this, there are (at least) two possibilities: (1) Set EORVAL to zero ($00); the exclusive or will than not change anything. (2) If you are using the Keyword Graphics Package, follow the call to SDUMP with a 'SCFLIP 0,0,319,199'; this will reverse the whole screen, restoring its original condition.

To load SDUMP together with the MTU keyword Graphics Package, when reserving memory space, do a 'POKE 53,96' instead of 'POKE 53,98' for a 32K system, and similarly for smaller systems. This will reserve the two pages needed by SDUMP.

Listing 2

```
9100 REM MTU VISIBLE MEMORY TO IDS 460 PAPER TIGER SCREEN DUMP
9110 PRINT#U:REM SPACE
9120 PRINT#U,CHR$(3);:REM ENTER GRAPHICS MODE
9130 VM=256*PEEK(832):REM START OF VISIBLE MEMORY
9140 PVMEM
9142 GRSHRT
9145 S=7:REM ROWS PER GROUP
9150 FOR R0=0 TO 199 STEP S:REM S ROWS AT A TIME
9160 :R1=R0+S-1:REM END OF ROW GROUP
9170 :IF R1>199 THEN R1=199
9180 :FOR C=0 TO 39:REM A BYTE (8 BIT COLUMNS) AT A TIME
9190 ::FOR I=0 TO 7:REM CLEAR VALUES TO BE PRINTED
9200 :::P(I)=0
9210 ::NEXT I
9220 ::V=VM+C:REM LOC OF BYTE
9225 ::P2=1:REM POWER OF 2 TO ADD
9230 ::FOR R=R0 TO R1:REM SCAN THE ROWS
9235 :::PRINT C;R
9240 :::B=PEEK(V):REM GET THE BYTE (8 BITS)
9250 :::V=V+40:REM LOC OF BYTE BELOW
9260 :::IF B=0 THEN 9315:REM FASTER IF EMPTY
9270 :::M=1:REM MASK (2^(7-I))
9280 :::REM ACCUMULATE VALUES FOR PRINTING
9290 :::FOR I=7 TO 0 STEP -1
9295 ::::IF (B AND M)<>0 THEN P(I)=P(I)+P2
9300 ::::M=M+M
9310 :::NEXT I
9315 :::P2=P2+P2
9320 ::NEXT R:REM DO THE ROWS
9330 ::REM NOW, PRINT THE 8 COLUMNS OF ROWS
9340 ::FOR I=0 TO 7
9350 :::PRINT#U,CHR$(P(I));
9360 :::IF P(I)=3 THEN PRINT#U,CHR$(P(I));:REM 3 IS SPECIAL
9370 ::NEXT I
9390 :NEXT C:REM END OF COLUMN LOOP
9400 :PRINT#U,CHR$(3);CHR$(14);:REM GRAPHICS LINE FEED/RETURN
9410 :VM=VM+S*40:REM DOWN S ROWS
9420 NEXT R0:REM END OF ROW GROUP LOOP
9430 PRINT#U,CHR$(3);CHR$(2):REM LEAVE GRAPHICS MODE
9439 JI
9440 VISMEM
9450 RETURN:REM DONE
```

Listing 3

```
00001  0000                ; SDUMP.ASM - MTU TO IDS PAPER TIGER 460 (440) SCREEN DUMP
00002  0000                ;
00003  0000                ; BY MARTIN J. COHEN, DECEMBER 1980
00004  0000                ;
00005  0000                ; ANYONE WHO WANTS TO CAN USE THIS PROGRAM,
00006  0000                ; ALTHOUGH SOME ACKNOWLEDGEMENT WOULD BE APPRECIATED
00007  0000                ;
00008  0000                ;
00009  0000                ; APPROXIMATE TIME NEEDED TO DUMP VISIBLE MEMORY:
00010  0000                ;
00011  0000                ; AT 1200 BAUD, 1 MIN, 30 SEC
00012  0000                ; AT 9600 BAUD, 45 SEC (WITH 3 MS DELAY SET BY NMSDLY, BELOW)
00013  0000                ;
00014  0000                ; THE ACTUAL CPU TIME NEEDED IS ABOUT 3 SECONDS!!
00015  0000                ;
00016  0000                ;
00017  0000                      * = $6000        ; TWO PAGES BELOW KGP CODE
00018  6000                ;
00019  6000  4C 18 60            JMP OUTVM        ; SKIP DATA AREA AND DUMP THE VIS MEM
00020  6003  4C 8D 60            JMP OUTROW       ; OUTPUT ROW STARTING AT VM
00021  6006  4C DF 60            JMP OUTCOL       ; OUTPUT A COLUMN OF 8 BYTES
00022  6009                ;
00023  6009                ; DATA AREA ...
00024  6009                ;
00025  6009                ; NOTE - TO RUN THIS ROUTINE ON A 440 INSTEAD OF A 460,
00026  6009                ; CHANGE THE FOLOWING VALUES AS INDICATED (VALUES IN DECIMAL):
00027  6009                ; RPFXC=0, RREP=33, RVAL=6, REND=2, RXGR=11.
00028  6009                ;
00029  6009  14            RPFXC   .BYTE 20         ; NUMBER OF BLANK PREFIX COLUMNS (440:0)
00030  600A  02            RPFXR   .BYTE 2          ; NUMBER OF BLANK PREFIX ROWS
00031  600B  1C            RREP    .BYTE 28         ; MAIN REPETITION COUNT (440:33)
00032  600C  07            RVAL    .BYTE 7          ; ROWS TO OUTPUT IN MAIN LOOP (440 6)
```

```
00033  600D  04          REND    .BYTE 4          ; ROWS TO OUTPUT AT END (440:2)
00034  600E              ; THE TOTAL NUMBER OF ROWS OUTPUT = RREP*RVAL + REND = 200
00035  600E  04          RDEV    .BYTE 4          ; OUTPUT DEVICE
00036  600F  0E          RXGR    .BYTE 14         ; GRAPHICS RETURN (440:11)
00037  6010  03          NMSDLY  .BYTE 3          ; MS TO DELAY AFTER EACH BYTE
00038  6011  FF          EORVAL  .BYTE $FF        ; VALUE TO EOR WITH SCREEN LOC ($FF TO
FLIP,  0 TO SKIP)
00039  6012  90          VMPAGE  .BYTE $90        ; STARTING PAGE OF VISIBLE MEMORY
00040  6013              ;
00041  6013  00 00       VM      .WORD 0          ; LOCAL STORAGE - LOC OF A VIS MEM ROW
00042  6015  28          BYTEPL  .BYTE 40         ; BYTES PER VM LINE
00043  6016  00          RREPX   .BYTE 0          ; STORAGE FOR REP COUNT
00044  6017  00          RVALX   .BYTE 0          ; STORAGE FOR ROW COUNT
00045  6018              ;
00046  6018              ; OUTVM - OUTPUT THE WHOLE VISIBLE MEMORY
00047  6018              ;
00048  6018  AD 12 60    OUTVM   LDA VMPAGE       ; SET LOC OF VM
00049  601B  8D 14 60            STA VM+1
00050  601E  A9 00               LDA #0
00051  6020  8D 13 60            STA VM
00052  6023  AD 0B 60            LDA RREP         ; SET MAJOR REP COUNT
00053  6026  8D 16 60            STA RREPX
00054  6029  20 88 60            JSR ENTRGR       ; ENTER GRAPHICS MODE
00055  602C  AE 0A 60            LDX RPFXR        ; SEE IF ANY PREFIX ROWS
00056  602F  F0 06               BEQ OUTVM1
00057  6031  20 C5 60    OUTVM0  JSR OUTRET       ; IF SO, OUTPUT THEM
00058  6034  CA                  DEX
00059  6035  D0 FA               BNE OUTVM0
00060  6037              ;
00061  6037              OUTVM1  = *
00062  6037  20 BA 61            JSR STOPTS       ; CHECK FOR STOP KEY
00063  603A  B0 3A               BCS OUTVMF
00064  603C  20 B1 60            JSR OUTPFX       ; OUTPUT A LINE P
00065  603F  AD 0C 60            LDA RVAL         ; SET ROW COUNT
00066  6042  8D 17 60            STA RVALX
00067  6045  20 8D 60            JSR OUTROW       ; OUTPUT A ROW
00068  6048  20 C5 60            JSR OUTRET       ; OUTPUT A RETURN
00069  604B  AE 0C 60            LDX RVAL         ; SET VM = VM+RVAL*40
00070  604E  18          OUTVM2  CLC
00071  604F  AD 13 60            LDA VM
00072  6052  6D 15 60            ADC BYTEPL
00073  6055  8D 13 60            STA VM
00074  6058  AD 14 60            LDA VM+1
00075  605B  69 00               ADC #0
00076  605D  8D 14 60            STA VM+1
00077  6060  CA                  DEX
00078  6061  D0 EB               BNE OUTVM2
00079  6063              ;
00080  6063  CE 16 60            DEC RREPX        ; COUNT ROWS
00081  6066  D0 CF               BNE OUTVM1
00082  6068              ;
00083  6068  20 B1 60            JSR OUTPFX       ; START OF LAST ROW
00084  606B  AD 0D 60            LDA REND         ; NUMBER OF ROWS
00085  606E  F0 06               BEQ OUTVMF       ; SKIP IF NONE
00086  6070  8D 17 60            STA RVALX
00087  6073  20 8D 60            JSR OUTROW       ; THERE IT GOES
00088  6076              OUTVMF  = *
00089  6076  20 C5 60            JSR OUTRET
00090  6079  20 7D 60            JSR EXITGR       ; LEAVE GRAPHICS MODE
00091  607C  60                  RTS              ; DONE
00092  607D              ;
00093  607D              ; EXITGR - LEAVE GRAPHICS MODE
00094  607D              ;
00095  607D  A9 03       EXITGR  LDA #3
00096  607F  20 6D 61            JSR OUTCH
00097  6082  A9 02               LDA #2
00098  6084  20 6D 61            JSR OUTCH
00099  6087  60                  RTS
00100  6088              ;
00101  6088              ; ENTRGR - ENTER GRAPHICS MODE
00102  6088              ;
00103  6088  A9 03       ENTRGR  LDA #3
00104  608A  4C 6D 61            JMP OUTCH
00105  608D              ;
00106  608D              ; OUTROW - OUTPUT THE ROW POINTED TO BY VM, RVALX DEEP
00107  608D              ;
00108  608D  AD 13 60    OUTROW  LDA VM           ; SET WHERE TO START
00109  6090  8D D1 60            STA V
```

```
00110  6093  AD 14 60          LDA VM+1
00111  6096  8D D2 60          STA V+1
00112  6099  AE 15 60          LDX BYTEPL      ; DO 40 COLUMNS
00113  609C  AD 17 60  OUTR1   LDA RVALX       ; SET DEPTH COUNT
00114  609F  8D D3 60          STA R
00115  60A2  20 DF 60          JSR OUTCOL      ; OUTPUT THOSE 8
00116  60A5  EE D1 60          INC V           ; BUMP LOC
00117  60A8  D0 03             BNE OUTR2
00118  60AA  EE D2 60          INC V+1
00119  60AD  CA          OUTR2  DEX             ; COUNT
00120  60AE  D0 EC             BNE OUTR1
00121  60B0  60                RTS             ; DONE
00122  60B1                   ;
00123  60B1                   ; OUTPFX - OUTPUT RPFXC SPACES TO START LINE
00124  60B1                   ;
00125  60B1  AE 09 60  OUTPFX LDX RPFXC
00126  60B4  F0 0E             BEQ OUTPF2      ; CHECK FOR NONE
00127  60B6  20 7D 60          JSR EXITGR
00128  60B9  A9 20             LDA #32         ; LOAD THE SPACE
00129  60BB  20 6D 61  OUTPF1 JSR OUTCH        ; OUTPUT IT
00130  60BE  CA                DEX             ; UNTIL DONE
00131  60BF  D0 FA             BNE OUTPF1
00132  60C1  20 88 60          JSR ENTRGR
00133  60C4  60          OUTPF2 RTS             ; THAT'S ALL
00134  60C5                   ;
00135  60C5                   ; OUTRET - OUTPUT A GRAPHICS RETURN
00136  60C5                   ;
00137  60C5  A9 03      OUTRET LDA #3
00138  60C7  20 6D 61          JSR OUTCH
00139  60CA  AD 0F 60          LDA RXGR
00140  60CD  20 6D 61          JSR OUTCH
00141  60D0  60                RTS
00142  60D1                   ;
00143  60D1                   ; OUTCOL - OUTPUT 8 COLUMNS OF BITS
00144  60D1                   ;
00145  60D1                   ; PARAMETERS (BELOW) ARE V AND R
00146  60D1                   ;
00147  60D1  00 00      V       .WORD 0         ; LOC IN VISIBLE MEMORY
00148  60D3  00         R       .BYTE 0         ; NUMBER OF ROWS TO PROCESS
00149  60D4             PO      * = *+8         ; RESULT TO OUTPUT
00150  60DC                   ;
00151  60DC  00         P2      .BYTE 0         ; POWER OF 2 BIT
00152  60DD  00         B       .BYTE 0         ; STORAGE FOR A BYTE
00153  60DE  00         M       .BYTE 0         ; A MASK
00154  60DF                   ;
00155  60DF             PGZ     = 1             ; PAGE ZERO LOCATION TO USE
00156  60DF                   ;
00157  60DF  48          OUTCOL PHA             ; SAVE REGS
00158  60E0  8A                TXA
00159  60E1  48                PHA
00160  60E2  98                TYA
00161  60E3  48                PHA
00162  60E4  A5 01             LDA PGZ         ; SAVE PAGE ZERO AREA
00163  60E6  48                PHA
00164  60E7  A5 02             LDA PGZ+1
00165  60E9  48                PHA
00166  60EA  20 BA 61          JSR STOPTS      ; SEE IF STOP PRESSED
00167  60ED  B0 72             BCS MOVEPF      ; IF SO, QUIT NOW
00168  60EF  A9 00             LDA #0          ; ZERO P(0:7)
00169  60F1  A2 07             LDX #7
00170  60F3  9D D4 60  CLP2   STA PO,X
00171  60F6  CA                DEX
00172  60F7  10 FA             BPL CLP2
00173  60F9
00174  60F9  A9 01             LDA #1          ; SET P2 TO 1
00175  60FB  8D DC 60          STA P2
00176  60FE  AD D1 60          LDA V           ; STORE VM LOC
00177  6101  85 01             STA PGZ
00178  6103  AD D2 60          LDA V+1
00179  6106  85 02             STA PGZ+1
00180  6108  A0 00      RLOOP  LDY #0
00181  610A  B1 01             LDA (PGZ)Y      ; GET VM BYTE
00182  610C  8D DD 60          STA B           ; SAVE IT
00183  610F  4D 11 60          EOR EORVAL      ; REVERSE IT FOR SHOW
00184  6112  91 01             STA (PGZ)Y
00185  6114  A5 01             LDA PGZ         ; POINT TO NEXT ROW
00186  6116  18                CLC
00187  6117  6D 15 60          ADC BYTEPL
```

```
00188  611A  85 01        STA PGZ
00189  611C  A5 02        LDA PGZ+1
00190  611E  69 00        ADC #0
00191  6120  85 02        STA PGZ+1
00192  6122               ;
00193  6122  A9 01        LDA #1          ; SET MASK TO 1
00194  6124  8D DE 60      STA M
00195  6127  A2 07        LDX #7          ; FOR I = 7 TO 0 STEP -1
00196  6129  AD DD 60  ILOOP  LDA B        ; IF B AND M <> 0
00197  612C  2D DE 60      AND M
00198  612F  F0 09        BEQ ILOOP1
00199  6131  BD D4 60      LDA PO,X        ; P(I)=P(I)+P2
00200  6134  0D DC 60      ORA P2
00201  6137  9D D4 60      STA PO,X
00202  613A  0E DE 60  ILOOP1 ASL M        ; SHIFT MASK LEFT
00203  613D  CA            DEX             ; SEE IF DONE
00204  613E  10 E9        BPL ILOOP
00205  6140               ;
00206  6140  0E DC 60      ASL P2          ; DOUBLE P2
00207  6143  CE D3 60      DEC R           ; SEE IF OUTER LOOP DONE
00208  6146  D0 C0        BNE RLOOP
00209  6148               ;
00210  6148               ; OUTPUT PO(0:7)
00211  6148               ;
00212  6148  A0 00        LDY #0
00213  614A  20 BA 61  MOVEP  JSR STOPTS   ; SEE IF STOP PRESSEDD
00214  614D  B0 12        BCS MOVEPF      ; IF SO, QUIT HERE
00215  614F  B9 D4 60      LDA PO,Y
00216  6152  20 6D 61      JSR OUTCH       ; OUTPUT A CHARACTER
00217  6155  C9 03        CMP #3          ; SEE IF 3
00218  6157  D0 03        BNE MOVEP1
00219  6159  20 6D 61      JSR OUTCH       ; IF SO, DO IT AGAIN
00220  615C  C8        MOVEP1 INY
00221  615D  C0 08        CPY #8          ; ONLY DO 8
00222  615F  D0 E9        BNE MOVEP
00223  6161               ;
00224  6161  68        MOVEPF PLA          ; RESTORE PAGE ZERO AREA
00225  6162  85 02        STA PGZ+1
00226  6164  68            PLA
00227  6165  85 01        STA PGZ
00228  6167  68            PLA             ; RESTORE REGS
00229  6168  A8            TAY
00230  6169  68            PLA
00231  616A  AA            TAX
00232  616B  68            PLA
00233  616C               ;
00234  616C  60            RTS
00235  616D               ;
00236  616D               ; OUTCH - OUTPUT A CHARACTER TO DEVICE RDEV
00237  616D               ;
00238  616D               ; THIS ROUTINE WAS SUPPLIED BY GREG YOB - THANKS MUCH
00239  616D               ;
00240  616D  8E B7 61  OUTCH  STX OUTCHX   ; SAVE REGS
00241  6170  8C B8 61      STY OUTCHY
00242  6173  48            PHA
00243  6174  A5 D4        LDA $D4         ; SAVE CURRENT DEVICE
00244  6176  8D B9 61      STA TMPDEV
00245  6179  AD 0E 60      LDA RDEV        ; SET MY DEVICE
00246  617C  85 D4        STA $D4
00247  617E  20 BA F0      JSR $F0BA       ; LISTEN
00248  6181  20 2D F1      JSR $F12D       ; ATTENTION
00249  6184           OUTCH1 = *
00250  6184  20 BA 61      JSR STOPTS      ; SEE IF STOP PRESSED
00251  6187  B0 11        BCS OUTCH2      ; IF SO, EXIT FROM HERE
00252  6189  A9 00        LDA #0          ; CLEAR STATUS
00253  618B  85 96        STA $96
00254  618D  68            PLA             ; REGET CHAR
00255  618E  48            PHA
00256  618F  85 A5        STA $A5         ; STORE WHERE IT SHOULD BE
00257  6191  20 EE F0      JSR $F0EE       ; OUTPUT
00258  6194  A5 96        LDA $96         ; SEE IF TIMED OUT
00259  6196  25 01        AND 1
00260  6198  D0 EA        BNE OUTCH1      ; IF SO, TRY AGAIN
00261  619A           OUTCH2 = *
00262  619A  20 83 F1      JSR $F183       ; UNLISTEN
00263  619D  AD B9 61      LDA TMPDEV      ; RESTORE DEVICE
00264  61A0  85 D4        STA $D4
00265  61A2  AE 10 60      LDX NMSDLY      ; DELAY A FEW MS
```

```
00266  61A5  F0 08           BEQ OUTCHF
00267  61A7  A0 C8    OUTCH3 LDY #200        ; 1 MS INNER LOOP (LOOP IS 5 CYCLES LONG)
00268  61A9  88        OUTCH4 DEY
00269  61AA  D0 FD           BNE OUTCH4
00270  61AC  CA               DEX
00271  61AD  D0 F8           BNE OUTCH3
00272  61AF           OUTCHF = *
00273  61AF  AE B7 61         LDX OUTCHX      ; AND REGS
00274  61B2  AC B8 61         LDY OUTCHY
00275  61B5  68               PLA
00276  61B6  60               RTS             ; DONE
00277  61B7           ;
00278  61B7  00       OUTCHX .BYTE 0
00279  61B8  00       OUTCHY .BYTE 0
00280  61B9  00       TMPDEV .BYTE 0
00281  61BA           ;
00282  61BA           ; STOPTS - SET CARRY IF STOP KEY PRESSED
00283  61BA           ;
00284  61BA           ; TAKEN FROM PAGE 5 OF MTU DOCUMENTATION FOR K-1002-6C
00285  61BA           ;
00286  61BA  AD 12 E8 STOPTS LDA $E812       ; LOOK AT KEYBOARD
00287  61BD  C9 EF           CMP #$EF        ; TEST FOR STOP KEY
00288  61BF  18             CLC             ; CARRY CLEAR FOR NO STOP
00289  61C0  D0 01           BNE STOPT1
00290  61C2  38             SEC             ; CARRY SET FOR YES STOP
00291  61C3  60     STOPT1 RTS
00292  61C4           ;
00293  61C4                  .END
ERRORS = 00000
SYMBOL TABLE
SYMBOL VALUE
 B        60DD    BYTEPL   6015    CLP2     60F3    ENTRGR   6088
 EORVAL   6011    EXITGR   607D    ILOOP    6129    ILOOP1   613A
 M        60DE    MOVEP    614A    MOVEP1   615C    MOVEPF   6161
 NMSDLY   6010    OUTCH    616D    OUTCH1   6184    OUTCH2   619A
 OUTCH3   61A7    OUTCH4   61A9    OUTCHF   61AF    OUTCHX   61B7
 OUTCHY   61B8    OUTCOL   60DF    OUTPF1   60BB    OUTPF2   60C4
 OUTPFX   60B1    OUTR1    609C    OUTR2    60AD    OUTRET   60C5
 OUTROW   608D    OUTVM    6018    OUTVM0   6031    OUTVM1   6037
 OUTVM2   604E    OUTVMF   6076    P2       60DC    PGZ      0001
 PO       60D4    R        60D3    RDEV     600E    REND     600D
 RLOOP    6108    RPFXC    6009    RPFXR    600A    RREP     600B
 RREPX    6016    RVAL     600C    RVALX    6017    RXGR     600F
 STOPT1   61C3    STOPTS   61BA    TMPDEV   61B9    V        60D1
 VM       6013    VMPAGE   6012
END OF ASSEMBLY
```

```
CROSS REFERENCE.....PAGE    1

B       $60DD   152   182   196
BYTEPL  $6015    42    72   112   187
CLP2    $60F3   170   172
ENTRGR  $6088    54   103   132
EORVAL  $6011    38   183
EXITGR  $607D    90    95   127
ILOOP   $6129   196   204
ILOOP1  $613A   198   202
M       $60DE   153   194   197   202
MOVEP   $614A   213   222
MOVEP1  $615C   218   220
MOVEPF  $6161   167   214   224
NMSDLY  $6010    37   265
OUTCH   $616D    96    98   104   129
                138   140   216   219   240
OUTCH1  $6184   249   260
OUTCH2  $619A   251   261
OUTCH3  $61A7   267   271
OUTCH4  $61A9   268   269
OUTCHF  $61AF   266   272
OUTCHX  $61B7   240   273   278
OUTCHY  $61B8   241   274   279
OUTCOL  $60DF    21   115   157
OUTPF1  $60BB   129   131
OUTPF2  $60C4   126   133
OUTPFX  $60B1    64    83   125
OUTR1   $609C   113   120
OUTR2   $60AD   117   119
OUTRET  $60C5    57    68    89   137

OUTROW  $608D    20    67    87   108
OUTVM   $6018    19    48
OUTVM0  $6031    57    59
OUTVM1  $6037    56    61    81
OUTVM2  $604E    70    78
OUTVMF  $6076    63    85    88
P2      $60DC   151   175   200   206
PGZ     $0001   155   162   164   177   179
                181   184   185   188   189
                191   225   227
PO      $60D4   149   170   199   201   215
R       $60D3   114   148   207
RDEV    $600E    35   245
REND    $600D    33    84
RLOOP   $6108   180   208
RPFXC   $6009    29   125
RPFXR   $600A    30    55
RREP    $600B    31    52
RREPX   $6016    43    53    80
RVAL    $600C    32    65    69
RVALX   $6017    44    66    86   113
RXGR    $600F    36   139
STOPT1  $61C3   289   291
STOPTS  $61BA    62   166   213   250   286
TMPDEV  $61B9   244   263   280
V       $60D1   109   111   116   118   147
                176   178
VM      $6013    41    49    51    71    73
                 74    76   108   110
VMPAGE  $6012    39    48
```