

GT40

ROM BOOTSTRAP
MD-11-DDGTD-B

EP DDGTD B DL A
COPYRIGHT 1976
FICHE 1 OF 1

NOV 1976
digital
MADE IN USA

This microfiche card contains a grid of frames. The first column on the left contains 16 frames, each displaying a list of data points, likely memory addresses and their corresponding values. The remaining three columns contain frames with various data representations, including what appears to be a hex dump or similar binary data format. The text is small and difficult to read due to the low resolution of the scan.

100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120

B01

GT40 ROM BOOTSTRAP TEST MAINDEC-11-DDGTD-B
DDGTD.P11

MACY11 27(732) 09-SEP-76 15:17 PAGE 1

1

.REM *

.TITLE GT40 ROM BOOTSTRAP TEST MAINDEC-11-DDGTD-B

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DDGTD-B-D
PRODUCT NAME: GT40 ROM VERIFY
DATE CREATED: NOVEMBER 1, 1973
MAINTAINER: DIAGNOSTIC GROUP
AUTHOR: RAYMOND SHOOP

COPYRIGHT (C) 1973, DIGITAL EQUIPMENT CORP., MAYNARD, MASS.

THIS SOFTWARE IS FURNISHED TO PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DEC'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DEC.

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

1. ABSTRACT

THE DDGTD-B DIAGNOSTIC PROGRAM IS WRITTEN TO BE USED AS AN AID TO HARDWARE DEBUGGING AND MAINTENANCE OF THE GT40 ROM BOOTSTRAP LOADER VERSION 1 OR 2.

THE AVAILABLE TESTS ARE
 PRG0 - LOGIC TESTS
 PRG1 - ROM DATA DUMP TO THE CONSOLE TELETYPE
 PRG2 - SINGLE ROM ADDRESS READ DATA LOOP

2. REQUIREMENTS2.1 EQUIPMENT

GT40 DISPLAY PROCESSOR WITH ROM BOOTSTRAP VERSION 1 OR 2.

2.2 STORAGE

THIS PROGRAM USES MEMORY LOCATIONS 0-7776 + 16000-16776(8).

3. LOADING PROCEDURE

PROCEDURE FOR A NORMAL BINARY TAPE SHOULD BE FOLLOWED.

4. STARTING PROCEDURE4.1 STARTING ADDRESSES

0200 PROGRAM 0, ROM LOGIC TEST.
 0204 PROGRAM 1, ROM DATA DUMP ON CONSOLE TTY.
 0210 PROGRAM 2, SINGLE ROM READ.

4.2 SWITCH SETTINGS

| | |
|-----------------|------------------------------------|
| CONSOLE SW 11=0 | NORMAL RUN (64. INTERACTIONS/TEST) |
| CONSOLE SW 11=1 | SUPPRES SUBPROGRAM INTERACTIONS |
| CONSOLE SW 08=0 | TEST AS VERSION 2 ROM (512. WORDS) |
| CONSOLE SW 08=1 | TEST AS VERSION 1 ROM (256. WORDS) |

5. PROGRAM DESCRIPTIONS

5.1 PRG0 - LOGIC TESTS

THE LOGIC TESTS CONSIST OF 4 ROUTINES TO TEST THE GT40 ROM
BOOTSTRAP LOGIC

5.1.1 ROUTINE DESCRIPTIONS

| ROUTINE | TESTS |
|---------|---|
| T1 | ADDRESSABILITY OF GT40 ROM BOOTSTRAP |
| T2 | DATA RELIABILITY |
| T3 | THAT GT40 ROM BOOTSTRAP TIMES OUT WHEN REFERENCED BY A DATIP BUS CYCLE |
| T4 | THAT DATA READ FROM THE ROM IS CORRECT |

5.2 PRG1 - ROM DATA DUMP

THIS PROGRAM TYPES OUT THE 512./256. WORDS OF ROM DATA ON THE
CONSOLE TELETYPE AND HALTS.

5.3 PRG2 - SINGLE ROM ADDRESS READ DATA LOOP

THIS PROGRAM CONTINUOUSLY READS DATA FROM THE ADDRESS IN THE
CONSOLE SWITCH REGISTER.

6. ERRORS

THE PROGRAM WILL ONLY HALT ON ERROR. THE PROGRAM DOES NOT
CONTAIN FACILITIES FOR REPORTING ERROR CONDITIONS.
TO PLACE THE PROGRAM INTO A SCOPE LOOP, REPLACE THE ERROR
HALT WITH A NOP.

7. EXECUTION TIME

PRG0 TAKES APPROX. 5 SECONDS PER PASS.
PRG1 N/A
PRG2 N/A

```

145
146
147
148
149
150
151
152
153
154 000034 002054
155 000036 000000
156
157 104400
158 177564
159 177566
160 177776
161 177570
162 177570
163 000500
164 000200
165 000200 000137 001024
166 000204 000137 001506
167 000210 000137 001670

```

```

.LIST ME,BIN,SEQ,LD
.ENABL ABS,AMA

:LOAD ADDRESS=0200
:DEPRESS START
:STACK POINTER IS AT 500

.LIST
=34
SCOPEC
0
:EQUATE STATEMENTS
SCOPE=TRAP
TPCSR=177564
TPDBR=177566
PSW=177776
SR=177570
DISPLAY=177570
STKPTR=500
=200
JMP PRMTRS
JMP PRG1
JMP PRG2

;ADDRESS OF DISPLAY REGISTER
;INITIAL STACK SETTING

```

```

168
169      001000      001000
170      001000      166000      ROMADD:  =1000      ; ROM ADDRESS
171      001002      001000      WORDS:  512.      ;256.      ; ROM LENGTH
172      001004      006000      IMAGE:  START      ;STARTA      ; ROM IMAGE
173      001006      172002      DSR:    172002      ; DISPLAY STATUS REGISTER
174      001010      000010      FILLER: 10      ; # OF FILLER CHAR
175      001012      000010      FILCNT: 10
176      001014      000000      ICNT:   0
177      001016      000000      DUMP:   0
178      001020      000000      CHARA:  0
179      001022      000000      TERM:   0
180      001024      012706      000500      PRMTRS: MOV      #STKPTR,%6      ; SET STACK PTR
181      001030      004737      002306      JSR      PC,SWITCH      ; CHECK ROM VERSION
182
183      ;PROGRAM 0 LOGIC TESTS
184
185      001034      005037      001014      PRGO:   CLR      ICNT      ; CLEAR PASS COUNT
186      001040      012706      000500      PRGOR:  MOV      #STKPTR,%6
187      001044      012737      001040      002130      MOV      #PRGOR,RETURN      ; SET RETURN ADDRESS FOR SCOPE
188      001052      013737      001014      177570      MOV      ICNT,%3DISPLAY      ; DISPLAY PASS COUNT
189
190      ;TEST! TEST ABILITY TO REFERENCE ROM WITHOUT TIMING OUT
191
192      001060      013700      001000      T1:     MOV      ROMADD,%0      ; GET ROM ADDRESS
193      001064      013701      001002      MOV      WORDS,%1      ; GET ADDRESS COUNTER
194      001070      012737      001130      000004      MOV      #ERROR1,4      ; SET UP TIME OUT VECTOR
195      001076      011003      MOV      (0),%3      ; REFERENCE
196      001100      005720      T1A:    MOV      (0)+      ; ROM
197      001102      064037      001016      TST      (0)+
198      001106      021010      ADD      -(0),DUMP      ;
199      001110      132020      CMP      (0),(0)      ;
200      001112      164037      001016      BITB    (0)+,(0)+      ;
201      001116      062700      000002      SUB      -(0),DUMP      ;
202      001122      005301      ADD      #2,%0      ; INCREMENT POINTER
203      001124      0013f4      DEC      %1      ; DECREMENT ADDRESS COUNTER
204      001126      000403      BNE     T1A      ; BRANCH IF NOT FINISHED
205      001130      022626      BR      T1B      ; GO TO SCOPE LOOP
206      001132      000000      ERROR1: CMP      (6)+,(6)+      ; REPOSITION STACK
207      001134      000760      HALT      ; ERROR, TIME-OUT ON ROM ADDRESS
208      001136      104400      BR      T1A      ; LOOP ON ERROR
209

```

```

210
211           ;TEST2 TEST THAT ROM DATA CAN BE READ RELIABLY.
212
213 001140 013700 001000      T2:  MOV  ROMADD,%0      ;GET ROM ADDRESS
214 001144 013701 001002      MOV  WORDS,%1      ;GET ADDRESS COUNTER
215 001150 012737 000006 000004  MOV  #6,4        ;INITIALIZE TIME OUT VECTOR
216 001156 005037 001016      T2A: CLR  DUMP      ;INITIALIZE DUMP
217 001162 011003      MOV  (0),%3      ;GET DATA
218 001164 062037 001016      ADD  (0)+,DUMP   ;ADD DATA TO DUMP
219 001170 163703 001016      SUB  DUMP,%3     ;SUBTRACT DATA FROM DATA
220 001174 001402      BEQ  T2B        ;BRANCH IF EQUAL
221 001176 000000      ERROR2: HALT    ;DATA ERROR
222 001200 000766      BR   T2A       ;LOOP ON ERROR
223 001202 044037 001016      T2B: BIC  -(0),DUMP ;CLEAR DUMP BITS
224 001206 001402      BEQ  T2C       ;BRANCH IF EQUAL TO 0
225 001210 000000      HALT    ;DATA ERROR
226 001212 000773      BR   T2B       ;LOOP ON ERROR
227 001214 021010      T2C: CMP  (0),(0) ;COMPARE DATA
228 001216 001402      BEQ  T2D       ;BRANCH IF EQUAL
229 001220 000000      HALT    ;DATA ERROR
230 001222 000774      BR   T2C       ;LOOP ON ERROR
231 001224 122040      T2D: CMPB (0)+,-(0) ;COMPARE DATA (BYTE OPERATION)
232 001226 001402      BEQ  T2E       ;BRANCH IF EQUAL
233 001230 000000      HALT    ;DATA ERROR
234 001232 000774      BR   T2D       ;LOOP ON ERROR
235 001234 005720      T2E: TST  (0)+   ;INCREMENT ADDRESS POINTER
236 001236 005301      DEC  %1        ;DECREMENT ADDRESS COUNTER
237 001240 001346      BNE  T2A       ;RETURN IF NOT DONE
238 001242 104400      SCOPE
239

```

H01

GT40 ROM BOOTSTRAP TEST MAINDEC-11-DDGTD-B
DDGTDB.P11

MACY11 27(732) 09-SEP-76 15:17 PAGE 7

```
240
241 ;TEST3 TEST THAT ROM TIMES OUT IF REFERENCED BY OTHER
242 ;THAN DATI BUS CYCLE
243
244 001244 012706 000500 T3: MOV #STKPTR,%6 ;SET STACK PTR
245 001250 013700 001000 MOV ROMADD,%0 ;GET ROM ADDRESS
246 001254 013701 001002 MOV WORDS,%1 ;GET ADDRESS COUNTER
247 001260 012737 001274 000004 T3AA: MOV #T3B,4 ;SET UP TIME OUT VECTOR
248 001266 010010 T3A: MOV %0,(0) ;ATTEMPT TO ALTER DATA
249 001270 000000 HALT ;HERE IF DID NOT TIME OUT
250 001272 000775 BR T3A ;LOOP ON ERROR
251 001274 012737 001312 000004 T3B: MOV #T3D,4 ;SET UP TIME OUT VECTOR
252 001302 022626 T3C: CMP (6)+,(6)+ ;REPOSITION STACK
253 001304 005210 T3C: INC (0) ;ATTEMPT TO ALTER DATA
254 001306 000000 HALT ;HERE IF DID NOT TIME OUT
255 001310 000775 BR T3C ;LOOP ON ERROR
256 001312 012737 001332 000004 T3D: MOV #T3F,4 ;SET UP TIME OUT VECTOR
257 001320 022626 T3E: CMP (6)+,(6)+ ;REPOSITION STACK
258 001322 005077 177452 T3E: CLR @ROMADD ;ATTEMPT TO ALTER DATA
259 001326 000000 HALT ;HERE IF DID NOT TIME OUT
260 001330 000774 BR T3E ;LOOP ON ERROR
261 001332 005720 T3F: TST (0)+ ;INCREMENT ADDRESS POINTER
262 001334 022626 CMP (6)+,(6)+ ;REPOSITION STACK
263 001336 005301 DEC %1 ;DECREMENT ADDRESS COUNTER
264 001340 001347 BNE T3AA ;RETURN IF NOT DONE
265 001342 012737 000006 000004 MOV #6,@#4 ;RESTORE TIME OUT TRAP
266 001350 104400 SCOPE ;SCOPE LOOP
```



```

267
268
269
270
271
272
273
274
275 001352 012700 000000
276 001356 013701 001004
277 001362 013703 001000
278 001366 011102
279 001370 011304
280 001372 020204
281 001374 001402
282 001376 000000
283 001400 000772
284
285 001402 022123
286 001404 005200
287 001406 023700 001002
288 001412 001365
289 001414 104400
290
291 001416 005237 001014
292 001422 012777 000001 177356
293 001430 012737 000207 177566
294 001436 105737 177564
295 001442 100375
296 001444 012737 000207 177566
297 001452 105737 177564
298 001456 100375
299 001460 013700 000042
300 001464 001406
301 001466 000005
302 001470 000005
303 001472 004710
304 001474 000240
305 001476 000240
306 001500 000240
307 001502 000137 001034
308

;COMPARE THE ROM DATA TO THE IMAGE DATA
;
;RO=WORD NUMBER
;R1=GOOD ADDRESS
;R2=GOOD DATA
;R3=BAD ADDRESS
;R4=BAD DATA

T4:  MOV    #0,%0          ;SET UP INITIAL WORD COUNT
     MOV    IMAGE,%1      ;SET UP STARTING ADDRESS OF ROM IMAGE
     MOV    ROMADD,%3     ;SET UP STARTING ROM ADDRESS
T4A:  MOV    (%1),%2       ;READ EXPECTED VALUE
     MOV    (%3),%4       ;READ ROM VALUE
     CMP    %2,%4         ;COMPARE EXPECTED TO THE VALUE READ
     BEQ    T4B           ;BRANCH IF CORRECT
     HALT                    ;ERROR, ROM VALUE FAILED TO EQUAL EXPECTED
     BR     T4A

T4B:  CMP    (%1)+,(%3)+  ;INCREMENT ADDRESSES POINTERS
     INC    %0            ;INCREMENT WORD COUNT
     CMP    WORDS,%0     ;COMPARE IF END WORD
     BNE    T4A          ;BRANCH IF NOT LAST WORD

T4E:  SCOPE

END:  INC    ICNT        ;INCREMENT PASS COUNT
     MOV    #1,%DSR     ;RING THE GT40 BELL
DONE0: MOV    #207,%#TPDDBR ;RING THE TELETYPE BELL
     TSTB  %#TPCSR
     BPL   .-4
     MOV   #207,%#TPDDBR
1$:  TSTB  %#TPCSR
     BPL   1$
     MOV   %#42,%0
     BEQ   DONE1
     RESET
     RESET
303: JSR   7,(0)          ;RETURN!
     NOP
     NOP
     NOP
DONE1: JMP   PRGO

```

```

309
310           ;THIS PROGRAM TYPES OUT ROM DATA
311
312 001506 012706 003500          PRG1:  MOV    #STKPTR,%6      ;INITIALIZE STACK
313 001512 012737 000006 000004  MOV    #6,%4          ;SET UP BUSS ERROR
314 001520 004737 002306          JSR    PC,SWITCH
315 001524 004537 001712          JSR    5,TYPEN
316 001530 002275          MB
317 001532 004537 001712          JSR    5,TYPEN          ;TYPE MESSAGE
318 001536 002252          M7          ;'ROM DATA'
319 001540 013701 001002          MOV    WORDS,%1      ;GET # OF WORDS
320 001544 013700 001000  PRG1A:  MOV    ROMADD,%0      ;GET STARTING ADDRESS
321 001550 012702 000010          MOV    #10,%2       ;GET ADDRESS INDICATOR
322 001554 105737 177564          TSTB  TPCSR         ;WAIT FOR
323 001560 100375          BPL   -.4           ;TELEPRINTER FLAG
324 001562 010037 002144  PRG1B:  MOV    %0,D2BTYP    ;GET ADDRESS
325 001566 004737 002146          JSR    7,02A        ;AND TYPE IT
326 001572 004537 001712          JSR    5,TYPEN      ;TYPE
327 001576 002301          M9          ;CR/LF
328 001600 012037 002144  PRG1C:  MOV    (0)+,D2BTYP  ;TYPE
329 001604 004737 002146          JSR    7,02A        ;DATA
330 001610 105737 177564          TSTB  TPCSR         ;WAIT FOR
331 001614 100375          BPL   -.4           ;TELEPRINTER FLAG
332 001616 012737 000040 177566  MOV    #' ',TPDBR   ;TYPE SPACE
333 001624 005301          DEC   %1           ;ALL DATA TYPED
334 001626 001410          BEQ   PRG1D        ;GO TO FINISH
335 001630 005302          DEC   %2
336 001632 001362          BNE   PRG1C        ;RETURN TO PRG1B
337 001634 012702 000010          MOV    #10,%2       ;GET ADDRESS INDICATOR
338 001640 004537 001712          JSR    5,TYPEN      ;TYPE
339 001644 002275          M9          ;CR/LF
340 001646 000745          BR    PRG1B        ;RETURN TO PRG1B
341 001650 004537 001712  PRG1D:  JSR    5,TYPEN
342 001654 002275          MB
343 001656 004537 001712          JSR    5,TYPEN
344 001662 002275          MB
345 001664 000000          HALT
346 001666 000707          BR    PRG1
347
348           ;ROUTINE TO LOOP ON A SINGLE ADDRESS
349
350 001670 012706 000500          PRG2:  MOV    #STKPTR,%6
351 001674 012737 000006 000004  MOV    #6,%4
352 001702 013700 177570          MOV    SR,%0
353 001706 011001          MOV    (0),%1
354 001710 000767          BR    PRG2

```

```

355 ;ROUTINE TO TYPE A MESSAGE
356
357 001712 010026 TYPEM: MOV %0,(6)+ ;SAVE REGISTER 0
358 001714 012500 MOV (5)+,%0 ;PLACE MESSAGE ADDRESS IN R0
359 001716 112037 001022 MOV (0)+,TERM ;GET TERMINATOR CHARACTER
360 001722 112037 001020 TYPEMA: MOV (0)+,CHARA ;GET NEXT CHARACTER
361 001726 123737 001020 001022 CMPB CHARA,TERM ;WAS NEXT CHARACTER THE TERM
362 001734 001005 BNE TYPEMB ;CHARACTER
363 001736 014600 MOV -(5),%0 ;RESTORE R0
364 001740 105737 177564 TSTB TPCSR
365 001744 100375 BPL -4
366 001746 000205 RTS 5 ;AND EXIT
367 001750 123727 001020 000045 TYPEMB: CMPB CHARA,#'% ;WAS CHARACTER %
368 001756 001027 BNE TYPEMC
369 001760 105737 177564 TSTB TPCSR ;TEST TELEPRINTER FLAG
370 001764 100375 BPL -4 ;AND WAIT FOR DONE
371 001766 012737 000215 177566 MOV #215,TPDBR ;LOAD TELEPRINTER WITH CAR. RET
372 001774 013737 001010 001012 MOV FILLER,FILCNT ;LOAD FILLER COUNT
373 002002 000403 BR 1$
374 002004 012737 000006 177566 2$: MOV #6,TPDBR ;PRINT FILLER CHAR
375 002012 105737 177564 1$: TSTB TPCSR ;TEST TELEPRINTER FLAG
376 002016 100375 BPL -4 ;AND WAIT FOR DONE
377 002020 005337 001012 DEC FILCNT ;FINISHED FILLERS ?
378 002024 001367 BNE 2$ ;BR IF NOT
379 002026 012737 000212 177566 MOV #212,TPDBR ;LOAD TELEPRINTER WITH LINE FEED
380 002034 000732 BR TYPEMA ;GET NEXT CHARACTER
381 002036 105737 177564 TYPEMC: TSTB TPCSR ;TEST TELEPRINTER FLAG
382 002042 100375 BPL -4 ;AND WAIT FOR DONE
383 002044 013737 001020 177566 MOV CHARA,TPDBR ;LOAD TELEPRINTER BUFFER
384 002052 000723 BR TYPEMA ;AND GET NEXT CHARACTER
385
386 ;SCOPE ROUTINE. THIS ROUTINE IS ENTERED AT THE END OF EACH SUBTEST.
387
388 002054 032737 040000 177570 SCOPEC: BIT #40000,SR ;TEST SR FOR SCOPE
389 002062 001023 BNE SCOPEB ;YES SCOPE
390 002064 032737 004000 177570 BIT #4000,SR ;TEST FOR ITERATION
391 002072 001007 BNE SCOPEB ;INHIBIT ITERATION
392 002074 023737 002126 002124 CMP SCOPEF,ICOUNT ;ITERATION COMPLETE
393 002102 001403 BEQ SCOPEG ;ITERATION COMPLETE GO TO SCOPEG
394 002104 005237 002126 INC SCOPEF ;INCREMENT ITERATION COUNT
395 002110 000410 BR SCOPEB ;GO TO SCOPEB
396 002112 005037 002126 SCOPEG: CLR SCOPEF ;CLEAR ITERATION COUNT
397 002116 011637 002130 MOV @%6,RETURN ;GET ADDRESS OF NEXT TEST
398 002122 000002 RTI ;EXIT
399 002124 000100 ICOUNT: 100
400 002126 000000 SCOPEF: 0 ;CONTAINS SUBTEST ITERATION COUNT
401 002130 000000 RETURN: .WORD 0 ;CONTAINS RETURN PC FOR SCOPE
402 002132 005726 SCOPEB: TST (6)+ ;POP PC
403 002134 012637 177776 MOV (6)+,PSW ;RESTORE CONDITION CODES
404 002140 000177 177764 JMP @RETURN
405

```

```

406
407
408 ;THIS ROUTINE CONVERTS AN OCTAL NUMBER TO ASCII AND TYPES IT ON THE TTY.
409
410 002144 000000 D2BTYP: 0
411 002146 013746 177564 02A: MOV TPCSR, -(6) ;SAVE TPCSR
412 002152 010246 MOV %2, -(6) ;SAVE R2
413 002154 010146 MOV %1, -(6) ;SAVE R1
414 002156 010046 MOV %0, -(6) ;SAVE R0
415 002160 013700 002144 MOV D2BTYP, %0 ;GET DATA TO BE TYPED
416 002164 012701 000006 MOV #6, %1 ;GET COUNTER
417 002170 005002 CLR %2 ;CLEAR WORKING REGISTER
418 002172 006100 ROL %0 ;MOV FIRST BIT (MSB) INTO
419 002174 006102 ROL %2 ;R2
420 002176 062702 000260 02AA: ADD #260, %2 ;FORM ASCII CODE
421 002202 105737 177564 TSTB TPCSR ;TEST TELEPRINTER
422 002206 100375 BPL -4 ;FLAG AND WAIT UNTIL DONE
423 002210 010237 177566 MOV %2, TPDBR ;LOAD TELEPRINTER BUFFER
424 002214 005002 CLR %2 ;CLEAR WORKING REGISTER
425 002216 006100 ROL %0 ;ROTATE THE
426 002220 006102 ROL %2 ;NEXT
427 002222 006100 ROL %0 ;OCTAL CHARACTER
428 002224 006102 ROL %2 ;INTO
429 002226 006100 ROL %0 ;REGISTER
430 002230 006102 ROL %2 ;TWO
431 002232 005301 DEC %1 ;DECREMENT COUNTER
432 002234 001360 BNE 02AA ;GO TO 02AA IF NOT 0
433 002236 012600 MOV (6)+, %0 ;FINISHED. RESTORE REGISTERS
434 002240 012601 MOV (6)+, %1
435 002242 012602 MOV (6)+, %2
436 002244 012637 177564 MOV (6)+, TPCSR ;AND TPCSR
437 002250 000207 RTS ;AND EXIT
438
439 ;ASCII MESSAGES
440 002252 022500 052107 032055 M7: .ASCII 'a%GT-40 ROM DATA%a'
441 002260 020060 047522 020115
442 002266 040504 040524 022445
443 002274 100
444 002275 100 022445 100 M8: .ASCII 'a%a'
445 002301 100 020040 100 M9: .ASCII 'a a'
446 002306 .EVEN
447
448 002306 032737 000400 177570 SWITCH: BIT #400, SR ;TEST BIT 8
449 002314 001007 BNE 1$ ;BR IF VERSION 1
450 002316 012737 001000 001002 MOV #512, WORDS ;SET UP VERSION 2 LENGTH
451 002324 012737 006000 001004 MOV #START, IMAGE ;SET UP VERSION 2 STARTING ADD.
452 002332 000406 BR 2$
453 002334 012737 000400 001002 1$: MOV #256, WORDS ;SET UP VERSION 1 LENGTH
454 002342 012737 016000 001004 2$: MOV #START, IMAGE ;SET UP VERSION 1 STARTING ADD.
455 002350 000207 RTS PC
456
457 .SBTTL ROM VERSION 2 VALUES

```


458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511

```
*****  
: EXCEPT FOR THE NEW ORGIN ADDRESS AND SEVERAL "!160000"  
: FOR ADDRESS FUDGING THIS IS AN EXACT COPY OF THE CONTENTS  
: OF THE GT-40 BOOTSTRAP VERSION #2  
*****
```

.TITLE SCROLLING ROM BOOTSTRAP FOR THE GT40

```
;  
; BOOTGT.T16 OCT 10, 1973
```

```
;  
: COPYRIGHT 1973, DIGITAL EQUIPMENT CORPORATION  
: 146 MAIN STREET  
: MAYNARD, MASSACHUSETTS 01754
```

```
;  
; WRITTEN BY JACK BURNES.
```

```
;  
: THIS PROGRAM IS THE SECOND VERSION THE THE ROM BOOTSTRAP FOR  
: THE GT40 DISPLAY TERMINAL. IT INCLUDES SCROLLING AND AN END OF  
: MEMORY SEARCH FOR THE LOADER.
```

```
.ENABL ABS,AMA ;ASSEMBLER DIRECTIVES FOR ABSOLUTE BINARY OUTPUT  
; NOTE: USE "MACDLX" TO ASSEMBLE THIS PROGRAM.
```

```
.SBTTL DEFINITION SECTION  
.PAGE
```

REGISTER DEFINITIONS

BASIC DEFINITIONS

S12
S13
S14
S15
S16
S17
S18
S19
S20
S21
S22
S23
S24
S25
S26
S27
S28
S29
S30
S31
S32
S33
S34
S35
S36
S37
S38
S39
S40
S41
S42
S43
S44
S45
S46
S47
S48
S49
S50
S51
S52
S53
S54
S55
S56
S57
S58
S59
S60
S61
S62
S63
S64
S65

R0=%0
R1=%1
R2=%2
R3=%3
R4=%4
R5=%5
SP=%6
PC=%7

;DEFINE STANDARD VALUES.

GT40 DEFINITIONS

000000
000001
000002
000003

000004

000005

CHAR=R0
POINTR=R1
TABCNT=R2
SCAN=R3

HOLD=R4

COUNTR=R5

;CONTAINS THE INPUT CHARACTER.
;POINTS TO NEXT INSERTION BYTE IN DISPLAY BUFFER
;CHARACTER COUNTER FOR THE "TAB" FEATURE.
;GENERALLY CONTAINS A POINTER WHICH
;IS USED WHEN SCANNING MEMORY FOR SOMETHING.
;TYPICALLY A TEMPORARY WHICH IS USED TO RETAIN
;A VALUE FOR A SHORT TIME.
;TYPICALLY USED AS A COUNTER.

LOADER DEFINITIONS

000000
000001
000002
000005
000003

L.BYT=CHAR
L.ADR=POINTR
L.ZC=TABCNT
L.CKSM=COUNTR
INDEX=SCAN

;CHARACTER INPUT FOR THE LOADER.
;CURRENT MEMORY ADDRESS TO BE LOADED.
;NUMBER OF DATA ITEMS TO LOAD.
;CHECKSUM ON THE INPUT DATA.
;INDICATES HOW TO ASSEMBLE THE 8 BIT CHARACTER.

619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674

GT40 BOOTSTRAP CODE

006000 .=6000
 .=ORIGIN ;DEFINE ORIGIN OF THE BOOTSTRAP.

COLD INITIALIZATION CODE

| | | | | | | |
|--------|--------|--------|---------|---------|-------------------------|--|
| 006000 | 000005 | | START: | RESET | | ;RESET ALL HARDWARE NOW. |
| 005002 | 012737 | 000007 | | MOV | #7,DL11IS | ;INITIALIZE DL-11 INPUT NOW. |
| 006010 | 012706 | 007776 | 175610 | MOV | #TMPEND,SP | ;ESTABLISH A GOOD TEMPORARY STACK |
| 006014 | 005237 | 175614 | | INC | DL1105 | ; POINTER FOR CORE SEARCH. |
| 006020 | 004337 | 166652 | | JSR | SCAN,OUTLIT!160000 | ;SET BREAK BIT |
| 006024 | 000000 | | | .WORD 0 | | ;FOR 2 CHARACTER TIMES |
| | | | | | | ;SEND TWO ZERO'S |
| 006026 | 012703 | 000004 | | MOV | #CORSTR,SCAN | ;GET ADDRESS OF BAD CORE TRAP VECTOR. |
| 006032 | 012723 | 166042 | | MOV | #NOTHERE!160000,(SCAN)+ | ;AND INSERT A POINTER TO US THERE. |
| 006036 | 005023 | | ENDCOR: | CLR | (SCAN)+ | ;NOW CLEAR ALL OF MEMORY BEYOND THE POINTER, |
| 006040 | 000776 | | | BR | ENDCOR | ;UNTIL WE RUN OUT OF MEMORY AND TRAP. |
| 006042 | 005743 | | NOTHER: | TST | -(SCAN) | ;WHEN WE TRAP OUT, WE COME HERE. |
| | | | | | | ;WE BACK UP POINTER TO GOOD CORE. |
| | | | | | | ;NOTE THAT IF WE TRAP OUT AGAIN, IT |
| | | | | | | ;IS STILL OK, BECAUSE WE WILL LOOP |
| | | | | | | ;UNTIL WE GET A GOOD CORE ADDRESS. |
| 006044 | 010306 | | | MOV | SCAN,SP | ;WHEN WE GET ONE, THAT IS LAST LOCATION |
| | | | | | | ;IN THE MACHINE, AND HENCE OUR SP. |
| 006046 | 105737 | 175614 | IS: | TSTB | DL1105 | ;SEE IF BREAK IS DONE |
| 006052 | 100375 | | | BPL | IS | ;NO GO BACK |
| 006054 | 005037 | 175614 | | CLR | DL1105 | ;CLEAR BREAK BIT |

RESTART INITIALIZATION CODE WHEN COMMUNICATIONS IS WORKING.

675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715

```

006060 052706 007776      RESTR: BIS      #TMPEND,SP      ;FORCE THE SP TO LIMIT OF EXISTING CORE.

006064 012703 006700      MOV      #BLIMIT-NUMLIN-NUMLIN,SCAN      ;NOW WE WILL FILL THE KEY AREAS OF THE
006070 012702 000040      MOV      #NUMLIN,TABCNT      ;DISPLAY BUFFER WITH INITIAL CR-LF'S.

006074 012723 005015      SETLP1: MOV     #CRLF,(SCAN)+      ;INSERT A CRLF NOW.
006100 005302      DEC     TABCNT      ;AND LOOP UNTIL DONE.
006102 003374      BGT     SETLP1      ;THUS DISPLAY CORE IS ALMOST CORRECT.

006104 012703 165432      MOV      #SETUP!160000,SCAN      ;NOW WE WILL INITIALIZE CORE FOR THE
;DISPLAY. PICK UP POINTER TO LIST.

006110 012302      SETLP2: MOV     (SCAN)+,TABCNT      ;GET NUMBER OF ITEMS TO INSERT.
006112 001405      BEQ     SETDUN      ;IF ZERO, WE ARE DONE.
006114 012301      MOV     (SCAN)+,POINTR      ;PICK UP FIRST CORE ADDRESS POINTER.

006116 012321      SETLP3: MOV     (SCAN)+,(POINTR)+      ;MOVE OVER A DATA ITEM NOW.
006120 005302      DEC     TABCNT      ;ALL DONE?
006122 003375      BGT     SETLP3      ;NOPE. MOVE OVER THE NEXT.
006124 000771      BR      SETLP2      ;YES. GET NEXT MAJOR LIST TO INSERT.

006126 012701 006776      SETDUN: MOV     #BLIMIT-2,POINTR      ;ESTABLISH THE BUFFER POINTER NOW.

```

716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771

VTOS (SCROLLING) PORTION OF THE BOOTSTRAP

```

NXTCHR: JSR PC GETCHR!160000
          CMP CHAR,#177
          BGE NXTCHR
          CMP CHAR,#40
          SGE NORMAL
          MOV CHAR,SCAN
          SUB #7,SCAN
          CMP SCAN,#7
          BHS NXTCHR
          ASL SCAN
          ADD SCAN,PC

          BR BELL
          BR NORMAL
          BR TAB
          BR LF
          BR VT
          BR FF

CR:      MOV #-1,TABCNT

NORMAL: JSR PC INSERT!160000
          INC TABCNT
          BR NXTCHR

TAB:     MOV #40,CHAR
          JSR PC INSERT!160000
          INC TABCNT
          BIT #7,TABCNT
          BNE TAB
          BR NXTCHR

VT:     MOVB (PC),COUNTR

          BR FFLOOP

BELL:   CLR GT40SR
          BR NXTCHR

FF:     MOV #NUMLIN,COUNTR
    
```

```

;GET A CHARACTER NOW.
;IS IT OUT OF RANGE?
;YEP. GET ANOTHER ONE.
;IS IT A PRINTING CHARACTER?
;YES. IT'S A NORMAL PRINTING CHARACTER.
;MOVE IT OVER SO WE CAN PLAY WITH IT.
;BIAS SO THAT BELL [7] IS ZERO.
;IF CHARACTER IS LESS THEN BELL OR
;GREATER THEN CR, THEN IGNORE.
;IF GOOD, MAKE IT WORD INDEX.
;AND GO TO THE CORRECT ROUTINE.

;7=BELL
;10=BACKSPACE
;11=TAB
;12=LINE FEED [LF]
;13=VERTICAL TAB [VT]
;14=FORM FEED [FF]
;15=CARRIAGE RETURN [CR]

;RESET TAB POSITION ON A CR, AND
;FALL THROUGH TO INSERT THE CHARACTER.

;INSERT THE CHARACTER IN THE BUFFER.
;UPDATE TAB POSITION NOW.
;AND GET NEXT CHARACTER.

;ON A TAB, INSERT BLANKS UNTIL THE
;NEXT CHARACTER POSITION IS A MULTIPLE
;OF 8.
;ARE WE DONE YET?
;NOPE.
;YES.

;THIS PUTS THE LOW BYTE OF THE
;BRANCH CODE IN COUNTR-SAVE A WORD

;RING BELL -WRITE IN GT40SR
;AND LOOP BACK

;FORM FEED IS DONE BY INSERTING LF'S.
    
```

```

772
773 006262 012700 000012      FFLOOP: MOV      #12, CHAR      ;MAKE THE CHARACTER A LINEFEED.
774 006266 004737 166304      JSR      PC, LFSUB!160000    ;DO A LINEFEED.
775 006272 005305              DEC      COUNTR              ;DONE?
776 006274 003372              BGT      FFLOOP              ;NOPE. KEEP SENDING THEM.
777 006276 000715              BR       NXTCHR              ;YES. NOW RETURN. DO NOT FALL THROUGH.
778
779
780 006300 012746 166132      LF:      MOV      #NXTCHR!160000, -(SP) ;RETURN TO NXTCHR AFTER PROCESSING
781                                         ;THE LF BY FAKING A JSR.
782
783 006304 013703 007012      LFSUB:  MOV      JMPADD, SCAN    ;GET POINTER TO FIRST CHAR ON SCREEN
784
785 006310 122300              LFLOOP: CMPB     (SCAN)+, CHAR    ;AND LOOK FOR A LINEFEED.
786 006312 001406              BEQ      LFOUND              ;GOT IT. SEARCH HAS ENDED.
787 006314 020327 007000      CMP      SCAN, #BLIMIT        ;ARE WE AT END OF BUFFER?
788 006320 103773              BLO      LFLOOP              ;NOPE. KEEP ON LOOKING.
789 006322 012703 001000      MOV      #BSTART, SCAN        ;IF AT TOP, RESET TO BOTTOM OF BUFFER
790 006326 000770              BR       LFLOOP              ;AND KEEP ON LOOKING.
791
792 006330 005203              LFOUND: INC      SCAN          ;WE'VE GOT THE LINE FEED. STOP SHOWING
793 006332 042703 000001      BIC      #1, SCAN            ;FIRST LINE BY CHANGING THE "DISJMP"
794 006336 010337 007012      MOV      SCAN, JMPADD         ;INSTRUCTION TO FIRST CHAR BEYOND LF.
795 006342 004737 166350      JSR      PC, INSERT!160000    ;INSERT THE LF IN THE BUFFER.
796 006346 005000              CLR      CHAR                ;AND THEN INSERT ONE NULL CHARACTER BECAUSE
797                                         ;THE "DISJMP" ADDRESS MUST BE EVEN, AND
798                                         ;THIS GUARANTEES WE WILL NOT LOSE A
799                                         ;A GOOD DATA CHARACTER. WE FALL THROUGH
800                                         ;TO INSERT THE NULL IN THE BUFFER.
801
802
803 006350 110021              INSERT: MOVB     CHAR, (POINTR)+ ;STICK IN THE CHARACTER NOW.
804 006352 032701 000001      BIT      #1, POINTR          ;IS NEXT POSITION EVEN OR ODD?
805 006356 001021              BNE      INSRTX              ;ODD. NO PROBLEMS. SPACE IS ALLOCATED.
806 006360 020127 007000      CMP      POINTR, #BLIMIT     ;EVEN. ARE WE AT THE END OF THE BUFFER?
807 006364 103410              BLO      INSRTL              ;NO. JUST MAKE ROOM FOR ANOTHER WORD.
808 006366 010103              MOV      POINTR, SCAN        ;AT THE END. MOVE THE STUFF TO THE
809 006370 012701 001000      MOV      #BSTART, POINTR     ;BEGINNING OF THE BUFFER.
810 006374 004737 166406      JSR      PC, INSRTL!160000    ;CALL THE ROUTINE TO SAVE SPACE.
811 006400 005023              CLR      (SCAN)+            ;AND CLEAR UP THE INSTRUCTIONS AT THE
812 006402 005013              CLR      (SCAN)             ;END OF THE BUFFER.
813 006404 000207              RTS      PC                  ;AND THEN RETURN.
814
815 006406 022121              INSRTL: CMP      (POINTR)+, (POINTR)+ ;BYPASS THE "DISJMP" BY ADDING 4 TO POINTR.
816 006410 012711 166474      MOV      #HEADER!160000, (POINTR) ;NOW INSERT THE DISJMP INSTRUCTION TO OUR HEADER
817 006414 012741 160000      MOV      #DISJMP, -(POINTR) ;AND IT'S ADDRESS (PUT THEM IN BACKWARDS).
818 006420 005041              CLR      -(POINTR)          ;MAKE AVAILABLE A NEW CHARACTER SPOT.
819
820 006422 000207              INSRTX: RTS      PC          ;FINALLY RETURN TO THE CALLER.
821
822
823
824
825
826 006424 012737 001000 172000 GTBUSE: MOV      #BSTART, GT40PC ;ON A BUS ERROR, WE MERELY RESTART THE GT40 AT
827

```

;THE RTI FOR THIS ROUTINE
;IS THE FIRST WORD OF THE TABLE
;BELOW-IT SAVES A WORD!

828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876

INITIALIZATION TABLE FOR THE SCROLLER

;

```

SETUP: .WORD 2 ;INITIALIZE 2 WORDS.--ALSO RTI FROM ABOVE
        .WORD 330 ;STARTING AT LOCATION 330
        .WORD GTBUSE!160000 ;FIRST WORD IS POINTER TO BUS ERROR ROUT
        .WORD 200 ;SECOND WORD IS NEW STATUS WORD ON INTERRUPT.

        .WORD 7 ;INITIALIZE THE END OF THE BUFFER TO
        .WORD BLIMIT-2 ;A CLEAR SPACE TO INSERT THE CHARACTER.
        .WORD 0 ;THIS IS THE "RUNNING" START. THIS IS
        .WORD DISJMP,HEADER!160000 ;FOLLOWED BY A DISJMP TO OUR HEADER BLOC
        .WORD DISJMP,BSTART ;AND THEN A DISJMP TO THE START OF THE BUFFER
        .WORD DISJMP,BLIMIT-NUMLIN-NUMLIN ;AND A DISJMP TO THE FIRST CHAR ON SCREE

        .WORD 1 ;FINALLY START THE GT40 GOING AT
        .WORD GT40PC ;THE POSITION INSTRUCTION IN THE
        .WORD HEADER!160000 ;HEADER BLOCK.

        .WORD 0 ;END OF INIT CODE
    
```

;

HEADER BLOCK FOR THE SCROLLER

```

HEADER: .WORD 103334 ;ENABL CHAR MODE,BLINKING
        .WORD 177 ;A BLINKING BOX-RUB OUT!
        .WORD 116!24 ;GO TO POINT MODE
        .WORD 171340 ;LOAD STATUS REGISTER
        .WORD 0,1352 ;POINT TO UPPER LEFT
        .WORD 103324 ;BACK TO CHAR MODE
        .WORD DISJMP,JMPADD-2 ;AND TO THE CHANGING JMP INST.
    
```

.SBTTL COMMUNICATIONS AND MISC. SUPPORT ROUTINES
.PAGE

877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932

COMMUNICATIONS HANDLING ROUTINES

THE DL-11 HANDLER

```

006516 105737 175610 GETDL: TSTB DL11IS ;CHECK THE HOST INPUT STATUS.
006522 100011 BPL GETDL1 ;HOST DID NOT SEND ANYTHING, YET.
006524 113700 175612 MOVB DL11IB,CHAR ;HOST SENT US A CHARACTER. PROCESS IT.
006530 012737 000007 175610 MOV #7,DL11IS ;REENABLE THE HOST TELECOMMUNICATIONS.
006536 042700 177600 BIC #-200,CHAR ;MAKE CHARACTER JUST SEVEN BITS.
006542 001765 BEQ GETDL ;IF NULL, IGNORE IT.
006544 000207 RTS PC ;ELSE RETURN NOW.

006546 105737 177560 GETDL1: TSTB KBDIS ;DID USER TYPE A CHARACTER?
006552 100361 BPL GETDL ;NO. GO BACK AND CHECK HOST MACHINE.
006554 113737 177562 175616 MOVB KBDIB,DL110B ;MOVE THE CHARACTER TO THE HOST.
006562 000755 BR GETDL ;AND CHECK AGAIN FOR INPUT.
  
```

THE "GET CHARACTER" ROUTINE

```

006564 004737 166516 GETCHR: JSR PC,GETDL!160000 ;GET A CHARACTER FROM THE HOST NOW.
006570 020027 000175 CMP CHAR,#ALTMOD ;IS IT AN "ALTMODE"
006574 001025 BNE GETEXT ;NO. EXIT NOW.

006576 004737 166516 JSR PC,GETDL!160000 ;YES. GET ANOTHER ONE NOW.
006602 020027 000114 CMP CHAR,#'L ;IS IT AN "L"
006606 001501 BEQ LOADER ;YES. START LOADING NOW.
006610 020027 000122 CMP CHAR,#'R ;IS IT AN "R"
006614 001015 BNE GETEXT ;NO. IGNORE THE ALTMODE AND JUST RETURN THE CHAR

006616 012737 173000 007010 PRESTR: MOV #DISTOP JMPADD-2 ;YES. RESET. STOP DISPLAY BY INSERTING A "DSTOP
006624 000137 166060 JMP RESTR!160000 ;INSTRUCTION IN THE BUFFER, AND RESTART.
  
```

THE "GET A SIX BIT CHARACTER" ROUTINE

```

933 ; -----
934 ;
935 ;
936 ;
937 006630 004737 166564 GETSIX: JSR PC,GETCHR!160000 ;GET A CHARACTER NOW.
938 006634 020027 000040 CMP CHAR,#40 ;IS IT A LEGAL PRINTING CHARACTER?
939 006640 002517 BLT L.BAD ;NOPE. ABORT
940 006642 020027 000137 CMP CHAR,#137 ;IT'S BIG ENOUGH. IS IT TOO BIG?
941 006646 003114 BGT L.BAD ;YEP. ABORT.
942 ;
943 006650 000207 GETEXT: RTS PC ;RETURN TO THE CALLER.
944 ;
945 ;
946 ; THIS OUTPUTS TWO CHARACTERS VIA A
947 ; JSR SCAN,OUTLIT
948 ; 'TWO CHARACTERS'
949 ;
950 006652 112337 175616 OUTLIT: MOVB (SCAN)+,DL110B
951 006656 112337 175616 MOVB (SCAN)+,DL110B ;DOUBLE BUFFERED
952 006662 000203 RTS SCAN ;RETURN
953 ;
954 ;
955 ;
956 ;
957 ;
958 ;
959 ;
960 ; THE "GET AN EIGHT BIT CHARACTER" ROUTINE
961 ; -----
962 ;
963 ;
964 ;
965 ; THIS ROUTINE DIFFERS FROM THE PREVIOUS ROUTINES
966 ; IN THAT IT WILL TAKE SIX BIT CHARACTERS AND ASSEMBLE
967 ; THEM FOR THE LOADER TO USE. NOTE THAT FROM THIS POINT
968 ; ON WE WILL SWITCH TO THE LOADER DEFINITIONS OF THE
969 ; REGISTERS. THUS THE CHARACTER IS RETURNED IN
970 ; REGISTER "L.BYT" RATHER THAN CHAR (THOUGH THEY ARE
971 ; PHYSICALLY THE SAME).
972 ;
973 ;
974 ;
975 006664 004737 166630 GET8: JSR PC,GETSIX!160000 ;GET A SIXBIT CHARACTER.
976 006670 010046 MOV L.BYT,-(SP) ;SAVE IT ON THE STACK.
977 006672 005723 TST (INDEX)+ ;UPDATE INDEX TO NEXT ITEM (ALL ARE *2)
978 006674 000163 166676 JMP GET8TB-2!160000(INDEX) ;AND DISPATCH ACCORDING TO THE INDEX.
979 ;
980 006700 000404 GET8TB: BR GET81 ;INDEX=2: ASSEMBLE FIRST CHAR
981 006702 000416 BR GET82 ;INDEX=4: ASSEMBLE SECOND CHAR
982 006704 000432 BR GET83 ;INDEX=6: ASSEMBLE THIRD AND LAST CHAR
983 ;INDEX=8: RESET INDEX TO 0 [2] AND RETRY.
984 ;
985 ;
986 006706 012703 000002 GET84: MOV #2,INDEX ;THE FOURTH INDEX IS THE SAME AS THE FIRST
987 ;INDEX. JUST RESET IT AND FALL THROUGH.
988 ;
  
```

```

989
990 006712 004737 166630      GET81: JSR      PC,GETSIX!160000      ;GET ANOTHER CHARACTER NOW.
991 006716 010004              MOV      L.BYT,HOLD      ;AND PRESERVE IT FOR NEXT TIME THROUGH.
992 006720 006300              ASL      L.BYT          ;NOW THROW AWAY LEFT MOST BITS OF
993 006722 006300              ASL      L.BYT          ;THE 8 BIT CHARACTER. NOW MERGE IN
994 006724 106300              ASLB     L.BYT          ;THE LEFT TWO BITS OF THE
995 006726 106116              ROLB     (SP)          ;NEW SIX BIT CHARACTER WITH THE SIX
996 006730 106300              ASLB     L.BYT          ;BITS FROM THE CHARACTER ON THE
997 006732 106116              ROLB     (SP)          ;STACK. 1ST CHARACTER IS NOW ASSEMBLED,
998 006734 012600              MOV      (SP)+,L.BYT    ;SO WE'LL RETURN IT TO THE USER.
999 006736 000207              RTS      PC             ;AND THEN WE SHALL RETURN TO HIM.
1000
1001
1002 006740 006300      GET82: ASL      L.BYT          ;THE SECOND CHARACTER IS CREATED FROM
1003 006742 006300              ASL      L.BYT          ;THE 4 RIGHT BITS OF THE PREVIOUS CHARACTER
1004 006744 106300              ASLB     L.BYT          ;AND THE FOUR MIDDLE BITS OF THE PRESENT
1005 006746 106104              ROLB     HOLD          ;8 BIT CHARACTER.
1006 006750 106300              ASLB     L.BYT          ;WE WILL CREATE THE NEW 8 BIT
1007 006752 106104              ROLB     HOLD          ;IN THIS REGISTER, SINCE IT
1008 006754 106300              ASLB     L.BYT          ;MORE CONVIENT. WE WILL MOVE OVER THE
1009 006756 106104              ROLB     HOLD          ;ANSWER AT THE END.
1010 006760 106300              ASLB     L.BYT          ;ONE MORE TO GO
1011 006762 106104              ROLB     HOLD          ;DONE.
1012 006764 010400              MOV      HOLD,L.BYT    ;BRING OVER THE VALUE.
1013 006766 012604              MOV      (SP)+,HOLD    ;AND REMEMBER THE LAST CHARACTER WE RECEIVED.
1014 006770 000207              RTS      PC             ;AND RETURN TO THE CALLER.
1015
1016
1017 006772 006100      GET83: ROL      L.BYT          ;FINAL CHARACTER IS EASY. JUST A
1018 006774 106100              ROLB     L.BYT          ;SIMPLE MERGER OF LEFT TWO BITS OF
1019 006776 006004              ROR      HOLD          ;PREVIOUS VALUE WITH RIGHT SIX BITS
1020 007000 106000              RORB     L.BYT          ;OF LAST (4TH) CHARACTER RECEIVED.
1021 007002 006004              ROR      HOLD          ;
1022 007004 106000              RORB     L.BYT          ;AND WE ARE DONE.
1023 007006 005726              TST      (SP)+         ;FINALLY THROW AWAY STACK.
1024 007010 000207              RTS      PC             ;AND RETURN TO THE CALLER.
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038

```

.S3TTL THE LOADER
 .PAGE

```

1039
1040
1041
1042          :          THE LOADER
1043          :          ---
1044
1045
1046
1047
1048 007012 012737 173000 007010 LOADER: MOV      #DISTOP, JMPADD-2      ;STOP THE GT40 BY INSERTING A "DISTOP" IN THE LI
1049
1050 007020 005003                CLR      INDEX                ;RESET THE 8 BIT ASSEMBLER TO THE FIRST CHAR
1051
1052
1053 007022 005005                L.LD2: CLR      L.CKSM                ;CLEAR THE CHECKSUM
1054 007024 004737 167114        JSR      PC, L.PTR!160000        ;GET A BYTE NOW.
1055 007030 105300                DECB    L.BYT                ;IS IT A ONE (HEADER)?
1056 007032 001373                BNE     L.LD2                ;NO. WAIT FOR THE ONE.
1057
1058 007034 004737 167114        JSR      PC, L.PTR!160000        ;YES. SKIP OVER THE NEXT CHARACTER NOW.
1059
1060 007040 004737 167126        JSR      PC, L.GWRD!160000        ;ASSEMBLE A WORD NOW.
1061 007044 010002                MOV     L.BYT, L.BC            ;MOVE OVER TO THE COUNTER.
1062 007046 162702 000004        SUB     #4, L.BC              ;REDUCE TO ACTUAL DATA COUNT.
1063 007052 022702 000002        CMP     #2, L.BC              ;ANY DATA AT ALL?
1064 007056 001433                BEQ     L.JMP                 ;NO. MUST BE END
1065 007060 004737 167126        JSR      PC, L.GWRD!160000        ;YES. ASSEMBLE A DATA WORD NOW.
1066 007064 010001                MOV     L.BYT, L.ADR          ;AND THIS MUST BE THE FIRST ADDRESS.
1067
1068
1069 007066 004737 167114        L.LD3: JSR      PC, L.PTR!160000  ;GET A BYTE OF DATA NOW.
1070 007072 002006                BGE     L.LD4                ;ALL DONE?
1071 007074 105705                TSTB   L.CKSM                ;YEP. COUNTER IS MINUS. CHECK CHECKSUM.
1072 007076 001751                BEQ     L.LD2                ;CHECKSUM GOOD. GET NEXT COMMAND.
1073
1074
1075 007100 004337 166652        L.BAD: JSR      SCAN, OUTLIT!160000 ;BAD LOAD INFORM HOST
1076 007104          175          102        .BYTE  ALTMOD, 'B            ;SEND ALTMODE B
1077 007106 000646                BR      PRESTR†              ;AND RESTART THE DISPLAY.
1078
1079
1080 007110 110021                L.LD4: MOVB   L.BYT, (L.ADR)+    ;INSERT BYTE INTO MEMORY.
1081 007112 000765                BR      L.LD3                ;AND GET THE NEXT BYTE.
1082
1083
1084
1085 007114 004737 166664        L.PTR: JSR      PC, GET8!160000    ;ASSEMBLE AN 8 BIT CHARACTER NOW.
1086 007120 060005                ADD     L.BYT, L.CKSM        ;UPDATE THE CHECKSUM NOW.
1087 007122 005302                DEC     L.BC                 ;DECREMENT THE CHARACTER COUNTER.
1088 007124 000207                RTS     PC                   ;AND RETURN TO THE CALLER NOW.
1089
1090
1091
1092 007126 004737 167114        L.GWRD: JSR     PC, L.PTR!160000  ;ASSEMBLE A WORD. FIRST GET A CHARACTER
1093 007132 010046                MOV     L.BYT, -(SP)         ;AND SAVE IT.
1094 007134 004737 167114        JSR     PC, L.PTR!160000     ;AND THEN GET ANOTHER ONE.

```



```

1095 007140 000300          SWAB  L.BYT          ;AND THEN REASSEMBLE THE MESS.
1096 007142 052600          BIS   (SP)+,L.BYT      ;WITH THE FEARSOME POWER OF THE 11.
1097 007144 000207          RTS   PC              ;AND RETURN TO THE CALLER.
1098
1099
1100
1101
1102 007146 004737 167126    L.JMP: JSR  PC,L.GWRD!160000      ;ALL DONE WITH THE LOAD. ASSEMBLE
1103 007152 010046          MOV  L.BYT, -(SP)          ;THE STARTING ADDRESS NOW.
1104 007154 004737 167114    JSR  PC,L.PTR!160000      ;AND DON'T FORGET TO CHECKSUM IT.
1105 007160 105705          TSTB L.CKSM
1106 007162 001346          BNE  L.BAD              ;A BAD CHECKSUM. ALL IS EVIL.
1107
1108 007164 004337 166652    JSR  SCAN,OUTLIT!160000    ;GOOD CHKSUM, INFORM HOST
1109 007170      175      107    .BYTE  ALTMOD,'G        ;WITH ALTMOD G
1110
1111 007172 032716 000001    BIT  #1,(SP)            ;DO WE WANT TO START EXECUTION?
1112 007176 001401          BEQ  L.JMP1            ;YES. AWAY WE GO.
1113
1114 007200 000000          L.HALT: HALT          ;IF NOT, HALT.
1115
1116 007202 000136          L.JMP1: JMP  @ (SP)+    ;IF GO, THEN GO ALREADY. WHEEEEE!
1117
1118
1119
1120          .SBTTL  THE SELF TEST
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133          .PAGE

```

```

;THIS IS GT40 QUICK TEST
;GIVES QUICK VISUAL TEST
;OF CONDITION OF MACHINE
;WITHOUT READING IN DIAG.

```

```

1134
1135
1136
1137
1138      100000      CHAR=100000
1139      104000      SHORTV=104000
1140      110000      LONGV=110000
1141      114000      POINT=114000
1142      120000      GRAPHX=120000
1143      124000      GRAPHY=124000
1144      130000      RELATV=130000
1145
1146      002000      INT0=2000
1147      002200      INT1=2200
1148      002400      INT2=2400
1149      002600      INT3=2600
1150      003000      INT4=3000
1151      003200      INT5=3200
1152      003400      INT6=3400
1153      003600      INT7=3600
1154
1155      000100      LPOFF=100
1156      000140      LPON=140
1157      000020      BLKOFF=20
1158      000030      BLKON=30
1159
1160      000004      LINE0=4
1161      000005      LINE1=5
1162      000006      LINE2=6
1163      000007      LINE3=7
1164
1165      160000      DJMP=160000
1166      164000      DNOP=164000
1167      170000      STATSA=170000
1168      173400      DSTOP=173400
1169
1170      000300      LPLITE=300
1171      000200      LPDARK=200
1172      000040      ITALO=40
1173      000060      ITAL1=60
1174      000004      SYNON=4
1175
1176
1177      174000      STATSB=174000
1178
1179      000100      INCR=100
1180      040000      INTX=40000
1181      001777      MAXX=1777
1182      001377      MAXY=1377
1183      020000      MINUSX=20000
1184      020000      MINUSY=MINUSX
1185      017600      MAXSX=17600
1186      000077      MAXSY=77
1187      000100      MINSJY=100
1188
1189
;LOAD GRAPH INCR
;INTENSIFY BIT
;BIGGEST X VECTOR
;BIGGEST Y VECTOR
;THE MINUS BIT
;BIGGEST X IN SHORTVEC
;      Y IN
;MINUS BIT FOR Y IN SHORTVEC
;BRIGHTEST
;STOP INTERRUPT
;ITALICS OFF
;ON
;SYNC ON

```

```

1190 007204 012737 167214 172000      MOV      #FILED!160000,GT40PC      ;START THE GT40
1191 007212 000001                      WAIT                               ;AND WAIT
1192                                     FILED: POINT!BLKOFF                ;POINT--INVISIBLE
1193 007214 114020                      0
1194 007216 000000                      MAXY
1195 007220 001377
1196
1197 007222 112004                      LONGV!INTO!LINE0                  ;DRAW.TOP LINE
1198 007224 041777                      INTX!MAXX
1199 007226 000000                      0
1200
1201 007230 112405                      LONGV!INT2!LINE1
1202 007232 040000                      INTX                               ;DRAW LINE TO RIGHT
1203 007234 021377                      MINUSX!MAXY
1204
1205 007236 113006                      LONGV!INT4!LINE2
1206 007240 061777                      INTX!MINUSX!MAXX                ;DRAW BOTTOM LINE
1207 007242 000000                      0
1208
1209 007244 113407                      LONGV!INT6!LINE3
1210 007246 040000                      INTX
1211 007250 001377                      MAXY                               ;DRAW LINE TO LEFT
1212
1213 007252 114000                      POINT
1214 007254 000400                      400
1215 007256 000500                      500
1216 007260 106200                      SHORTV!INT1
1217 007262 057677                      57677                            ;+X+Y
1218 007264 106600                      SHORTV!INT3
1219 007266 077677                      77677                            ;+X-Y
1220 007270 107200                      SHORTV!INT5
1221 007272 077777                      77777                            ;-X-Y
1222 007274 107600                      SHORTV!INT7
1223 007276 057777                      57777                            ;-X+Y
1224
1225 007300 114000                      POINT
1226 007302 001400                      1400
1227 007304 000500                      500
1228 007306 133030                      RELATV!INT4!BLKON
1229 007310 057677                      57677                            ;+X+Y
1230 007312 077677                      77677                            ;+X-Y
1231 007314 077777                      77777                            ;-X-Y
1232 007316 057777                      57777                            ;-X+Y
1233
1234 007320 114000                      POINT
1235 007322 000400                      400
1236 007324 000100                      100
1237 007326 174120                      STATSB!INCR+20                  ;TRY GRAPH MODES
1238 007330 114000                      POINT
1239 007332 001000                      1000
1240 007334 000200                      200
1241
1242 007336 120000                      GRAPHX
1243 007340 001010                      1010
1244 007342 001020                      1020
1245 007344 001030                      1030

```

| | | |
|------|--------|--------|
| 1246 | 007346 | 001040 |
| 1247 | 007350 | 001050 |
| 1248 | | |
| 1249 | 007352 | 114000 |
| 1250 | 007354 | 001000 |
| 1251 | 007356 | 001200 |
| 1252 | | |
| 1253 | 007360 | 124000 |
| 1254 | 007362 | 001020 |
| 1255 | 007364 | 001030 |
| 1256 | 007366 | 001040 |
| 1257 | 007370 | 001050 |
| 1258 | 007372 | 001060 |
| 1259 | | |
| 1260 | 007374 | 160000 |
| 1261 | 007376 | 167214 |

1040
1050

POINT
1000
1200

GRAPHY
1020
1030
1040
1050
1060

DJMP
FILED!160000

.SBTTL PAPER TAPE BOOT

```

1264
1265
1266
1267
1268      177550
1269      177560
1270
1271 007400 012701 160000
1272 007404 012702 000004
1273 007410 012703 167500
1274 007414 010712
1275 007416 012706 000024
1276 007422 014304
1277 007424 005714
1278 007426 100775
1279 007430 010712
1280 007432 012706 000024
1281 007436 010441
1282
1283 007440 040601
1284 007442 010111
1285 007444 011102
1286 007446 005214
1287 007450 105714
1288 007452 100376
1289 007454 116412 000002
1290 007460 005211
1291 007462 120227 000375
1292 007466 001366
1293 007470 105222
1294 007472 000142
1295
1296
1297
1298 007474 177560
1299 007476 177550
1300
1301

```

```

: PAPER TAPE BOOT
HSR=177550      :HIGH SPEED READER ADDRESS
LSR=177560      :LOW SPFED READER ADDRESS
:
:      .=ORIGIN+1400
PTBOOT: MOV      #160000,R1      ;SET MEMORY CHECK LIMITS
:      MOV      #4,R2           ;TRAP ADDRESS IS LOC. 4
:      MOV      #DEV+4!160000,R3 ; POINTER TO DEVICE ADDRESSES
:      MOV      PC,@R2         ;PRESET TRAP ADDRESS IN LOC. 4
:      MOV      #24,SP        ;STACK SET UP AT SPECIAL ADDRESS
DEV1:   MOV      -(R3),R4      ;GET DEVICE ADDRESS
:      TST      @R4           ;CHECK AVAILABILITY OF DEVICE
:      BMI      DEV1         ;CHECK DEVICE FOR ERRORS
:      MOV      PC,@R2       ;RESET TRAP ADDRESS AT LOC. 4
:      MOV      #24,SP      ;SPECIAL ADDRESS USED AS MASK LATER
:      MOV      R4,-(R1)     ;DO MEM CHK:READER STATUS ADDRESS
:      IS MOVED
:      SET R1=X7752,MASK IN SP=24
:      STORE OWN ADDRESS IN POINTER
LOOP:   MOV      @R1,R2       ;GET BYTE POINTER
:      INC      @R4          ;ENABLE READER
:      TSTB    @R4          ;TEST DONE BIT
:      BPL     -2           ;WAIT UNTIL READY
:      MOVB    2(R4),@R2    ;THEN PICK IT UP AND STORE IT
:      INC     @R1         ;BUMP POINTER
:      CMPB   R2,#375      ;STORED JUMP OFFSET?
:      BNE    LOOP        ;NOT YET
:      INCB   (R2)+       ;YES, ALL DONE
:      JMP    -(R2)       ;GO EXECUTE AS BRANCH
:
: DEVICE ADDRESSES FOLLOW - DO NOT CHANGE THE ORDER
DEV:   LSR
:      HSR
:
: .SBTTL CASSETTE BOOT

```

```

1302
1303
1304
1305      177500
1306
1307      007500  012700  177500
1308      007504  005010
1309      007506  010701
1310      007510  062701  000052
1311      007514  012702  000375
1312      007520  112103
1313
1314      007522  112110
1315      007524  100413
1316      007526  130310
1317      007530  001776
1318      007532  105202
1319      007534  100772
1320      007536  116012  000002
1321      007542  120337  000000
1322      007546  001767
1323      007550  000000
1324      007552  000755
1325
1326      007554  005710
1327      007556  100774
1328      007560  005007
1329
1330      007562  017640
1331
1332      007564  002415
1333
1334      007566  112024
1335
1336      007570  000000  000000
1337      007574  167500
1338      007576  000340
1339
1340
1341

```

```

: CASSETTE BOOT
TACS=177500 ;TA-11 CONTROL AND STATUS REGISTER
TABOOT: MOV =ORIGIN+1500
          CLR #TACS,R0
RES:     MOV (R0)
          MOV PC,R1 ;SELECT UNIT #0
          ADD #TABLE-.R1 ;USE FOR PIC
          MOV #375,R2 ;R1 HOLDS ADDR. OF COMMAND TABLE
          MOVB (R1)+,R3 ;MEMORY PTR. AND DATA FLAG
          ;TEST BITS

LOOP1:   MOVB (R1)+,(R0) ;COMMAND FROM TABLE TO TACS
          BMI DONE ;WHEN COMMAND CODE NEG., QUIT
LOOP2:   BITB R3,(R0) ;TEST READY AND T-REQ BITS IN TACS
          BEQ LOOP2 ;LOOP 'TIL SOMETHING COMES UP
          INCB R2 ;ADVANCE MEMORY POINTER
          BMI LOOP1 ;IF MINUS, TRY NEXT COMMAND
          MOVB 2(R0),(R2) ;READ DATA INTO MEMORY
          CMPB R3,#0 ;FIRST BYTE READ SHOULD BE '240'
          BEQ LOOP2 ;IF O.K., GO READ ANOTHER BYTE
STOP:   HALT ;HALT ON ERROR
        BR RES ;RESTART ON CONTINUE

DONE:   TST (R0) ;CHECK FOR ERROR
        BMI STOP ;HALT ON ERROR
        CLR PC ;= 'JMP #0'

TABLE:  .WORD 17640 ;. BYTE 240: READY+T-REQ.
        .WORD 2415 ;. BYTE 37: ILBS+READY+GO
        .WORD 112024 ;. BYTE 15: SFB+GO
        .WORD 0 ;. BYTE 5: READ+GO
        .WORD 0 ;. BYTE 24: READ+ILBS
        .WORD 0 ;. BYTE 224: READ+ILBS+E.O.TABLE
        .WORD TABOOT!160000 ;THESE ARE FILLER WORDS
        .WORD 340 ;POWER UP VECTOR AND PRIORITY

```

```

.SBTTL MR11-DB 900T

```

:MR11-DB BULK STORAGE PROGRAM LOADER LISTING

```

1342
1343
1344
1345
1346 007600 010702
1347 007602 000451
1348 007604 177462
1349 007606 000005
1350
1351 007610 010702
1352 007612 000445
1353 007614 177406
1354 007616 000005
1355
1356
1357 007620 010702
1358 007622 000417
1359 007624 177344
1360 007626 000005
1361 007630 004003
1362 007632 100000
1363 007634 024000
1364
1365
1366 007636 010702
1367 007640 000410
1368 007642 172524
1369 007644 060003
1370 007646 060011
1371 007650 000200
1372 007652 100000
1373
1374
1375 007654 010702
1376 007656 000423
1377 007660 176716
1378
1379
1380 007662 000005
1381 007664 010200
1382 007666 005720
1383 007670 012001
1384 007672 005311
1385 007674 005720
1386 007676 012041
1387 007700 031011
1388 007702 001776
1389 007704 005720
1390 007706 031041
1391 007710 001406
1392 007712 000112
1393
1394
1395 007714 167600
1396 007716 000340
1397

```

```

:      . =ORIGIN+1600                ;KEEP TRACK OF ORIGIN
RF11:  MOV PC,R2                    ;FIXED HEAD DISK (256 KW)
      BR OTHER
      177462
      5
RK11:  MOV PC,R2                    ;MOVING HEAD DISK (CARTRIDGE)
      BR OTHER
      177406
      5
TC11:  MOV PC,R2
      BR TAPES
      177344
      5
      4003
      100000
      24000
      ;ADDRESS OF WORD COUNT
      ;LAST COMMAND
      ;FIRST COMMAND
      ;DONE MASK
      ;ERROR MASK
TM11:  MOV PC,R2
      BR TAPES
      172524
      60003
      60011
      200
      100000
      ;ADDRESS OF BYTE COUNT
      ;LAST COMMAND
      ;FIRST COMMAND
      ;DONE MASK
      ;ERROR MASK
RF11:  MOV PC,R2                    ;MOVING HEAD DISK (PACK)
      BR OTHER
      176716
TAPES:  RESET
      MOV R2,R0
      TST (0)+
      MOV (0)+,R1
      DEC (1)
      TST (0)+
      MOV (0)+,-(1)
      BIT (0),(1)
      BEQ ,-2
      TST (0)+
      BIT (0),-(1)
      BEQ OTHER
AGAIN:  JMP (2)
      ;GET THE ADDRESS OF THE BRANCH
      ;R0 TO POINT AT LAST COMMAND
      ;GET THE WORD COUNT ADDRESS
      ;SET UP FOR ADVANCE 1 RECORD
      ;MOVE R0 TO FIRST COMMAND
      ;COMMAND WORD TO COMMAND REG.
      ;LOOK FOR DONE INDICATORS
      ;NONE SET, TRY AGAIN
      ;DONE FIRST COMMAND, CHECK FOR ERROR
      ;LOOK FOR SET ERROR BITS
      ;NO ERRORS - TRY THE READ
      ;RERUN FOR ERRORS
RFVEC: RF11!160000
      340
      ;RF11 POWER UP VECTOR

```


1398 007720 010702
 1399 007722 000401
 1400 007724 177450
 1401
 1402
 1403 007726 000005
 1404 007730 010200
 1405 007732 005720
 1406 007734 012001
 1407 007736 012711 177000
 1408 007742 011041
 1409 007744 032711 100200
 1410 007750 001775
 1411 007752 100757
 1412 007754 005007
 1413
 1414 007756 000000
 1415 007760 167610
 1416 007762 000340
 1417 007764 167720
 1418 007766 000340
 1419 007770 167654
 1420 007772 000340
 1421 007774 167620
 1422 007776 000340
 1423
 1424
 1425

RC11: MOV PC,R2 ;FIXED HEAD DISK (64KW)
 BR OTHER
 177450 ;ADRS OF WORD COUNT (COMMAND+2)
 ;COMMAND WORD (5) IS THE RESET

OTHER: RESET
 MOV R2,R0 ;R0 TO POINT AT WORD COUNT ADRS
 TST (0)+ ;POINT TO ADDRESS
 MOV (0)+,R1 ;WORD COUNT ADDRESS TO R1
 MOV #-1000,(1) ;LOAD WORD COUNT
 MOV (0)-,(1) ;COMMAND TO COMMAND REGISTER
 BIT #100200,(1) ;CHECK FOR ERROR OR DONE
 BEQ #-4 ;IF NEITHER, KEEP LOOKING
 BMI AGAIN ;ERROR, TRY AGAIN
 CLR PC

0 ;FILLER

RKVEC: RK11!160000 ;RK POWER UP VECTOR
 340

RCVEC: RC11!160000 ;RC POWER UP VECTOR
 340

RPVEC: RP11!160000 ;RP POWER UP VECTOR
 340

TCVEC: TC11!160000 ;TC11 POWER UP VECTOR
 340

.SBTTL ROM VERSION 1 VALUES
 .PAGE

1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479
1480
1481

```

.USABL AMA
:DATA PATTERN STORED IN THE GT40 BOOTSTRAP VERSION 1
:
:***** THIS IS A IMAGE LISTING OF THE GT40 <VT40> BOOTSTRAP *****
:
:THE DATA IS A MIRROR IMAGE OF THAT IN THE BOOTSTRAP ROMS
:ONLY THE ADDRESS FIELD IS CHANGED
:BOOTVT.S09 5/2/72 <SPECIAL>
:
:VT-40 BOOTSTRAP LOADER, VERSION S09, RELEASE R01; 5/2/72
:
:COPYRIGHT 1972, DIGITAL EQUIPMENT CORPORATION.
:146 MAIN STREET
:
:MAYNARD, MASSACHUSETTS 01754
:
:WRITTEN BY JACK BURNES, SENIOR SYSTEMS ARCHITECT!
:
:THIS ROUTINE IS INTENDED TO BE LOADED IN THE ROM PORTION OF THE VT-40.
:
:REGISTER DEFINITIONS:
:
:R0=%0
:R1=%1
:R2=%2
:R3=%3
:R4=%4
:R5=%5
:R6=%6
:R7=%7
:
:SP=R6
:PC=R7
:
:RET1=R0 ;RETURN OF VALUE REGISTER.
:INP1=R1 ;ARGUMENT FOR CALLED FUNCTION
:INP2=R2 ;SECOND ARGUMENT.
:WORK1=R3 ;FIRST WORK REGISTER.
:WORK2=R4 ;SECOND WORKING REGISTER.
:SCR1=R5 ;SCRATCH REGISTER.
:
:LCKSM=WORK1 ;OVERLAPPING DEFINITIONS FOR LOADER PORTION.
:LBYT=RET1
:LBC=SCR1
:LADR=INP1

```

```

000000
000001
000002
000003
000004
000005
000006
000007
000006
000007
000000
000001
000002
000003
000004
000005
000003
000000
000005
000001

```

| | | | | | | |
|------|--------|--------|--------|----------------------------|--|--|
| 1482 | | | | | | |
| 1483 | 036000 | | | COREND=36000 | | ;FIRST LOCATION OF NON-CORE. |
| 1484 | 166000 | | | ROMORG=166000 | | ;WHERE THE ROM PROGRAM SHOULD GO. |
| 1485 | | | | | | |
| 1486 | 000000 | | | STARTX=0 | | ;WHERE TO START DISPLAYING THE X POSITIONS. |
| 1487 | 001360 | | | STARTY=1360 | | ;WHERE TO START DISPLAYING THE Y. |
| 1488 | | | | | | |
| 1489 | | | | | | |
| 1490 | 022000 | | | VT40PC=172000-150000 | | ;VT40 PROGRAM COUNTER. |
| 1491 | 027560 | | | KBDIS=27560 | | ;TTY INPUT STATUS. |
| 1492 | 025614 | | | P100S=25614 | | ;PDP-10 OUTPUT STATUS. |
| 1493 | 025610 | | | P10IS=25610 | | ;PDP-10 INPUT STATUS. |
| 1494 | | | | | | |
| 1495 | 027562 | | | KBDIB=KBDIS+2 | | ;TTY INPUT BUFFER. |
| 1496 | 025612 | | | P10IB=P10IS+2 | | ;PDP-10 INPUT CHARACTER. |
| 1497 | 025616 | | | P10OB=P10OS+2 | | ;PDP-10 OUTPUT BUFFER. |
| 1498 | | | | | | |
| 1499 | | | | | | |
| 1500 | 045776 | | | P100C=COREND-2+10000 | | ;CHARACTER TO BE SENT TO THE PDP-10 |
| 1501 | 045772 | | | P10IC=P100C-4 | | ;INPUT CHARACTER FROM IO PLUS ONE SAVE CHARACTER |
| 1502 | 015770 | | | STKSRT=P10IC-2-30000 | | ;FIRST LOCATION OF STACK. |
| 1503 | | | | | | |
| 1504 | | | | | | |
| 1505 | 160000 | | | JMPCIS=160000 | | ;THE VT-40 DISPLAY JUMP INSTRUCTION. |
| 1506 | | | | | | |
| 1507 | | | | | | |
| 1508 | 000024 | | | PWRFAL=24 | | ;POWER FAIL RESTART LOCATION. |
| 1509 | | | | | | |
| 1510 | | | | | | |
| 1511 | | | | | | |
| 1512 | | | | | | |
| 1513 | | | | | | |
| 1514 | | | | | | |
| 1515 | | | | | | |
| 1516 | | | | | | |
| 1517 | | | | | | |
| 1518 | | | | | | |
| 1519 | | | | | | |
| 1520 | 016000 | | | .=16000 | | |
| 1521 | | | | .=ROMORG | | ;SET THE ORIGIN NOW!!!! |
| 1522 | | | | | | |
| 1523 | | | | | | |
| 1524 | | | | | | |
| 1525 | | | | | | |
| 1526 | | | | | | |
| 1527 | | | | | | |
| 1528 | | | | | | |
| 1529 | 016000 | 012705 | 000026 | STARTA: MOV #PWRFAL+2,SCR1 | | ;PICK UP POINTER TO P.F. STATUS. |
| 1530 | 016004 | 005015 | | CLR 2SCR1 | | ;CLEAR IT OUT TO BE SURE. |
| 1531 | 016006 | 010745 | | MOV PC,-(SCR1) | | ;SET UP THE RESTART LOCATION. |
| 1532 | | | | | | |
| 1533 | 016010 | 000005 | | RESET | | ;RESET THE BUS. |
| 1534 | | | | | | |
| 1535 | 016012 | 012767 | 000007 | MOV #7,P10IS | | ;INITIALIZE PDP-10 INPUT |
| 1536 | 016020 | 012767 | 000001 | MOV #1,KBDIS | | ;INITIALIZE TTY INPUT. |
| 1537 | 016026 | 012767 | 000201 | MOV #201,P10OS | | ;INITIALIZE PDP-10 OUTPUT. |

```

1538
1539
1540
1541 016034 012706 015770 RSTRT: MOV #STKSRT,SP ;SET UP THE STACK NOW!
1542 016040 005001 CLR LADR ;CLEAR ADDRESS POINTER.
1543 016042 012702 160000 MOV #JMPDIS,INP2 ;PLACE A DISPLAY JUMP INSTRUCTION IN A REGISTER.
1544 016046 010221 MOV INP2,(LADR)+ ;MOVE IT TO LOCATION 0.
1545 016050 012711 166756 MOV #DISPRG+15000,(LADR) ;MOVE ADDRESS POINTER INTO 2.
1546 016054 012701 000030 MOV #PWRFAL+4,LADR ;SET UP WHERE WE WILL STORE CHARACTERS.
1547 016060 005000 CLR RET1 ;PREPARE TO INSERT A ZERO CHARACTER.
1548 016062 004767 000022 JSR PC,DOCHAR ;INSERT IT NOW.
1549 016066 005067 003706 CLR VT40PC ;CLEAR THE DISPLAY PROGRAM COUNTER AND START.
1550
1551 016072 004767 000210 MAJOR: JSR PC,GTCHR ;GT A CHARACTER NOW.
1552 016076 000240 NOP
1553 016100 000240 NOP
1554 016102 000240 NOP
1555 016104 012746 166072 MOV #MAJOR+15000,-(SP) ;INSERT IN DISPLAY BUFFER NOW.
1556
1557 016110 010105 DOCHAR: MOV LADR,SCR1 ;GT CURRENT BUUFER POSITION NOW.
1558 016112 022525 CMP (SCR1)+,(SCR1)+ ;BYPASS CURRENT DISPLAY JUMP.
1559 016114 005025 CLR (SCR1)+ ;CLEAR FUTURE ADDRESS FOR JUMP.
1560 016116 010225 MOV INP2,(SCR1)+ ;STICK IN TEMPORARY JUMP WHILE WE REPLACE CURREN
1561 016120 005015 CLR (SCR1) ;A DISPLAY JUMP TO ZERO.
1562 016122 005011 CLR (LADR) ;NOW REPLACE CURRENT DISPLAY JUMP BY THE CHARACT
1563 016124 050021 BIS RET1,(LADR)+ ;IT'S DONE THIS WAY TO WASTE 2 CYCLES.
1564 016126 010211 MOV INP2,(LADR) ;TO AVOID TIMING PROBLEMS WITH THE VT40.
1565 016130 000207 RTS PC ;AND FINALLY RETURN.
1566
1567
1568
1569
1570
1571
1572
1573
1574
1575
1576
1577
1578
1579
1580
1581
1582 016132 004767 000124 GT8: JSR PC,GTSIX ;GT SIX BITS NOW.
1583 016136 010046 MOV RET1,-(SP) ;SAVE THE CHARACTER NOW.
1584 016140 000401 BR GTP84 ;BYPASS THE 8'ER
1585 016142 005002 GT84: CLR INP2 ;RESET THE MAGIC REGISTER NOW.
1586 016144 005722 GTP84: TST (INP2)+ ;INCREMENT WHERE TO GO.
1587 016146 066207 166250 ADD GT8TB+15000(INP2),PC ;UPDATE PC NOW.
1588
1589 016152 GT8P=.
1590
1591 016152 004767 000104 GT81: JSR PC,GTSIX ;GT A CHARACTER NOW.
1592 016156 010004 MOV RET1,WORK2 ;SAVE FOR A SECOND.
1593 016160 006300 ASL RET1

```

| | | | | | | | | | |
|------|--------|--------|---------|--------|-------------|-------------------|--|--|---|
| 1594 | 016162 | 006300 | | ASL | RET1 | | | | |
| 1595 | 016164 | 106300 | | ASLB | RET1 | | | | ;SHIFT TO LEFT OF BYTE |
| 1596 | 016166 | 106116 | | ROLB | QSP | | | | |
| 1597 | 016170 | 106300 | | ASLB | RET1 | | | | ;PACK THEM IN. |
| 1598 | 016172 | 106116 | | ROLB | QSP | | | | |
| 1599 | 016174 | 012600 | | MOV | (SP)+,RET1 | | | | ;A GOOD 8 BIT THING. |
| 1600 | 016176 | 000207 | | RTS | PC | | | | ;POP AND RETURN NOW. |
| 1601 | | | | | | | | | |
| 1602 | 016200 | 006300 | GT82: | ASL | RET1 | | | | |
| 1603 | 016202 | 006300 | | ASL | RET1 | | | | |
| 1604 | 016204 | 106300 | | ASLB | RET1 | | | | |
| 1605 | 016206 | 106104 | | ROLB | WORK2 | | | | |
| 1606 | 016210 | 106300 | | ASLB | RET1 | | | | |
| 1607 | 016212 | 106104 | | ROLB | WORK2 | | | | |
| 1608 | 016214 | 106300 | | ASLB | RET1 | | | | |
| 1609 | 016216 | 106104 | | ROLB | WORK2 | | | | |
| 1610 | 016220 | 106300 | | ASLB | RET1 | | | | |
| 1611 | 016222 | 106104 | | ROLB | WORK2 | | | | |
| 1612 | 016224 | 010400 | | MOV | WORK2,RET1 | | | | |
| 1613 | 016226 | 012604 | | MOV | (SP)+,WORK2 | | | | |
| 1614 | 016230 | 000207 | | RTS | PC | | | | |
| 1615 | | | | | | | | | |
| 1616 | 016232 | 006100 | GT83: | ROL | RET1 | | | | |
| 1617 | 016234 | 006100 | | ROL | RET1 | | | | |
| 1618 | 016236 | 006004 | | ROR | WORK2 | | | | |
| 1619 | 016240 | 106000 | | RORB | RET1 | | | | |
| 1620 | 016242 | 006004 | | ROR | WORK2 | | | | |
| 1621 | 016244 | 106000 | | RORB | RET1 | | | | |
| 1622 | 016246 | 005726 | | TST | (SP)+ | | | | ;FINAL CHARACTER ASSEMBLED. |
| 1623 | 016250 | 000207 | | RTS | PC | | | | ;FUDGE STACK. |
| 1624 | | | | | | | | | ;AND RETURN NOW. |
| 1625 | | 016250 | GT8TB = | | .-2 | | | | |
| 1626 | | | | | | | | | ;FUSH ZERO CONDITION BACK INTO NEVER-NEVER LAND. |
| 1627 | 016252 | 000000 | | .WORD | GT81-GT8P | | | | |
| 1628 | 016254 | 000026 | | .WORD | GT82-GT8P | | | | |
| 1629 | 016256 | 000060 | | .WORD | GT83-GT8P | | | | |
| 1630 | 016260 | 177770 | | .WORD | GT84-GT8P | | | | |
| 1631 | | | | | | | | | |
| 1632 | | | | | | | | | |
| 1633 | 016262 | 004767 | 000020 | GTSIX: | JSR | PC,GTCHR | | | |
| 1634 | 016266 | 020027 | 000040 | | CMP | RET1,#40 | | | |
| 1635 | 016272 | 002546 | | | BLT | L9AD | | | |
| 1636 | 016274 | 020027 | 000137 | | CMP | RET1,#137 | | | |
| 1637 | 016300 | 003143 | | | BGT | L9AD | | | |
| 1638 | 016302 | 000207 | | | RTS | PC | | | |
| 1639 | | | | | | | | | |
| 1640 | | | | | | | | | |
| 1641 | | | | | | | | | |
| 1642 | 016304 | 005726 | GTCHP: | TST | (SP)+ | | | | ;UPDATE THE STACK. |
| 1643 | | | | | | | | | |
| 1644 | 016306 | 012700 | 015772 | GTCHR: | MOV | #P10IC-30000,RET1 | | | |
| 1645 | 016312 | 004767 | 000064 | GTCHL: | JSR | PC,CHECK | | | ;SET UP POINTER TO THE INPUT CHARACTER. |
| 1646 | 016316 | 005710 | | | TST | QRET1 | | | |
| 1647 | 016320 | 001774 | | | BEQ | GTCHL | | | ;ANY CHARACTER'S THERE? |
| 1648 | 016322 | 011046 | | | MOV | QRET1,-(SP) | | | |
| 1649 | 016324 | 005020 | | | CLR | (RET1)+ | | | ;PUSH THE CHAR ON THE STACK. ;CLEAR THE CHAR GOT FLAG NOW. |

| | | | | | | | | |
|------|--------|--------|--------|---------|------|-------------|--|--|
| 1650 | 016326 | 042716 | 177600 | | BIC | #-200,(SP) | | ;CLEAR AWAY PARITY NOW. |
| 1651 | 016332 | 001764 | | | BEQ | GTCHP | | ;IF ZERO, GT ANOTHER |
| 1652 | 016334 | 022716 | 000177 | | CMP | #177,(SP) | | |
| 1653 | 016340 | 001761 | | | BEQ | GTCHP | | ;ALSO IGNORE RUBOUTS. |
| 1654 | 016342 | 022710 | 000175 | | CMP | #175,@RET1 | | ;WAS IT A "175" |
| 1655 | 016346 | 001007 | | | BNE | GTNP | | ;NOPE. |
| 1656 | 016350 | 011610 | | | MOV | (SP),@RET1 | | ;YEP. RESET IN CASE OF ABORT. |
| 1657 | 016352 | 021027 | 000122 | | CMP | @RET1,#122 | | ;IS IT AN R |
| 1658 | 016356 | 001626 | | | BEQ | RSTR | | ;YEP. RESTART |
| 1659 | 016360 | 021027 | 000114 | | CMP | @RET1,#114 | | ;IS IT AN L |
| 1660 | 016364 | 001455 | | | BEQ | LOAD | | ;YEP. LOAD. |
| 1661 | | | | | | | | |
| 1662 | 016366 | 011610 | | GTNP: | MOV | (SP),@RET1 | | ;NOW DO THE FDUGING. |
| 1663 | 016370 | 012600 | | | MOV | (SP),RET1 | | |
| 1664 | 016372 | 020027 | 000175 | | CMP | RET1,#175 | | |
| 1665 | 016376 | 001743 | | | BEQ | GTCHR | | ;IF ALTMODE, LOOP |
| 1666 | 016400 | 000207 | | | RTS | PC | | |
| 1667 | | | | | | | | |
| 1668 | | | | | | | | |
| 1669 | | | | | | | | |
| 1670 | | | | | | | | |
| 1671 | | | | | | | | |
| 1672 | | | | | | | | |
| 1673 | | | | | | | | |
| 1674 | | | | | | | | |
| 1675 | 016402 | 005767 | 027370 | CHECK: | TST | P100C | | ;DO WE WANT TO OUTPUT? |
| 1676 | 016406 | 071410 | | | BEQ | CHECK1 | | ;NO. |
| 1677 | 016410 | 105767 | 007200 | | TSTB | P100S | | ;WE DO. IS THE IO READY? |
| 1678 | 016414 | 100005 | | | BPL | CHECK1 | | ;NOT QUITE. |
| 1679 | 016416 | 016767 | 027354 | 007172 | MOV | P100C,P100B | | ;IT'S READY. SEND THE CHARACTER. |
| 1680 | 016424 | 005067 | 027346 | | CLR | P100C | | ;AND THE SAVED CHARACTER. |
| 1681 | | | | | | | | |
| 1682 | 016430 | 105767 | 011124 | CHECK1: | TSTB | KBDIS | | ;HEY. IS THE KEYBOARD READY? |
| 1683 | 016434 | 100014 | | | BPL | CHECK3 | | ;NOPE. NO LUCK. |
| 1684 | 016436 | 116746 | 011120 | | MOV | KBDIB,-(SP) | | ;YEP. SAVE THE CHARACTER NOW. |
| 1685 | 016442 | 012767 | 000001 | 011110 | MOV | #1,KBDIS | | ;AND REENABLE THE COMMUNICATIONS DEVICE. |
| 1686 | | | | | | | | |
| 1687 | 016450 | 004767 | 177726 | CHECK2: | JSR | PC,CHECK | | ;IS THE OUTPUT READY? |
| 1688 | 016454 | 005767 | 027316 | | TST | P100C | | |
| 1689 | 016460 | 001373 | | | BNE | CHECK2 | | ;IF NOT, WAIT TILL DONE. |
| 1690 | 016462 | 012667 | 007130 | | MOV | (SP)+,P100B | | ;AND THEN SEND OUT THE CHARACTER. |
| 1691 | | | | | | | | |
| 1692 | | | | | | | | |
| 1693 | 016466 | 105767 | 007116 | CHECK3: | TSTB | P10IS | | ;IS THE IO TALKING TO ME. |
| 1694 | 016472 | 100011 | | | BPL | CHECK4 | | ;NOPE. EXIT. |
| 1695 | 016474 | 116767 | 007112 | 027270 | MOV | P10IB,P10IC | | ;GT THE CHARACTER NOW. |
| 1696 | 016502 | 052767 | 177400 | 027262 | BIS | #-400,P10IC | | ;MAKE SURE IT'S NONE ZERO. |
| 1697 | 016510 | 012767 | 000007 | 007072 | MOV | #7,P10IS | | ;REINITIALIZE COMMUNICATION LINE. |
| 1698 | | | | | | | | |
| 1699 | 016516 | 000207 | | CHECK4: | RTS | PC | | ;AND RETURN. |
| 1700 | | | | | | | | |
| 1701 | | | | | | | | |
| 1702 | | | | | | | | |
| 1703 | | | | | | | | |
| 1704 | | | | | | | | |
| 1705 | | | | | | | | |

```

1706
1707
1708
1709
1710
1711
1712
1713
1714
1715
1716
1717
1718 016520 005002
1719 016522 012712 172000
1720 016526 012706 015770
1721
1722 016532 005003
1723 016534 004767 000070
1724 016540 105300
1725 016542 001373
1726 016544 004767 000060
1727
1728 016550 004767 000072
1729 016554 010005
1730 016556 162705 000004
1731 016562 022705 000002
1732 016566 001437
1733 016570 004767 000052
1734 016574 010001
1735
1736 016576 004767 000026
1737 016602 002010
1738 016604 105703
1739 016606 001751
1740
1741 016610 012700
1742
1743 016612 102 175
1744 016614 004767 000110
1745 016620 000167 177210
1746
1747 016624 110021
1748 016626 000763
1749
1750 016630 004767 177276
1751 016634 060003
1752 016636 042700 177400
1753 016642 005305
1754 016644 000207
1755
1756 016646 004767 177756
1757 016652 010046
1758 016654 004767 177750
1759 016660 000300
1760 016662 052600
1761 016664 000207

```

; THE L O A D E R

```

LOAD: CLR INP2 ;RESET TO FIRST 8 BIT CHARACTER.
MOV #172000,(INP2) ;AND ALSO CLEVERLY STOP THE VT40.
MOV #STKSRT,SP ;RESET STACK POINTER NOW.

LLD2: CLR LCKSM ;CLEAR THE CHECKSUM
JSR PC,LPTR ;GT A BYTE NOW.
DECB LBYT ;IS IT ONE?
BNE LLD2 ;NOPE. WAIT AWHILE
JSR PC,LPTR ;YEP. GT NEXT CHARACTER.

JSR PC,LGWRD ;GT A WORD.
MOV LBYT,LBC ;GT THE COUNTER NOW.
SUB #4,LBC ;CHOP OFF EXTRA STUFF.
CMP #2,LBC ;NULL?
BEQ L JMP ;YEP. MUST BE END.
JSR PC,LGWRD ;NOPE. GT THE ADDRESS.
MOV LBYT,LADR ;AND REMEMBER FOR OLD TIMES SAKE.

LLD3: JSR PC,LPTR ;GT A BYTE (DATA)
BGE LLD4 ;ALL DONE WITH THE COUNTER?
TSTB LCKSM ;YEP. GOOD CHECK SUM?
BEQ LLD2 ;NOPE. LOAD ERROR.

LBAD: MOV (PC)+,RET1 ;SEND OUT SOME CHARACTERS NOW.
; .BYTE 175,102 ;"CTRL BAD"
; .BYTE 102,175 ;"BAD CTRL"
JSR PC,SENDIT
JMP RSTRT

LLD4: MOVB LBYT,(LADR)+ ;PLACE THE BYTE IN CORE.
BR LLD3 ;GT ANOTHER ONE.

LPTR: JSR PC,GT8 ;GT 8 BITS NOW.
ADD LBYT,LCKSM ;UPDATE CHECKSUM
BIC #177400,LBYT ;CLEAN UP THE BYTE NOW.
DEC LBC ;UPDATE THE COUNTER.
RTS PC ;RETURN NOW.

LGWRD: JSR PC,LPTR ;GT A CHARACTER.
MOV LBYT,-(SP) ;SAVE FOR A SECOND.
JSR PC,LPTR ;GT ANOTHER CHARACTER.
SWAB LBYT ;NOW ASSEMBLE THE WORD.
BIS (SP)+,LBYT ;AND RETURN WITH A 16 BITER.
RTS PC

```


| | | | | | | |
|------|--------|--------|--------|---------|-------|---|
| 1762 | | | | | | |
| 1763 | 016666 | 004767 | 177754 | LJMP: | JSR | PC, LGWRD ;GT A WORD |
| 1764 | 016672 | 010046 | | | MOV | LBYT, -(SP) ;SAVE ON THE STACK. |
| 1765 | 016674 | 004767 | 177730 | | JSR | PC, LPTR ;GT A CHARACTER. |
| 1766 | 016700 | 105703 | | | TSTB | LCKSM ;IS IT ZERO? |
| 1767 | 016702 | 001342 | | | BNE | LBAD ;YEP. WHAT CRAP. |
| 1768 | 016704 | 032716 | 000001 | | BIT | #1, (SP) ;IS IT ODD? |
| 1769 | 016710 | 001406 | | | BEQ | LJMP1 ;YEP. START PROGRAM GOING NOW. |
| 1770 | 016712 | 012700 | | | MOV | (PC)+, RET1 ;TELL PDP-10 WE'VE LOADED OK. |
| 1771 | | | | | | ;"CTRL GOOD" |
| 1772 | 016714 | 107 | 175 | | .BYTE | 175, 107 ;"GOOD CTRL" |
| 1773 | 016716 | 004767 | 000006 | | .BYTE | 107, 175 |
| 1774 | 016722 | 000000 | | | JSR | PC, SENDIT |
| 1775 | 016724 | 000776 | | | HALT | |
| 1776 | | | | | BR | .-2 |
| 1777 | 016726 | 000136 | | LJMP1: | JMP | 2(SP)+ ;AND AWAY WE GO. |
| 1778 | | | | | | |
| 1779 | | | | | | |
| 1780 | | | | | | |
| 1781 | | | | | | |
| 1782 | | | | | | |
| 1783 | | | | | | |
| 1784 | | | | | | |
| 1785 | | | | | | |
| 1786 | | | | | | |
| 1787 | | | | | | |
| 1788 | | | | | | |
| 1789 | | | | | | |
| 1790 | | | | | | |
| 1791 | | | | | | |
| 1792 | | | | | | |
| 1793 | | | | | | |
| 1794 | | | | | | |
| 1795 | 016730 | 004767 | 177446 | SENDIT: | JSR | PC, CHECK ;POLL THE OUTPUT DEVICE NOW. |
| 1796 | 016734 | 005767 | 027036 | | TST | P100C ;OUTPUT CLEAR? |
| 1797 | 016740 | 001373 | | | BNE | SENDIT ;NOPE. LOOP AWHILE LONGER. |
| 1798 | 016742 | 010067 | 006650 | | MOV | RET1, P100B ;SEND OUT THE CHARACTER. |
| 1799 | 016746 | 105000 | | | CLRB | RET1 ;CLEAR THE BYTE. |
| 1800 | 016750 | 000300 | | | SWAB | RET1 ;AND SWAP THEM NOW. |
| 1801 | 016752 | 001366 | | | BNE | SENDIT ;IF NOT EQUAL, REPEAT. |
| 1802 | 016754 | 000207 | | | RTS | PC |
| 1803 | | | | | | |
| 1804 | | | | | | |
| 1805 | | | | | | |
| 1806 | | | | | | |
| 1807 | | | | | | |
| 1808 | | | | | | |
| 1809 | | | | | | |
| 1810 | | | | | | |
| 1811 | | | | | | |
| 1812 | | | | | | |
| 1813 | | | | | | |
| 1814 | | | | | | |
| 1815 | | | | | | |
| 1816 | | | | | | |
| 1817 | | | | | | |

1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837
1838
1839
1840
1841
1842
1843
1844
1845
1846
1847
1848
1849
1850

016756 170256
016760 115124
016752 000000
016764 001360
016766 100000
016770 160000
016772 000030
016774 000000
016776 000000

000001

;
:
:
;

THIS IS THE INITIALIZING VT40 PROGRAM WHICH WILL
JUMP TO THE PROGRAM AFTER THE POWER FAIL LOCATIONS
WHICH WILL JUMP TO ZERO WHICH WILL JUMP BACK TO HERE.

DISPRG: .WORD 170256
.WORD 115124
.WORD STARTX
.WORD STARTY
.WORD 100000
.WORD JMPDIS
.WORD PWRFAL+4
.WORD 0
.WORD 0

;LOAD STATUS REGISTER FOR NORMAL OPERATION.
;SET POINT MODE, "NORMAL".
;X COORDINATE
;Y COORDINATE
;SET CHARACTER MODE.
;THEN JUMP TO THE POWERFAIL LOCATION.
;TO DISPLAY USERS CHARACTERS.

.END

| | | | | | | | | | | | | | | | | | | | | |
|----------------|----------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|--|--|--|--|--|--|
| OTHER | 007726 | 1347 | 1352 | 1376 | 1391 | 1399 | 1403* | | | | | | | | | | | | | |
| OUTLIT | 006652 | 647 | 950* | 1075 | 1108 | | | | | | | | | | | | | | | |
| O2A | 002146 | 325 | 329 | 411* | | | | | | | | | | | | | | | | |
| O2AA | 002176 | 420* | 432 | | | | | | | | | | | | | | | | | |
| PC | =%000C07 | 181* | 314* | 455* | 530* | 724* | 734* | 749* | 757* | 764 | 774* | 795* | 810* | 813* | | | | | | |
| | | 820* | 899* | 915* | 919* | 937* | 943* | 975* | 990* | 999* | 1014* | 1024* | 1054* | 1058* | | | | | | |
| | | 1060* | 1065* | 1069* | 1085* | 1088* | 1092* | 1094* | 1097* | 1102* | 1104* | 1274 | 1279 | 1309 | | | | | | |
| | | 1328* | 1346 | 1351 | 1357 | 1365 | 1375 | 1398 | 1412* | 1466* | 1531 | 1548* | 1551* | 1565* | | | | | | |
| | | 1582* | 1587* | 1591* | 1600* | 1614* | 1623* | 1633* | 1638* | 1645* | 1666* | 1687* | 1699* | 1723* | | | | | | |
| | | 1726* | 1728* | 1733* | 1736* | 1741 | 1744* | 1750* | 1754* | 1756* | 1758* | 1761* | 1763* | 1765* | | | | | | |
| | | 1770 | 1773* | 1795* | 1802* | | | | | | | | | | | | | | | |
| POINT = 114000 | | 1141* | 1193 | 1213 | 1225 | 1234 | 1239 | 1249 | | | | | | | | | | | | |
| POINTR=%000031 | | 539* | 559 | 694* | 696* | 702* | 803* | 804 | 806 | 808 | 809* | 815 | 816* | 817* | | | | | | |
| | | 818* | | | | | | | | | | | | | | | | | | |
| PRESTR | 006624 | 926* | 1077 | | | | | | | | | | | | | | | | | |
| PRGO | 001034 | 185* | 307 | | | | | | | | | | | | | | | | | |
| PRGOR | 001040 | 186* | 187 | | | | | | | | | | | | | | | | | |
| PRG1 | 001506 | 166 | 312* | 346 | | | | | | | | | | | | | | | | |
| PRG1A | 001544 | 320* | | | | | | | | | | | | | | | | | | |
| PRG1B | 001562 | 324* | 340 | | | | | | | | | | | | | | | | | |
| PRG1C | 001600 | 329* | 336 | | | | | | | | | | | | | | | | | |
| PRG1D | 001650 | 334 | 341* | | | | | | | | | | | | | | | | | |
| PRG2 | 001670 | 167 | 350* | 354 | | | | | | | | | | | | | | | | |
| PRMTRS | 001024 | 165 | 180* | | | | | | | | | | | | | | | | | |
| PSW = 177776 | | 160* | 403* | | | | | | | | | | | | | | | | | |
| PTBOOT | 007400 | 1271* | | | | | | | | | | | | | | | | | | |
| PWRFAL= 000024 | | 1508* | 1529 | 1546 | 1834 | | | | | | | | | | | | | | | |
| P101B = 025612 | | 1496* | 1695 | | | | | | | | | | | | | | | | | |
| P101C = 045772 | | 1501* | 1502 | 1644 | 1695* | 1696* | | | | | | | | | | | | | | |
| P101S = 025610 | | 1493* | 1496 | 1535* | 1693 | 1697* | | | | | | | | | | | | | | |
| P100B = 025616 | | 1497* | 1679* | 1690* | 1798* | | | | | | | | | | | | | | | |
| P100C = 045776 | | 1500* | 1501 | 1675 | 1679 | 1680* | 1688 | 1796 | | | | | | | | | | | | |
| P100S = 025614 | | 1492* | 1497 | 1537* | 1677 | | | | | | | | | | | | | | | |
| RCVEC | 007764 | 1417* | | | | | | | | | | | | | | | | | | |
| RC11 | 007720 | 1398* | 1417 | | | | | | | | | | | | | | | | | |
| RELATV= 130000 | | 1144* | 1228 | | | | | | | | | | | | | | | | | |
| RES | 007506 | 1309* | 1324 | | | | | | | | | | | | | | | | | |
| RESTR | 006060 | 678* | 926 | | | | | | | | | | | | | | | | | |
| RETURN | 002130 | 187* | 397* | 401* | 404 | | | | | | | | | | | | | | | |
| RET1 =%000C00 | | 1468* | 1476 | 1547* | 1563 | 1583 | 1592 | 1593* | 1594* | 1595* | 1597* | 1599* | 1602* | 1603* | | | | | | |
| | | 1604* | 1606* | 1608* | 1610* | 1612* | 1616* | 1617* | 1619* | 1621* | 1634 | 1636 | 1644* | 1646 | | | | | | |
| | | 1649 | 1649* | 1654 | 1656* | 1657 | 1659 | 1662* | 1663* | 1664 | 1741* | 1770* | 1798 | 1799* | | | | | | |
| | | 1800* | | | | | | | | | | | | | | | | | | |
| RFVEC | 007714 | 1395* | | | | | | | | | | | | | | | | | | |
| RF11 | 007600 | 1346* | 1395 | | | | | | | | | | | | | | | | | |
| RKVEC | 007760 | 1415* | | | | | | | | | | | | | | | | | | |
| RK11 | 007610 | 1351* | 1415 | | | | | | | | | | | | | | | | | |
| ROMADD | 001000 | 170* | 192 | 213 | 245 | 258* | 277 | 320 | | | | | | | | | | | | |
| ROMORG= 166000 | | 1484* | | | | | | | | | | | | | | | | | | |
| RPVEC | 007770 | 1419* | | | | | | | | | | | | | | | | | | |
| RP11 | 007654 | 1375* | 1419 | | | | | | | | | | | | | | | | | |
| RSTRT | 016034 | 1541* | 1658 | 1745 | | | | | | | | | | | | | | | | |
| RC =%000000 | | 523* | 538 | 1307* | 1308* | 1314* | 1316 | 1320 | 1326 | 1381* | 1404* | 1456* | 1468 | | | | | | | |
| R1 =%000001 | | 524* | 539 | 1271* | 1281* | 1283* | 1284* | 1285 | 1290* | 1309* | 1310* | 1312 | 1314 | 1383* | | | | | | |
| | | 1406* | 1457* | 1469 | | | | | | | | | | | | | | | | |
| R2 =%000002 | | 525* | 540 | 1272* | 1274* | 1279* | 1285* | 1289* | 1291 | 1293* | 1294 | 1311* | 1318* | 1320* | | | | | | |

| | | | | | | | | | | | | | | | |
|--------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|------|
| ROL | 418 | 419 | 425 | 426 | 427 | 428 | 429 | 430 | 1017 | 1616 | 1617 | | | | |
| ROLB | 995 | 997 | 1005 | 1007 | 1009 | 1011 | 1018 | 1596 | 1598 | 1605 | 1607 | 1609 | 1611 | | |
| ROR | 1019 | 1021 | 1618 | 1620 | | | | | | | | | | | |
| RORB | 1020 | 1022 | 1619 | 1621 | | | | | | | | | | | |
| RTI | 398 | | | | | | | | | | | | | | |
| RTS | 366 | 437 | 455 | 813 | 820 | 899 | 943 | 952 | 999 | 1014 | 1024 | 1098 | 1097 | 1565 | 1600 |
| | 1614 | 1623 | 1638 | 1666 | 1699 | 1754 | 1761 | 1802 | | | | | | | |
| SUB | 200 | 219 | 730 | 1062 | 1730 | | | | | | | | | | |
| SWAB | 1095 | 1759 | 1800 | | | | | | | | | | | | |
| TRAP | 157 | | | | | | | | | | | | | | |
| TST | 196 | 235 | 261 | 402 | 657 | 977 | 1023 | 1277 | 1326 | 1382 | 1385 | 1389 | 1405 | 1586 | 1622 |
| | 1642 | 1646 | 1675 | 1688 | 1796 | | | | | | | | | | |
| TSTB | 294 | 297 | 322 | 330 | 364 | 369 | 375 | 381 | 421 | 664 | 893 | 901 | 1071 | 1105 | 1287 |
| | 1677 | 1682 | 1693 | 1738 | 1766 | | | | | | | | | | |
| WAIT | 1191 | | | | | | | | | | | | | | |
| .ASCII | 440 | 444 | 445 | | | | | | | | | | | | |
| .BYTE | 1076 | 1109 | 1743 | 1772 | | | | | | | | | | | |
| .DSABL | 1426 | | | | | | | | | | | | | | |
| .ENABL | 146 | 499 | | | | | | | | | | | | | |
| .END | 1850 | | | | | | | | | | | | | | |
| .EVEN | 446 | | | | | | | | | | | | | | |
| .LIST | 222 | 145 | 152 | | | | | | | | | | | | |
| .MLIST | 222 | 152 | | | | | | | | | | | | | |
| .PAGE | 511 | 555 | 618 | 715 | 876 | 1038 | 1133 | 1425 | | | | | | | |
| .REM | 2 | | | | | | | | | | | | | | |
| .REPT | 152 | | | | | | | | | | | | | | |
| .SBTTL | 457 | 510 | 617 | 714 | 875 | 1037 | 1120 | 1263 | 1301 | 1341 | 1424 | | | | |
| .TITLE | 1 | 465 | | | | | | | | | | | | | |
| .WORD | 401 | 648 | 847 | 848 | 849 | 850 | 852 | 853 | 854 | 855 | 856 | 957 | 859 | 860 | 861 |
| | 863 | 867 | 868 | 869 | 870 | 871 | 872 | 873 | 1330 | 1332 | 1334 | 1336 | 1337 | 1338 | 1627 |
| | 1628 | 1629 | 1630 | 1828 | 1829 | 1830 | 1831 | 1832 | 1833 | 1834 | 1835 | 1836 | | | |

ERRORS DETECTED: 0
 DEFAULT GLOBALS GENERATED: 0

*DDGTDB, DDGTDB, SEQ/SOL/CRF/DS:ERFZ/EN:ABS=DSKM:DDGTDB.P11
 RUN-TIME: 4 9 2 SECONDS
 RUN-TIME RATIO: 55/18=3.0
 CORE USED: 8K (15 PAGES)

