

M7142

ACCPT TST
CZVTOA0

AH F208A MC
COPYRIGHT 1979
FICHE 1 OF 1

MAR 1979
digital
MADE IN USA

.REM *

IDENTIFICATION

PRODUCT CODE: AC-F207A-MC
PRODUCT NAME: CZVTOAO M7142 (SERIAL VIDEO BOARD)
ACCEPTANCE TEST
PROGRAM DATE: JANUARY 1979
MAINTAINER: DIAGNOSTICS - MERRIMACK

COPYRIGHT (C) 1979 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
ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

1. ABSTRACT

THIS PROGRAM IS AN ACCEPTANCE TEST OF THE M7142 SERIAL VIDEO BOARD.
THE PROGRAM CONSISTS OF FIVE PARTS, ALL OF WHICH REQUIRE
OPERATOR INSPECTION OR INTERACTION. THE PROGRAM IS CAPABLE
OF HANDLING MULTIPLE UNITS IN A SEQUENTIAL DLV-11 FASHION (REF 8.2).

ONLY ONE M7142 IS TESTED AT ONE TIME.

THE PROGRAM WILL DEFAULT TO THE CONSOLE TTY (REF 5.0 AND 8.2).
ALL CHARACTERS AND COMMANDS ARE TESTED. IN THE KEYBOARD CHARACTER
TEST THE BREAK, AND REPEAT 'FUNCTION' KEYS ARE NOT TESTED.

PART 1 CONSISTS OF A SERIES OF TEST PATTERNS DISPLAYED ON THE
SCREEN (REF 9. FOR DESCRIPTION). THE OPERATOR MUST VISUALLY
INSPECT EACH TEST PATTERN FOR ERROR DETECTION.

PART 2 IS A KEYBOARD CHARACTER TEST. THIS TEST IS TO DETERMINE
THAT THE TERMINAL IS GENERATING THE EXPECTED ASCII CODES.
IN THIS TEST AN OPERATOR WILL BE REQUIRED TO FOLLOW THE
INSTRUCTIONS DISPLAYED ON THE SCREEN AND EXECUTE THEM.
DUE TO THE FLEXIBILITY OF DIFFERENT PROCESSORS OR OPTIONS,
PARITY BIT TESTING MUST BE SELECTED BY THE OPERATOR. THE OPERATOR
SELECTS THE TYPE OF PARITY TO BE TESTED BY SW 00-01

PART 3 IS A KEYBOARD OCTAL VALUE LOOP. WHEN A KEY IS DEPRESSED,
THE OCTAL VALUE WILL BE DISPLAYED ON THE SCREEN.
IF THE KEY DEPRESSED WAS PRINTABLE, IT WILL ALSO BE DISPLAYED
ON THE SCREEN. SOME NON-PRINTING CHARACTERS (SUCH AS CTRL-C, CTRL-D...
OR CR, LF...) HAVE BEEN 'DEFINED' IN THE PROGRAM SO IF ANY OF THESE
KEYS ARE DEPRESSED, IT'S TWO LETTER EQUIVALENT WILL BE DISPLAYED.

PART 4 IS A KEYBOARD ECHO LOOP. WHEN A KEY IS DEPRESSED, THE
CHARACTER IS ECHOED TO THE SCREEN. NO TESTING OF THE CHARACTER
IS PERFORMED. THIS ALLOWS THE OPERATOR A 'LOCAL' MODE OF OPERATION
BETWEEN THE M7142 AND THE HOST COMPUTER.

PART 5 IS A CHARACTER ATTRIBUTE TEST (M.S.B. SELETECTED). A FULL
SCREEN OF FIVE BLOCKS (FIVE LINES EACH) IS PRINTED WITH ALTERNATE
BLOCKS HAVING ANY ENABLED CHARACTER ATTRIBUTES SELECTED BY SETTING THE
MOST SIGNIFICANT BIT (MSB) IN EACH OF THE CHARACTER CODES
AS THEY ARE PRINTED.

THIS TEST IS DESIGNED TO TEST THAT M.S.B.=1 IN THE CHARACTER CODE
WILL CAUSE ANY ENABLED CHARACTER ATTRIBUTES TO BE SELECTED.

101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156

2. REQUIREMENTS

2.1 EQUIPMENT

PDP-11 FAMILY COMPUTER WITH 6K WORDS OF READ/WRITE MEMORY
M7142 SERIAL VIDEO BOARD WITH (SCREEN AND KEYBOARD) CONNECTED
VIA A DLV-11 TYPE INTERFACE.

- "FORM-FEED INIT." MUST BE JUMPER ENABLED.
- "SI/SO CHARACTER ATTRIBUTE SELECTION" MUST BE ENABLED
IN ORDER FOR TEST 'F' TO RUN CORRECTLY.
- "MSB SET IN CHAR. CODE TO SELECT CHAR. ATTRIBUTE" MUST BE
ENABLED IN ORDER FOR TEST 'M' TO RUN CORRECTLY.
- AT LEAST ONE CHAR. ATTRIBUTE MUST BE SELECTED FOR TESTS
'F' AND 'M' TO RUN CORRECTLY.

2.2 STORAGE

THIS PROGRAM USES 8K OF MEMORY.

3. LOADING PROCEDURE

PROCEDURE FOR NORMAL BINARY TAPES SHOULD BE FOLLOWED.

4. STARTING PROCEDURE

4.1 CONTROL SWITCH SETTINGS

STANDARD PDP-11 FORMAT

SW 15 = 1	(100000)	HALT ON ERROR
SW 14 = 1	(040000)	LOOP ON PRESENT TEST
SW 13 = 1	(020000)	INHIBIT ERROR TYPEOUTS
SW 12 = 1	(010000)	INHIBIT PROGRAM SUB-TEST DELAY
SW 10 = 1	(002000)	SKIP KEYBOARD CHARACTER TEST (1)
SW 09 = 1	(001000)	SELECT NON-DEFAULT (NON-CONSOLE)
		ADDRESSES FOR M7142'S BEING TESTED
SW 08 = 1	(000400)	LOOP ON TEST IN SWR <7:0>

KEYBOARD CHARACTER TEST ONLY

SW02 =	1	ENABLE PARITY BIT TEST
SW00-01=	00	EVEN PARITY CHECK
SW00-01=	01	ODD PARITY CHECK
SW00-01=	10	ALWAYS A 0
SW00-01=	11	ALWAYS A 1

SPECIAL NOTE: IF THE COMPUTER UTILIZED IS A LSI 11 OR A COMPUTER
WITHOUT A SWITCH REGISTER, THE PROGRAM WILL UTILIZE LOCATIONS

157
158
159
160
161
162

174 AND 176 AS A "DISPLAY" REGISTER AND A "SWITCH" REGISTER
RESPECTIVELY. THE OPERATOR WILL BE RESPONSIBLE FOR THE
LOADING OF THE "SWITCH" REGISTER LOCATION PRIOR TO STARTING
OR RESTARTING THE PROGRAM.

163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191

4.2 STARTING ADDRESS OR ADDRESSES

200 IS THE STARTING ADDRESS OF THE ACCEPTANCE TEST
204 IS THE RESTART ADDRESS OF THE ACCEPTANCE TEST
210 IS THE STARTING ADDRESS OF THE FULL KEYBOARD CHAR. TEST
214 IS THE STARTING ADDRESS OF THE KEYBOARD OCTAL VALUE LOOP
220 IS THE STARTING ADDRESS OF THE KEYBOARD ECHO LOOP
224 IS THE STARTING ADDRESS OF THE CHARACTER ATTRIBUTE
TEST (M.S.B. SELECTED)

5. OPERATING PROCEDURE

THE OPERATOR MUST INSERT THE CORRECT INFORMATION IN THE SWITCH
REGISTER WHEN REQUIRED BY THE PROGRAM OR AN ERROR WILL OCCUR.
ONCE STARTED, THE TEST WILL RUN IN ITS NORMAL MANNER
WHICH INCLUDES THE EXECUTION OF A 'QUICK' KEYBOARD TEST
THAT REQUIRES OPERATOR INTERVENTION UNLESS SW 10 IS SET
IN THE SWITCH REGISTER. HOWEVER, IF SW 10 IS NOT SET
THE PROGRAM DOES 'TIMEOUT' AND PRINT A 'TIMEOUT MESSAGE' IF
A KEY IS NOT DEPRESSED IN A REASONABLE PERIOD OF TIME.

IF THE M7142 IS NOT THE CONSOLE TERMINAL WITH IT'S FIRST DEVICE
ADDRESS AT 177560, THEN SW 09 MUST BE SET TO TELL THE PROGRAM
WHERE IT IS. SW 09 MUST ALSO BE SET IF MORE THAN ONE M7142
IS TO BE TESTED.

192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242

6. ERRORS

MOST ERRORS MUST BE DETECTED BY VISUAL INSPECTION OF PATTERNS
PRINTED ON THE SCREEN. ONLY THE KEYBOARD TESTS RESULT IN ANY
ERROR REPORTING AND TYPEOUT. REFER TO THE 'ERROR POINTER TABLE'
FOR TYPE AND DESCRIPTION OF ERRORS.
THE ERROR INFORMATION CONSISTS OF THE FOLLOWING:

ERRPC - LOCATION AT WHICH AN ERROR WAS DETECTED
VTNOW - CURRENT DLV-11 BUS ADDRESS OF M7142 UNDER TEST
TSTNUM- TEST NUMBER OF FAILING TEST
EXPT - EXPECTED INPUT CHARACTER
RCVD - RECEIVED INPUT CHARACTER

7. RESTRICTIONS

THE OPERATOR SHOULD SET SW 09 IF THE M7142 UNDER TEST
IS NOT THE CONSOLE TTY, OR NOT AT THE ADDRESS = 177560.

8. MISCELLANEOUS

8.1 EXECUTION TIME

EXECUTION TIME WILL VARY WITH THE 'BAUD' RATE, THE PROGRAM
WILL TYPE 'END PASS' ON THE CONSOLE WHEN A PASS HAS BEEN COMPLETED.
THE KEYBOARD LOOP AND CHARACTER TEST WILL NOT EXIT UNTIL
THE PROGRAM IS RESTARTED.

8.2 DEVICE ADDRESS PROGRAM LOCATIONS (AT APPROX. 1222)

THE LOCATION 'FIRST' CONTAINS THE FIRST DLV11 ADDRESS IF SEVERAL M7142'S
ARE BEING TESTED. THE DEFAULT IS THE CONSOLE ADDRESS <177560>
THE LOCATION 'LAST' CONTAINS THE LAST DLV11 ADDRESS IF SEVERAL M7142'S
ARE BEING TESTED. LOCATION 'VTNOW' CONTAINS THE CURRENT DLV11 BASE
ADDRESS. IF SEVERAL M7142'S ARE BEING TESTED, THEIR ADDRESSES ARE
ASSUMED TO BE 10 OCTAL BYTES APART.

*NOTE: IF THE LOCS. 'FIRST' OR 'LAST' ARE CHANGED, THE OPERATOR MUST
START THE TEST AGAIN AT LOC. 200. THE PROGRAM WILL USE THE BASE
ADDRESS TO UPDATE THE ACTUAL PROGRAM VALUES.

243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286

9. PROGRAM DESCRIPTION <SCREEN>

9.1 A FULL SCREEN OF THE LETTER E

THIS TEST WILL FILL THE SCREEN WITH THE LETTER E.
THIS TEST WILL LOAD THE SCREEN RAM WITH A SINGLE
CHARACTER IN ALL LOCATIONS. THIS TEST IS USED TO
VISUALLY CHECK THE LINEARITY AND CENTERING OF THE
M7142/CRT SYSTEM.

9.2 B SIMPLE CHARACTER SET

ONE FULL LINE OF EACH CHARACTER
(CODES 40 THRU 176) OF THE CHARACTER
SET. THIS WILL ALSO TEST THAT ALL WORDS
IN THE SCREEN RAM CAN BE LOADED.
THIS WILL ALSO EXERCISE THE SCROLLING FUNCTION.

IE: AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA
BB
CC
DDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDDD

9.3 C INCREMENTING AND SLIDING CHARACTER SET

ONE FULL LINE OF AN INCREMENTING
CHARACTER SET ACROSS THE FULL SCREEN
STARTING WITH THE CODE 40 (SPACE)
THE NEXT LINE SHOULD BEGIN WITH THE
"!" CHARACTER (CODE 41) ETC. CONTINUE
THE PATTERN BY INCREMENTING THE
FIRST COLUMN CHARACTER UNTIL THE
FULL CHARACTER SET HAS BEEN EXHAUSTED.

THIS PATTERN WILL VERIFY THAT THE FULL
CHARACTER SET CAN BE DISPLAYED IN EACH
SCREEN WORD.

9.4 D CURSOR MOTION TEST (CR,TAB,LF,BKSP,VT USED)
(FORM-FEED INITIALIZED)

THIS TEST IS THE BASIC CURSOR MOTION TEST USING THE
CARRIAGE-RETURN, TAB, LINE-FEED, BACKSPACE, AND VERTICAL
TAB CAHRACTERS.

THE FIRST PART OF THE TEST PRINTS A LINE OF TEXT,
"CARRIAGE-RETURN'S" BACK TO THE LEFT MARGIN AND THEN
"TAB'S" OVER THE TEXT TO BE SURE A "TAB" DOESN'T
ERASE CHARACTERS.

THE SECOND PART OF THE TEST MOVES THE CURSOR DOWN BY
PRINTING SOME LINE-FEEDS, PRINTS A MESSAGE, AND THEN
BACKSPACES OVER THAT MESSAGE TO BE SURE A BACKSPACE
DOES NOT ERASE CHARACTERS.

THE THIRD PART OF THE TEST DOES A SERIES OF
VERTICAL TABS TO BE SURE THAT A VERTICAL TAB ADVANCES
THE CURSOR DOWN THE SCREEN ONE LINE.

THE FOLLOWING DIAGRAM IS WHAT THE SCREEN SHOULD LOOK LIKE
WHEN THE TEST IS COMPLETE:

```
-----  
+ TAB OVER-NO CHARACTER ERASE <CR><CR> +  
+<LF> +  
+ <LF> +  
+ <LF> +  
+<CR> <CR><LF> BACKSPACE OVER-NO CHAR. ERASE +  
+<VT> +  
+ <VT> +  
+ <VT> +  
+ <VT> +  
+ <VT><CR> +  
+ +  
+ +  
-----
```

NOTE: "FORM-FEED-INIT." FUNCTION MUST BE ENABLED.

331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386

9.5 3 DIRECT CURSOR ADDRESSING (D.C.A.) FORM-FEED INITIALIZED

THIS TEST WILL BE RUN USING THE 'ESC-Y' SEQUENCE.
THIS TEST WILL RANDOMLY FILL THE SCREEN.
THE END RESULT WILL BE A FULL SCREEN OF 25 STATEMENTS,
EACH STATEMENT ROTATED RIGHT ONE CHAR. WITH RESPECT
TO THE LINE ABOVE IT.

NOTE: 'FORM-FEED-INIT.' FUNCTION MUST BE ENABLED.

9.6 F CHARACTER ATTRIBUTE TESTS (SO/SI SELECTED)
(FORM-FEED INITIALIZED)

THIS TEST PRINTS A FULL SCREEN OF FIVE BLOCKS (FIVE LINES EACH)
WITH ALTERNATE BLOCKS HAVING ANY ENABLED CHARACTER ATTRIBUTES
SELECTED USING A 'SO (CNTRL-N)' TO SELECT CHAR. ATTRIBUTES AND
'SI (CNTRL-O)' TO DESELECT CHAR. ATTRIBUTES.

NOTE: THE ABILITY TO SELECT CHAR. ATTRIBUTES USING SO/SI
----- AND AT LEAST ONE CHAR. ATTRIBUTE MUST BE ENABLED.

9.7 G SCROLL SPEED JUMP TESTS (FORM-FEED INITIALIZED)

THIS TEST USES SETS OF VERTICAL TABS TO TEST THE
SCROLLING ABILITIES OF THE M7142. THE SCREEN IS
FIRST FILLED WITH 25 LINES OF SINGLE CHARACTERS ACROSS
ALL COLUMNS. THE TOP LINE WILL CONTAIN ALL A'S AND
THE BOTTOM LINE SHOULD BE A SERIES OF Y'S AND A SCROLL COUNT
AT THE RIGHT END OF THE LAST LINE. THEN 3 SETS OF 25
VERTICAL TABS ARE SENT (WITH A DELAY BETWEEN SETS) TO SEE
IF THE SCREEN SCROLLS CORRECTLY. AFTER EACH SET OF VERTICAL
TABS, THE TOP LINE SHOULD STILL BE ALL A'S, BUT NOW THE BOTTOM
LINE SHOULD BE A SERIES OF Y'S AND THE 'SCROLL-COUNT' WILL BE
INCREMENTED.

9.8 H COLUMN ADDRESS TEST

THIS TEST PRINTS 25 LINES OF A FULL LINE OF AN INCREMENTING
CHARACTER SET. EACH LINE STARTS WITH THE CHARACTER '/',
FOLLOWED BY '0123456789:;<' AND SO ON SO THAT EACH COLUMN ON THE
SCREEN IS FILLED WITH A SINGLE CHARACTER.

THIS PATTERN WILL VERIFY THAT EACH COLUMN IS UNIQUELY ADDRESSABLE.

9.9 I 'QUICK' KEYBOARD CHARACTER TEST

IF SW 10 IS SET (=1), THEN THIS TEST IS NOT EXECUTED.

THIS IS A PARTIAL TEST IF THE KEYBOARD INCLUDED IN EVERY PASS OF THIS ACCEPTANCE PROCEDURE TO PROVIDE A QUICK CHECK OF THE KEYBOARD INTERFACE UNLESS SW 10 IS SET. SW 0 AND 1 HAVE THE SAME USE AS IN THE FULL-BLOWN KEYBOARD CHARACTER TEST. THE FULL-BLOWN TEST SHOULD BE RUN TO ADEQUATELY CHECK THE KEYBOARD. (SEE SECTION 9.10).

ONLY PRINTING CHARACTERS ARE ECHOED AS THE KEYS ARE PRESSED.

9.10 J FULL KEYBOARD CHARACTER TEST

THIS TEST IS DESIGNED TO VERIFY THAT CORRECT CHARACTER CODES AND PARITY BIT ARE GENERATED WHEN A KEY IS DEPRESSED. THIS TEST REQUIRES THE OPERATOR TO EXECUTE THE INSTRUCTIONS DISPLAYED ON THE SCREEN. THE OPERATOR SHOULD ONLY DEPRESS ONE KEY AT A TIME, WITH SOME EXCEPTIONS. THE PROGRAM WILL INFORM THE OPERATOR WHICH ROW TO TEST. ONLY PRINTING CHARACTERS ARE ECHOED AS THE KEYS ARE PRESSED.

IN TESTING THE PARITY BIT, SW 0 AND 1 ARE USED TO INFORM THE PROGRAM OF THE EXPECTED PARITY. AN INCORRECT SWITCH SETTING WILL RESULT IN AN ERROR.

9.11 K KEYBOARD OCTAL VALUE LOOP

THIS LOOP IS PROVIDED TO ENABLE THE OPERATOR TO EXAMINE THE OCTAL VALUE OF A CHARACTER. WHEN A KEY IS DEPRESSED, THE OCTAL VALUE WILL BE DISPLAYED. IF THE CHARACTER WAS A PRINTABLE CHARACTER, IT WILL BE DISPLAYED. THOSE CODES DEFINED AS "CONTROL" WILL BE DISPLAYED AS A TWO LETTER MNEMONIC (IE. DE=DELETE, BL=BELL, CL=CURSOR LEFT ETC.)

9.12 L KEYBOARD ECHO LOOP

WHEN A KEY IS DEPRESSED, THE CHARACTER WILL BE DISPLAYED. NO MODIFICATION OR DATA TEST IS PERFORMED. THIS TEST CAN BE USED TO DETERMINE IF THERE IS A 'UART' OR SERIAL LINE PROBLEM.

9.13 M CHARACTER ATTRIBUTE TEST (M.S.B. OF CHAR. CODE SELECTED)

THIS TEST IS STARTED BY USING A SEPERATE STARTING ADDRESS AND IS DESIGNED TO TEST THAT SETTING THE M.S.B. OF THE CHARACTER CODES SENT TO THE M7142 WILL CAUSE ENABLED CHARACTER ATTRIBUTES TO BE DISPLAYED.

A FULL SCREEN OF FIVE 'BLOCKS' (FIVE LINES EACH) IS PRINTED WITH ALTERNATE BLOCKS HAVING ANY ENABLED CHARACTER ATTRIBUTES

443
444
445
446
447
448
449
450
451

SELECTED USING 'M.S.B.' ON EACH OF THE CHARACTERS AS THEY ARE
PRINTED.

NOTE: THE ABILITY TO SELECT CHARACTER ATTRIBUTES USING
----- THE M.S.B. AND AT LEAST ONE CHAR. ATTRIBUTE MUST BE
ENABLED.

```

452 .TITLE CZVTOAO
453 ;*COPYRIGHT (C) 1978
454 ;*DIGITAL EQUIPMENT CORP.
455 ;*MAYNARD, MASS. 01754
456 ;*
457 ;*PROGRAM BY DIAGNOSTIC ENGINEERING
458 ;*
459 ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
460 ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
461 ;*
462 .SBTTL BASIC DEFINITIONS
463
464 ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
465 001100 STACK= 1100
466 .EQUIV EMT,ERROR ;:BASIC DEFINITION OF ERROR CALL
467 .EQUIV IOT,SCOPE ;:BASIC DEFINITION OF SCOPE CALL
468
469 ;*MISCELLANEOUS DEFINITIONS
470 000011 HT= 11 ;:CODE FOR HORIZONTAL TAB
471 000012 LF= 12 ;:CODE FOR LINE FEED
472 000015 CR= 15 ;:CODE FOR CARRIAGE RETURN
473 000200 CRLF= 200 ;:CODE FOR CARRIAGE RETURN-LINE FEED
474 177776 PS= 177776 ;:PROCESSOR STATUS WORD
475 .EQUIV PS,PSW
476 177774 STKLMT= 177774 ;:STACK LIMIT REGISTER
477 177772 PIRQ= 177772 ;:PROGRAM INTERRUPT REQUEST REGISTER
478 177570 DSWR= 177570 ;:HARDWARE SWITCH REGISTER
479 177570 DDISP= 177570 ;:HARDWARE DISPLAY REGISTER
480
481 ;*GENERAL PURPOSE REGISTER DEFINITIONS
482 000000 R0= 0 ;:GENERAL REGISTER
483 000001 R1= 1 ;:GENERAL REGISTER
484 000002 R2= 2 ;:GENERAL REGISTER
485 000003 R3= 3 ;:GENERAL REGISTER
486 000004 R4= 4 ;:GENERAL REGISTER
487 000005 R5= 5 ;:GENERAL REGISTER
488 000006 R6= 6 ;:GENERAL REGISTER
489 000007 R7= 7 ;:GENERAL REGISTER
490 000006 SP= 6 ;:STACK POINTER
491 000007 PC= 7 ;:PROGRAM COUNTER
492
493 ;*PRIORITY LEVEL DEFINITIONS
494 000000 PR0= 0 ;:PRIORITY LEVEL 0
495 000040 PR1= 40 ;:PRIORITY LEVEL 1
496 000100 PR2= 100 ;:PRIORITY LEVEL 2
497 000140 PR3= 140 ;:PRIORITY LEVEL 3
498 000200 PR4= 200 ;:PRIORITY LEVEL 4
499 000240 PR5= 240 ;:PRIORITY LEVEL 5
500 000300 PR6= 300 ;:PRIORITY LEVEL 6
501 000340 PR7= 340 ;:PRIORITY LEVEL 7
502
503 ;*'SWITCH REGISTER' SWITCH DEFINITIONS
504 100000 SW15= 100000
505 040000 SW14= 40000
506 020000 SW13= 20000
507 010000 SW12= 10000

```

508	004000	SW11=	4000
509	002000	SW10=	2000
510	001000	SW09=	1000
511	000400	SW08=	400
512	000200	SW07=	200
513	000100	SW06=	100
514	000040	SW05=	40
515	000020	SW04=	20
516	000010	SW03=	10
517	000004	SW02=	4
518	000002	SW01=	2
519	000001	SW00=	1
520		.EQUIV	SW09,SW9
521		.EQUIV	SW08,SW8
522		.EQUIV	SW07,SW7
523		.EQUIV	SW06,SW6
524		.EQUIV	SW05,SW5
525		.EQUIV	SW04,SW4
526		.EQUIV	SW03,SW3
527		.EQUIV	SW02,SW2
528		.EQUIV	SW01,SW1
529		.EQUIV	SW00,SW0

530
 531 ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)

532	100000	BIT15=	100000
533	040000	BIT14=	40000
534	020000	BIT13=	20000
535	010000	BIT12=	10000
536	004000	BIT11=	4000
537	002000	BIT10=	2000
538	001000	BIT09=	1000
539	000400	BIT08=	400
540	000200	BIT07=	200
541	000100	BIT06=	100
542	000040	BIT05=	40
543	000020	BIT04=	20
544	000010	BIT03=	10
545	000004	BIT02=	4
546	000002	BIT01=	2
547	000001	BIT00=	1
548		.EQUIV	BIT09,BIT9
549		.EQUIV	BIT08,BIT8
550		.EQUIV	BIT07,BIT7
551		.EQUIV	BIT06,BIT6
552		.EQUIV	BIT05,BIT5
553		.EQUIV	BIT04,BIT4
554		.EQUIV	BIT03,BIT3
555		.EQUIV	BIT02,BIT2
556		.EQUIV	BIT01,BIT1
557		.EQUIV	BIT00,BIT0

558
 559 ;*BASIC "CPU" TRAP VECTOR ADDRESSES

560	000004	ERRVEC=	4	::TIME OUT AND OTHER ERRORS
561	000010	RESVEC=	10	::RESERVED AND ILLEGAL INSTRUCTIONS
562	000014	TBITVEC=	14	::'T' BIT
563	000014	TRTVEC=	14	::TRACE TRAP

```

564      000014      BPTVEC= 14      ;;BREAKPOINT TRAP (BPT)
565      000020      IOTVEC= 20      ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
566      000024      PWRVEC= 24      ;;POWER FAIL
567      000030      EMTVEC= 30      ;;EMULATOR TRAP (EMT) **ERROR**
568      000034      TRAPVEC=34      ;;"TRAP" TRAP
569      000060      TKVEC= 60      ;;TTY KEYBOARD VECTOR
570      000064      TPVEC= 64      ;;TTY PRINTER VECTOR
571      000240      PIRQVEC=240     ;;PROGRAM INTERRUPT REQUEST VECTOR
572
573      .SBTTL  OPERATIONAL SWITCH SETTINGS
574      ;*
575      ;*      SWITCH      USE
576      ;*      -----
577      ;*      15      HALT ON ERROR
578      ;*      14      LOOP ON TEST
579      ;*      13      INHIBIT ERROR TYPEOUTS
580      ;*      12      INHIBIT SUB-TEST DELAY'S
581      ;*      10      SKIP KEYBOARD CHARACTER TEST
582      ;*      9       SELECT NON-CONSOLE ADDRESS
583      ;*      8       LOOP ON TEST IN SWR<7:0>
584
585      .SBTTL  TRAP CATCHER
586
587      000000      .=0
588      ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
589      ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
590      ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
591      000174      .=174
592      000174      000000      DISPREG: .WORD 0      ;;SOFTWARE DISPLAY REGISTER
593      000176      000000      SWREG: .WORD 0      ;;SOFTWARE SWITCH REGISTER
594
595      .SBTTL  STARTING ADDRESS(ES)
596      000200      000137      001274      JMP @NBEGIN ;;JUMP TO STARTING ADDRESS OF PROGRAM
597      000204      000137      001314      JMP RBEGIN ;;JUMP TO RESTART ADDRESS
598      000210      000137      001340      JMP BEGIN1 ;;JUMP TO FULL KEYBOARD CHAR. TEST
599      000214      000137      001360      JMP BEGIN2 ;;JUMP TO CHAR OCTAL VALUE LOOP
600      000220      000137      001330      JMP BEGIN3 ;;JUMP TO ASCII ECHO LOOP
601      000224      000137      001350      JMP BEGIN4 ;;JUMP TO CHAR. ATTRIB. TEST (BIT7 SELECT)
602
603      .SBTTL  ACT11 HOOKS
604
605      ;*****
606      ;HOOKS REQUIRED BY ACT11
607      $SVPC=.      ;SAVE PC
608      .=46      ;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SEOP
609      $ENDAD      ;2)SET LOC.52 TO ZERO
610      .=52      ;; RESTORE PC
611      .WORD 0
612      .=$SVPC
  
```

```

611      .SBTTL  COMMON TAGS
612
613      ;*****
614      ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
615      ;*USED IN THE PROGRAM.
616
617      001100      .=1100
618      001100      $CMTAG:      ;:START OF COMMON TAGS
619      001100      000000      $PASS:      .WORD      0      ;:CONTAINS PASS COUNT
620      001102      000      $TSTNM:      .BYTE      0      ;:CONTAINS THE TEST NUMBER
621      001103      000      $RFLG:      .BYTE      0      ;:CONTAINS ERROR FLAG
622      001104      000000      $ICNT:      .WORD      0      ;:CONTAINS SUBTEST ITERATION COUNT
623      001106      000000      $LPADR:      .WORD      0      ;:CONTAINS SCOPE LOOP ADDRESS
624      001110      000000      $LPERR:      .WORD      0      ;:CONTAINS SCOPE RETURN FOR ERRORS
625      001112      000000      $ERTTL:      .WORD      0      ;:CONTAINS TOTAL ERRORS DETECTED
626      001114      000      $ITEMB:      .BYTE      0      ;:CONTAINS ITEM CONTROL BYTE
627      001115      001      $ERMAX:      .BYTE      1      ;:CONTAINS MAX. ERRORS PER TEST
628      001116      000000      $ERRPC:      .WORD      0      ;:CONTAINS PC OF LAST ERROR INSTRUCTION
629      001120      000000      $GDADR:      .WORD      0      ;:CONTAINS ADDRESS OF 'GOOD' DATA
630      001122      000000      $BDADR:      .WORD      0      ;:CONTAINS ADDRESS OF 'BAD' DATA
631      001124      000000      $GDDAT:      .WORD      0      ;:CONTAINS 'GOOD' DATA
632      001126      000000      $BDDAT:      .WORD      0      ;:CONTAINS 'BAD' DATA
633      001130      000000      .WORD      0      ;:RESERVED--NOT TO BE USED
634      001132      000000      .WORD      0
635      001134      000      $AUTOB:      .BYTE      0      ;:AUTOMATIC MODE INDICATOR
636      001135      000      $INTAG:      .BYTE      0      ;:INTERRUPT MODE INDICATOR
637      001136      000000      .WORD      0
638      001140      177570      $SWR:      .WORD      DSWR      ;:ADDRESS OF SWITCH REGISTER
639      001142      177570      $DISPLAY:      .WORD      DDISP      ;:ADDRESS OF DISPLAY REGISTER
640      001144      177560      $TKS:      177560      ;:TTY KBD STATUS
641      001146      177562      $TKB:      177562      ;:TTY KBD BUFFER
642      001150      177564      $TPS:      177564      ;:TTY PRINTER STATUS REG. ADDRESS
643      001152      177566      $TPB:      177566      ;:TTY PRINTER BUFFER REG. ADDRESS
644      001154      000      $NULL:      .BYTE      0      ;:CONTAINS NULL CHARACTER FOR FILLS
645      001155      002      $FILLS:      .BYTE      2      ;:CONTAINS # OF FILLER CHARACTERS REQUIRED
646      001156      012      $FILLC:      .BYTE      12      ;:INSERT FILL CHARS. AFTER A 'LINE FEED'
647      001157      000      $TPFLG:      .BYTE      0      ;:'TERMINAL AVAILABLE' FLAG (BIT<07>=0=YES)
648      001160      000000      $REGAD:      .WORD      0      ;:CONTAINS THE ADDRESS FROM
649      ;:WHICH ($REGO) WAS OBTAINED
650      001162      000000      $REGO:      .WORD      0      ;:CONTAINS (($REGAD)+0)
651      001164      000000      $REG1:      .WORD      0      ;:CONTAINS (($REGAD)+2)
652      001166      077      $QUES:      .ASCII      /?/      ;:QUESTION MARK
653      001167      015      $CRLF:      .ASCII      <15>      ;:CARRIAGE RETURN
654      001170      000012      $LF:      .ASCII      <12>      ;:LINE FEED
655      ;*****

```

```

656 .SBTTL ERROR POINTER TABLE
657
658 ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
659 ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
660 ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
661 ;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
662 ;*NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
663
664 ;* EM ;:POINTS TO THE ERROR MESSAGE
665 ;* DH ;:POINTS TO THE DATA HEADER
666 ;* DT ;:POINTS TO THE DATA
667 ;* DF ;:POINTS TO THE DATA FORMAT
668
669
670 001172 $ERRTB:
671
672
673 ;ITEM 1
674 001172 011665 EM1 ;ERROR FLAG ON HOST TRANSMIT STATUS
675 001174 012024 DH1 ;ERRPC VTNOW TSTNUM
676 001176 012122 DT1 ;$ERRPC VTNOW TSTNUM
677 001200 000000 0
678
679 ;ITEM 2
680 001202 011732 EM2 ;NO HOST INPUT FLAG RECEIVED
681 001204 012024 DH1 ;ERRPC VTNOW TSTNUM
682 001206 012122 DT1 ;$ERRPC VTNOW TSTNUM
683 001210 000000 0
684
685 ;ITEM 3
686 001212 011761 EM3 ;INCORRECT OR UNEXPECTED INPUT CHARACTER
687 001214 012053 DH3 ;ERRPC VTNOW TSTNUM EXPT RCVD
688 001216 012132 DT3 ;$ERRPC VTNOW TSTNUM $GDDAT $BDDAT
689 001220 000000 0
690
691 FIRST: 177560 ;FIRST DEVICE ADDRESS OF SEQUENTIAL DLV-11-A/B TYPE DEVI
692 001222 177560 ;DEFAULT TO THE CONSOLE ADDRESS
693
694 001224 000000 LAST: 0 ;LAST DEVICE ADDRESS OF DLV-11-A/B TYPE
695 001226 177560 VTNOW: 177560 ;CURRENT DEVICE BUS ADDRESS
696 001230 000000 TSTNUM: 0 ;ERROR PATTERN
697
698 001232 000015 TIME0: 15 ;CHARACTER FLAG TIMEOUT CONSTANT
699 001234 000005 SUBTST: 5 ;SUBTEST DELAY CONSTANT
700 001236 000070 LASTLN: 70 ;LAST VALID LINE # +40
701 001240 003720 TOTALC: 2000. ;TOTAL CHARACTER COUNT
702 001242 000031 VHO: 25. ;VERTICAL LINE COUNT
703 001244 000014 VH1: 12. ;1/2 VERTICAL LINE COUNT
704 001246 000006 VH2: 6. ;1/4 VERTICAL LINE COUNT
705 001250 000000 PRICNT: 0
706 001252 177560 VTIS: 177560 ;DEVICE ADDRESSES
707 001254 177562 VTIB: 177562 ;IN DATA
708 001256 177564 VTOS: 177564 ;OUT STAT
709 001260 177566 VTOB: 177566 ;OUT DATA
710 001262 000000 STCHAR: 0 ;TEMP REG'S
711 001264 000200 LASTCH: 200 ;FIRST NON-VALID CHARACTER
  
```

```

712 001266 000000      TEMPO: 0
713 001270 000120      WIDTH: 80.
714 001272 000000      SAVE4: 0
715
716
717 001274 012737 002112 002110 BEGIN: MOV #TST1,WHERE ;STARTING ACCEPTANCE TEST ADDRESS
718 001302 005037 001272      CLR SAVE4
719 001306 005037 001250      CLR PRTCNT
720 001312 000430      BR GINA
721 001314 005037 001250 RBEGIN: CLR PRTCNT ;RESTART ADDRESS
722 001320 012737 002112 002110 MOV #TST1,WHERE
723 001326 000417      BR GIN
724 001330 012737 004606 002110 BEGIN3: MOV #KRBECH,WHERE ;START AT ECHO LOOP
725 001336 000413      BR GIN
726 001340 012737 004052 002110 BEGIN1: MOV #KRBTST,WHERE ;STARTING KEYBOARD CHAR. TEST ADDRESS
727 001346 000407      BR GIN
728 001350 012737 004644 002110 BEGIN4: MOV #CHRATT,WHERE ;STARTING CHARACTER ATTRIBUTE (BIT8) ADDRESS
729 001356 000403      BR GIN
730 001360 012737 004316 002110 BEGIN2: MOV #KRBECH,WHERE ;STARTING CHARACTER LOOP ADDRESS
731 001366 012737 000001 001272 GIN: MOV #1,SAVE4
732 001374 000005      GINA: RESET
733
734      .SBTTL INITIALIZE THE COMMON TAGS
735      .:CLEAR THE COMMON TAGS ($CMTAG) AREA
736      MOV # $CMTAG,R6 ;:FIRST LOCATION TO BE CLEARED
737      CLR (R6)+ ;:CLEAR MEMORY LOCATION
738      CMP #SWR,R6 ;:DONE?
739      BNE .-6 ;:LOOP BACK IF NO
740      MOV #STACK,SP ;:SETUP THE STACK POINTER
741      .:INITIALIZE A FEW VECTORS
742      MOV # $SCOPE,@#IOTVEC ;:IOT VECTOR FOR SCOPE ROUTINE
743      MOV #340,@#IOTVEC+2 ;:LEVEL 7
744      MOV # $ERROR,@#EMTVEC ;:EMT VECTOR FOR ERROR ROUTINE
745      MOV #340,@#EMTVEC+2 ;:LEVEL 7
746      MOV # $TRAP,@#TRAPVEC ;:TRAP VECTOR FOR TRAP CALLS
747      MOV #340,@#TRAPVEC+2 ;:LEVEL 7
748      MOV # $PWRDN,@#PWRVEC ;:POWER FAILURE VECTOR
749      MOV #340,@#PWRVEC+2 ;:LEVEL 7
750      MOV #ENDCT,SEOPCT ;:SETUP END-OF-PROGRAM COUNTER
751      MOV #.,SLPADR ;:INITIALIZE THE LOOP ADDRESS FOR SCOPE
752      .:SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
753      .:EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
754      MOV @PERRVEC,-(SP) ;:SAVE ERROR VECTOR
755      MOV #64$,@#ERRVEC ;:SET UP ERROR VECTOR
756      MOV #DSWR,SWR ;:SETUP FOR A HARDWARE SWICH REGISTER
757      MOV #DDISP,DISPLAY ;:AND A HARDWARE DISPLAY REGISTER
758      CMP #-1,DSWR ;:TRY TO REFERENCE HARDWARE SWR
759      BNE 66$ ;:BRANCH IF NO TIMEOUT TRAP OCCURRED
760      BR 65$ ;:AND THE HARDWARE SWR IS NOT = -1
761      BR 65$ ;:BRANCH IF NO TIMEOUT
762      MOV #65$,(SP) ;:SET UP FOR TRAP RETURN
763      MOV #SWREG,SWR ;:POINT TO SOFTWARE SWR
764      MOV #DISPREG,DISPLAY
765      MOV (SP)+,@#ERRVEC ;:RESTORE ERROR VECTOR
766
767 001600 005037 005172      CLR IGNORE
  
```

768	001604	005037	005154		CLR	LOOP	
769	001610	012737	014532	000020	MOV	#MSCOPE,@#10TVEC	
770	001616	005737	001272		TST	SAVE4	;TEST FLAG
771	001622	001052			BNE	SBEGIN	;BR IF NON-ZERO
772	001624	104401			TYPE		
773	001626	006426			TITLE		
774	001630	032777	001000	177302	BIT	#B119,@SWR	
775	001636	001444			BEQ	SBEGIN	;BR IF CLEARED
776	001640	012737	001740	000004	MOV	#12\$,4	;SET UP SLAVE TIMEOUT
777	001646	012737	000340	000006	MOV	#340,6	;LOCK OUT ANY INTERRUPTS
778	001654	104401			11\$: TYPE		
779	001656	011520			WHAT0		;FIND OUT THE DEVICE ADDRESS
780	001660	104410			RDOCT		
781	001662	012637	001222		MOV	(SP)+,FIRST	;SAVE THE ADDRESS
782	001666	022737	160000	001222	CMP	#160000,FIRST	
783	001674	101367			BHI	11\$	;BR IF INVALID
784	001676	005777	177320		TST	@FIRST	;TEST IF VALID
785	001702	005037	001224		CLR	LAST	
786	001706	104401			TYPE		
787	001710	011563			WHAT1		;FIND OUT THE LAST ADDRESS
788	001712	104410			RDOCT		
789	001714	012637	001224		MOV	(SP)+,LAST	;SAVE LAST ADDRESS
790	001720	005777	177300		TST	@LAST	;TEST IF VALID
791	001724	012737	000006	000004	MOV	#6,@#4	
792	001732	005037	000006		CLR	@#6	
793	001736	000404			BR	SBEGIN	
794	001740	022626			12\$: CMP	(SP)+,(SP)+	;ADJUST STACK FROM TIMEOUT
795	001742	104401			TYPE		
796	001744	010303			INVLID		;INVALID ADDRESS MESSAGE
797	001746	000742			BR	11\$	;LET OPERATOR HAVE AND TRY
798	001750	013737	001222	001226	SBEGIN: MOV	FIRST,VINOW	;LOAD INITIAL DEVICE ADDRESS
799							
800	001756	012700	001252		RSTRT: MOV	#VTIS,R0	;LOAD POINTER
801	001762	013701	001226		MOV	VINOW,R1	;LOAD INPUT STAT
802	001766	010120			MOV	R1,(R0)+	
803	001770	005721			TST	(R1)+	
804	001772	010120			MOV	R1,(R0)+	
805	001774	005721			TST	(R1)+	
806	001776	010120			MOV	R1,(R0)+	
807	002000	005721			TST	(R1)+	
808	002002	010110			MOV	R1,(R0)	

```

809
810
811
812 002004 012737 003720 001240 13$: MOV #2000.,TOTALC ;LOAD TOTAL CHARACTER COUNT
813 002012 012737 000031 001242 MOV #25.,VH0 ;LOAD MAX VERTICAL LINE COUNT
814 002020 012737 000070 001236 MOV #70,LASTLN ;LOAD LAST LINE VALUE +40 FOR DCA TESTING
815
816 002026 013737 001242 001244 6$: MOV VH0,VH1 ;LOAD OTHER LINE COUNTS
817 002034 006237 001244 ASR VH1
818 002040 013737 001244 001246 MOV VH1,VH2
819 002046 006237 001246 ASR VH2
820 002052 012737 000177 001264 MOV #177,LASTCH ;LOAD FIRST NON-VALID CHARACTER
821
822 002060 004537 005340 12$: JSR R5,AMSG ;INIT THE DEVICE WITH A FORM-FEED
823 002064 011205 FRMFED
824 002066 004737 005266 JSR PC,ADelay ;WAIT FOR FORM-FEED TO FINISH
825 002072 004537 005340 JSR R5,AMSG ;PRINT TITLE OF DIAGNOSTIC
826 002076 006426 TITLE
827 002100 004737 005214 JSR PC,DELAY
828 002104 000177 000000 JMP @WHERE
829 002110 002112 WHERE: TST1
830
831
832 ;*****
833 ;*TEST 1 A FULL SCREEN OF A CHARACTER
834 ;*****
835 002112 000004 TST1: SCOPE
836
837 002114 004537 005340 JSR R5,AMSG
838 002120 006516 M91
839 002122 012701 000105 FILLWC: MOV #'E,R1 ;LOAD CHARACTER BYTE
840 002126 004737 004754 JSR PC,FILBUF ;LOAD THE LINE WITH CHAR
841 002132 013737 001242 002162 MOV VH0,10$ ;LOAD COUNT
842 002140 012737 000001 005160 1$: MOV #1,XOFFOK ;INSURE XOFF/XON CONTROL.
843 002146 004737 005056 JSR PC,XPRNT ;DISPLAY THE LINE
844 002152 005337 002162 DEC 10$
845 002156 001370 BNE 1$ ;LOOP UNTIL DONE
846 002160 000401 BR 11$ ;EXIT
847 002162 000000 10$: 0
848 002164 004737 005214 11$: JSR PC,DELAY
849
850
851
852 ;*****
853 ;*TEST 2 B SINGLE CHARACTER ACROSS ALL COLS. <ALL CHARACTERS>
854 ;*****
855 002170 000004 TST2: SCOPE
856 002172 004537 005340 JSR R5,AMSG ;DISPLAY HEADING
857 002176 006566 M93
858 002200 012737 000040 001262 MOV #40,STCHAR ;SET-UP STARTING CHARACTER
859 002206 013737 001242 001266 MOV VH0,TEMPO ;LOAD COUNT
860 002214 013701 001262 1$: MOV STCHAR,R1 ;LOAD R1= TO CHARACTER
861 002220 004737 004754 JSR PC,FILBUF ;LOAD A BUFFER WITH THAT CHARACTER
862 002224 004737 005056 JSR PC,XPRNT ;DISPLAY A FULL LINE FROM THE BUFFER
863 002230 005337 001266 DEC TEMPO ;DONE ?
864 002234 001005 BNE 2$ ;FINISHED

```

```

865 002236 004737 005214 JSR PC,DELAY
866 002242 013737 001242 001266 MOV VH0,TEMPO
867 002250 005237 001262 2$: INC STCHAR ;UPDATE THE CHARACTER
868 002254 023737 001264 001262 CMP LASTCH,STCHAR ;TEST FOR FINAL CHARACTER
869 002262 001354 BNE 1$ ;BRANCH IF NOT COMPLETED
870 002264 004737 005214 JSR PC,DELAY ;TEST DELAY SWITCH
871
872
873
874
875
876 ;*****
877 ;*TEST 3 C ROTATING CHARACTERS ACROSS ALL COLS. <ALL CHARACTERS>
878 ;*****
878 002270 000004 TST3: SCOPE
879 002272 004537 005340 JSR R5,AMSG ;DISPLAY HEADING
880 002276 006632 M94
881 002300 012737 000040 001262 MOV #40,STCHAR ;SET-UP STARTING CHARACTER
882 002306 013737 001242 001266 MOV VH0,TEMPO ;LOAD TEMP
883 002314 013701 001262 1$: MOV STCHAR,R1 ;LOAD R1=TO CHARACTER
884 002320 004537 005010 JSR R5,LIC ;LOAD A BUFFER STARTING WITH
885 002324 001270 WIDTH ; THAT CHARACTER AND WIDTH <BYTE>
886 002326 004737 005056 JSR PC,XPRNT ;DISPLAY A FULL LINE FROM THE BUFFER
887 002332 005337 001266 DEC TEMPO ;DONE ?
888 002336 001005 BNE 2$ ;BR IF YES
889 002340 004737 005214 JSR PC,DELAY
890 002344 013737 001242 001266 MOV VH0,TEMPO
891 002352 005237 001262 2$: INC STCHAR ;UPDATE THE STARTING CHARACTER
892 002356 023737 001264 001262 CMP LASTCH,STCHAR ;TEST FOR FINAL CHARACTER
893 002364 001353 BNE 1$ ;BRANCH IF NOT COMPLETED
894 002366 004737 005214 JSR PC,DELAY ;TEST DELAY SWITCH
895
896
897 ;*****
898 ;*TEST 4 D CURSOR MOTION TEST (FORM-FEED INIT USED)
899 ;*****
900 002372 000004 TST4: SCOPE
901 002374 004537 005340 JSR R5,AMSG ;ERASE-SCREEN (FORM-FEED INIT.)
902 002400 011205 FRMFED
903 002402 004737 005266 JSR PC,ADelay ;WAIT FOR FORM-FEED INIT TO FINISH
904 002406 004537 005340 JSR R5,AMSG ;DISPLAY HEADING
905 002412 006665 M96
906
907 002414 012700 022066 MOV TAB: MOV #BUFFER,R0 ;LOAD POINTER TO BUFFER
908 002420 012701 012540 MOV #MTAB,R1 ;LOAD POINTER TO TAB MESSAGE
909 002424 112120 1$: MOV (R1)+,(R0)+ ;PUT TAB MESSAGE INTO BUFFER
910 002426 100376 BPL 1$
911 002430 004737 005056 JSR PC,XPRNT ;DISPLAY THIS LINE
912 002434 004737 005214 JSR PC,DELAY ;TEST DELAY SWITCH
913
914 002440 012700 022066 MOV BSP: MOV #BUFFER,R0 ;LOAD POINTER TO BUFFER
915 002444 012701 012654 MOV #MBKSP,R1 ;LOAD POINTER TO BACKSPACE MESSAGE
916 002450 112120 2$: MOV (R1)+,(R0)+ ;PUT MESSAGE INTO BUFFER
917 002452 100376 BPL 2$
918 002454 004737 005056 JSR PC,XPRNT ;DISPLAY THIS LINE
919 002460 004737 005214 JSR PC,DELAY ;TEST DELAY SWITCH
920

```

```

921 002464 012700 022066      MOVVT:  MOV    #BUFFER,RO      ;LOAD POINTER TO BUFFER
922 002470 012701 013006      MOV    #MVT,R1        ;LOAD POINTER TO VERT. TAB MESSAGE
923 002474 112120              3$:  MOV    (R1)+,(RO)+    ;PUT VERT. TAB MESSAGE INTO BUFFER
924 002476 100376              BPL     3$
925 002500 004737 005056      JSR     PC,XPRNT        ;DISPLAY THIS LINE
926 002504 004737 005214      JSR     PC,DELAY        ;TEST DELAY SWITCH
927
928
929
930
931
932
933
934
935 002510 000004              ::*****
          ;*TEST 5          E      DIRECT CURSOR ADDRESS TEST (FORM-FEED INIT USED)
          ;*****
936
937
938
939
940
941
942
943 002512 004537 005340      4$:  JSR     R5,AMSG
944 002516 011205              FRMFED
945 002520 004737 005266      JSR     PC,ADELAY        ;WAIT FOR FORM-FEED INIT TO FINISH
946 002524 004537 005340      JSR     R5,AMSG        ;IDENTIFY TEST
947 002530 011136              M98
948 002532 112737 000033 022066      MOV    #33,BUFFER    ;LOAD 'ESC' CODE
949 002540 112737 000131 022067      MOV    #'Y,BUFFER+1    ;LOAD ASCII 'Y' (D.C.A. ENABLE)
950
951 002546 012737 123456 015702      DCATST: MOV    #123456,$LONUM    ;PRIME RANDOM NUMBER GENERATOR
952 002554 012737 176543 015700      MOV    #176543,$HINUM
953 002562 013737 001240 003030      MOV    TOTALC,OVRAL    ;LOAD CHAR COUNT
954 002570 012700 022106      MOV    #BUFFER+20,RO    ;LOAD DESTINATION BUFFER
955 002574 012701 016142      2$:  MOV    #MSGTXT,R1    ;LOAD MESSAGE POINTER
956 002600 012120              1$:  MOV    (R1)+,(RO)+    ;LOAD 2 CHARACTERS
957 002602 122711 000377      CMPB    #377,(R1)        ;TEST FOR LAST CHAR
958 002606 001374              BNE     1$              ;BR UNTIL DONE
959 002610 012737 177777 022072      MOV    #-1,BUFFER+4    ;LOAD MESSAGE TERMINATOR (IN 4&5)
960
961 002616 004737 015602      GENER:  JSR     PC,$RAND        ;GENERATE RANDOM NUMBER
962 002622 013700 015700      MOV    $HINUM,RO        ;GET RANDOM NUMBER
963 002626 042700 177700      BIC     #177700,RO        ;RANDOM NO. MUST BE TWO DIGITS
964 002632 020027 000037      CMP     RO,#37          ;NO, MUST BE LESS THAN 40
965 002636 101767              BLOS    GENER          ;LOWER, REGENERATION
966 002640 020037 001236      CMP     RO,LASTLN        ;NO, MUST NOT BE GREATER THAN 70
967 002644 101364              BHI     GENER          ;GREATER, REGENERATION
968 002646 010037 003022      MOV     RO,YADDS        ;STORE RANDOM Y COORDINATE
969 002652 010001              MOV     RO,R1          ;COPY DATA
970 002654 012737 022106 003026      MOV     #BUFFER+20,SET    ;LOAD BASE POINTER
971 002662 162701 000040      SUB     #40,R1          ;MINIMUM Y INDEX
972 002666 001405              BEQ     GENRX          ;RESULT, MINIMUM Y COORDINATE
973 002670 062737 000120 003026  1$:  ADD     #80.,SET    ;SETUP Y INDEX LOCAT'ON FOR PRINTOUT
974 002676 005301              DEC     R1
975 002700 001373              BNE     1$              ;Y COORDINATE IS SET
976 002702 004737 015602      GENRX:  JSR     PC,$RAND        ;GENERATE RANDOM NUMBER

```

977	002706	013700	015700	MOV	\$HINUM,RO	;GET A RANDOM NUMBER
978	002712	042700	177600	BIC	#177600,RO	;RANDOM NO. MAY BE LESS THAN 200
979	002716	020027	000037	CMP	RO,#37	; MUST NOT BE LESS THAN 40
980	002722	101767		BLOS	GENRX	;LOWER, REGENERATION
981	002724	020027	000157	CMP	RO,#157	; MUST NOT BE GREATER THAN 157
982	002730	101364		BHI	GENRX	;GREATER, REGENERATION
983	002732	010037	003024	MOV	RO,XADDS	;STORE RANDOM X COORDINATE
984	002736	162700	000040	SUB	#40,RO	;SETUP MINIMUM X INDEX
985	002742	060037	003026	ADD	RO,SET	;SETUP X COOR, FOR PNTOUT.
986	002746	013701	003026	MOV	SET,R1	;SETUP CHECK
987	002752	105711		TSTB	(R1)	;HAS CURRENT CHAR, ALREADY BEEN USED?
988	002754	001720		BEQ	GENER	;YES, REGENERATE
989	002756	113737	003022	MOVB	YADDS,BUFFER+2	;LOAD Y COORDINATE
990	002764	113737	003024	MOVB	XADDS,BUFFER+3	;LOAD X COORDINATE
991	002772	111137	022072	MOVB	(R1),BUFFER+4	;LOAD CHARACTER TO BE PRINTED
992	002776	105011		CLRB	(R1)	;INDICATE USE OF CURSOR POSITION
993	003000	004737	005056	JSR	PC,XPRNT	;EXECUTE AND PRINT CHARACTER
994	003004	005337	003030	DEC	OVRL	;MAXIMUM NO. OF COORDINATES
995	003010	001302		BNE	GENER	;BR BACK UNTIL DONE
996						
997	003012	004737	005214	JSR	PC,DELAY	;DELAY TESTING
998	003016	000137	003032	JMP	TST6	;SKIP OVER STORAGE LOCS. TO NEXT TEST
999						

1000	003022	000000	YADDS: 0
1001	003024	000000	XADDS: 0
1002	003026	000000	SET: 0
1003	003030	000000	OVRL: 0
1004			
1005			
1006			
1007			
1008			
1009			
1010			
1011			

```

*****
; *TEST 6      F      CHARACTER ATTRIBUTE TEST (SI-SO SELECT)
*****
TST6:  SCOPE

```

1012	003032	000004				
1013						
1014	003034	004537	005340	JSR	R5,AMSG	;ERASE AND INIT SCREEN WITH FORM-FEED
1015	003040	011205		FRMFED		
1016	003042	004737	005266	JSR	PC,ADELAY	;WAIT FOR FORM-FEED TO FINISH
1017	003046	004537	005340	JSR	R5,AMSG	;DISPLAY HEADER
1018	003052	007043		M9221		
1019	003054	004537	005340	JSR	R5,AMSG	;SELECT ANY ENABLED CHAR. ATTRIBUTES
1020	003060	011512		SO		
1021	003062	004737	005174	JSR	PC,BLKPRT	;GO PRINT ATTR. MSG. FIVE TIMES
1022	003066	004537	005340	JSR	R5,AMSG	;DESELECT ANY CHAR. ATTRIBUTES
1023	003072	011515		SI		
1024	003074	004737	005174	JSR	PC,BLKPRT	;GO PRINT ATTR. MSG FIVE TIMES
1025	003100	004537	005340	JSR	R5,AMSG	;SELECT ANY ENABLED CHAR. ATTRIBUTES
1026	003104	011512		SO		
1027	003106	004737	005174	JSR	PC,BLKPRT	;GO PRINT ATTR. MSG. FIVE TIMES
1028	003112	004537	005340	JSR	R5,AMSG	;DESELECT ANY CHAR. ATTRIBUTES
1029	003116	011515		SI		
1030	003120	004737	005174	JSR	PC,BLKPRT	;GO PRINT ATTR. MSG. FIVE TIMES
1031	003124	004537	005340	JSR	R5,AMSG	;SELECT ANY ENABLED CHAR. ATTRIBUTES
1032	003130	011512		SO		

CHARACTER ATTRIBUTE TEST (SI-SO SELECT)

SEQ 0023

```

1033 003132 004737 005174      JSR    PC,BLKPR1      ;GO PRINT ATTR. MSG. FIVE TIMES
1034 003136 004537 005340      JSR    R5,AMSG        ;DESELECT ANY CHAR. ATTRIBUTES
1035 003142 011515              SI
1036 003144 004737 005214      JSR    PC,DELAY        ;AND GO CHECK DELAY
1037
1038                               ;*****
1039                               ;*TEST 7          G      SCROLL SPEED JUMP TEST
1040                               ;*****
1041 003150 000004      TST7:  SCOPE
1042
1043 003152 004537 005340      JSR    R5,AMSG        ;ERASE AND INIT SCREEN WITH FORM-FEED
1044 003156 011205      FRMFED
1045 003160 004737 005266      JSR    PC,ADELAY      ;WAIT FOR FORM-FEED TO FINISH
1046 003164 004537 005340      JSR    R5,AMSG        ;DISPLAY HEADER
1047 003170 006746      M97
1048 003172 012737 000101 001262 1$: MOV    #101,STCHAR    ;SETUP STARTING CHARACTER
1049 003200 013701 001262      MOV    STCHAR,R1      ;LOAD R1 WITH CHAR. CODE
1050 003204 004737 004754      JSR    PC,FILBUF      ;LOAD LINE BUFFER WITH THAT CHAR.
1051 003210 004737 005056      JSR    PC,XPRNT      ;DISPLAY A FULL LINE FROM THE BUFFER
1052 003214 005237 001262      INC     STCHAR        ;PUT NEXT CHAR. CODE IN R1
1053 003220 023727 001262 000132  CMP    STCHAR,#132    ;SEE IF HAVE DONE 25 LINES YET
1054 003226 001364      BNE     1$
1055 003230 004537 005340      JSR    R5,AMSG        ;PUT SCROLL COUNT ON LAST LINE
1056 003234 011210      SCRL
1057
1058 003236 012737 000003 001266  MOV    #3,TEMPO      ;SET-UP LOOP COUNTER
1059 003244 112737 000060 011236  MOVB   #60,SCRNO     ;INIT SCROLL COUNT MESSAGE
1060 003252 012701 000031      MOV    #25.,R1        ;SET-UP VT COUNTER
1061 003256 012700 000013      MOV    #13,R0         ;PUT VERT. TAB CHAR. CODE IN R0
1062 003262 110077 175772      MOVB   R0,@VT0B      ;LOAD CHAR. IN OUTPUT BUFFER
1063 003266 105777 175764      4$:  TSTB   @VTOS        ;WAIT UNTIL PRINTED
1064 003272 100375      BPL     4$
1065 003274 005301      DEC     R1                 ;DECREMENT VT COUNTER
1066 003276 001371      BNE     3$                 ;LOOP UNTIL 25. VT'S PRINTED
1067 003300 105237 011236      INCB   SCRNO         ;INCREMENT SCROLL COUNT MSG.
1068 003304 004537 005340      JSR    R5,AMSG        ;UPDATE SCROLL COUNT MSG ON LAST LINE
1069 003310 011210      SCRL
1070 003312 004737 005214      JSR    PC,DELAY      ;THEN DELAY BETWEEN SETS OF VT'S
1071 003316 005337 001266      DEC     TEMPO
1072 003322 001353      BNE     2$                 ;LOOP UNTIL 3 SETS HAVE BEEN DONE
1073
1074                               ;*****
1075                               ;*TEST 10          H      COLUMN ADDRESS TEST
1076                               ;*****
1077 003324 000004      TST10: SCOPE
1078 003326 004537 005340      JSR    R5,AMSG        ;DISPLAY HEADING
1079 003332 007006      M99
1080 003334 012737 000057 001262  MOV    #57,STCHAR    ;SET-UP STARTING CHARACTER
1081 003342 013737 001242 001266  MOV    V#0,TEMPO    ;LOAD TEMP
1082 003350 005037 001162      CLR     $REGO        ;CLEAR LINE COUNTER
1083 003354 013701 001262      1$:  MOV    STCHAR,R1      ;LOAD R1=TO CHARACTER
1084 003360 004537 005010      JSR    R5,LIC         ;LOAD A BUFFER (TING WITH
1085 003364 001270      WIDTH      ; THAT CHARACTER AND WIDTH <BYTE>
1086 003366 004737 005056      JSR    PC,XPRNT      ;DISPLAY A FULL LINE FROM THE EUFFER
1087 003372 005337 001266      DEC     TEMPO        ;DONE ?
1088 003376 001005      BNE     2$                 ;BR IF YES

```

1089	003400	004737	005214		JSR	PC,DELAY	
1090	003404	013737	001242	001266	MOV	VH0,TEMPO	
1091	003412	005237	001162		2\$: INC	\$REGO	;INCREMENT THE LINE COUNTER
1092	003416	023737	001242	001162	CMP	VH0,\$REGO	;TEST FOR FULL SCREEN
1093	003424	001353			BNE	1\$	;BRANCH IF NOT COMPLETED
1094	003426	004737	005214		JSR	PC,DELAY	;TEST DELAY SWITCH
1095							
1096							
1097							
1098							
1099	003432	000004			1ST11: SCOPE		
1100	003434	032777	002000	175476	QKBTST: BIT	#BIT10,@SWR	;SEE IF BIT10 OF SWR SET
1101	003442	001402			BEQ	1\$	;IF IT ISN'T EXECUTE KEYBD. CHAR. TEST
1102	003444	000137	003624		JMP	\$EOP	;IF IT IS GO TO \$EOP
1103	003450	012703	012146		1\$: MOV	#V52RW,R3	;SET UP FOR LOWER CASE CHAR.
1104	003454	012706	001100		2\$: MOV	#STACK,SP	
1105	003460	004537	005340		A: JSR	R5,AMSG	;DISPLAY HEADER
1106	003464	007206			MKB		
1107	003466	012302			MOV	(R3)+,R2	;LOAD ROW #
1108	003470	004537	005340		JSR	R5,AMSG	;ISSUE ROW1 MESSAGE.
1109	003474	007375			MKBB2		
1110	003476	004737	005456		2\$: JSR	PC,TSTROW	;CHECK THE ROW
1111							
1112	003502	012302			B: MOV	(R3)+,R2	;LOAD ROW #
1113	003504	004537	005340		JSR	R5,AMSG	;2ND ROW
1114	003510	007463			MKBC		
1115	003512	004737	005456		JSR	PC,TSTROW	;CHECK 2ND ROW
1116							
1117	003516	012302			C: MOV	(R3)+,R2	;LOAD ROW #
1118	003520	004537	005340		JSR	R5,AMSG	;ISSUE ROW 2 MESSAGE.
1119	003524	007524			MKBD2		
1120	003526	004737	005456		2\$: JSR	PC,TSTROW	;CHECK ROW 3
1121							
1122	003532	012302			MOV	(R3)+,R2	;LOAD ROW #
1123	003534	004537	005340		JSR	R5,AMSG	
1124	003540	007614			MKBE		
1125	003542	004737	005456		JSR	PC,TSTROW	;CHECK ROW 4
1126							
1127	003546	012702	012164		MOV	#ROW5,R2	
1128	003552	004537	005340		JSR	R5,AMSG	
1129	003556	007705			MKBF		
1130	003560	004737	005456		JSR	PC,TSTROW	;CHECK ROW 5
1131							
1132							
1133							
1134	003564	004537	005340		QKBDON: JSR	R5,AMSG	;END OF KEYBOARD TEST
1135	003570	010235			MKBR		
1136	003572	000137	003624		JMP	\$EOP	;GO TO END OF PASS
1137	003576	013053			DISPCH: MTEXT0		
1138	003600	013157			MTEXT1		
1139	003602	013260			MTEXT2		
1140	003604	013362			MTEXT3		
1141	003606	013445			MTEXT4		
1142	003610	013547			MTEXT5		
1143	003612	013650			MTEXT6		
1144	003614	013702			MTEXT7		

;COMPLETION OF KEYBOARD-KEYPAD TEST

```

1145 003616 014002 MTEXT8
1146 003620 000000 0
1147 003622 000000 0
1148
1149
1150 .SBTTL END OF PASS ROUTINE
1151
1152 ;*****
1153 ;*INCREMENT THE PASS NUMBER ($PASS)
1154 ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
1155 ;*TYPE 'END PASS #XXXXX' (WHERE XXXXX IS A DECIMAL NUMBER)
1156 ;*IF THERES A MONITOR GO TO IT
1157 ;*IF THERE ISN'T JUMP TO NOWEOP
1158
1159 $EOP:
1160 003624 000004 SCOPE
1161 003626 005737 001224 TST LAST ;TEST IF MORE
1162 003632 001414 BEQ 1$ ;BR IF NONE
1163 003634 023737 001224 001226 CMP LAST,VTNOW ;IS THIS THE LAST ONE
1164 003642 001410 BEQ 1$ ;BR IF YES
1165 003644 062737 000010 001226 ADD #10,VTNOW
1166 003652 012737 002112 002110 MOV #TST1,WHERE
1167 003660 000137 001756 JMP RSTRT ;TEST NEXT ONE
1168 003664 005737 001100 1$: TST $PASS ;TEST IF FIRST PASS
1169 003670 001002 BNE 2$ ;BR IF NOT
1170 003672 104401 011642 TYPE ,PASHED ;TYPE EOP HEADER
1171 003676 000005 2$: RESET
1172 003700 004737 005266 JSR PC,ADelay ;EXTRA DELAY
1173
1174 003704 005037 001102 CLR $TSTNM ;;ZERO THE TEST NUMBER
1175 003710 005237 001100 INC $PASS ;;INCREMENT THE PASS NUMBER
1176 003714 042737 100000 001100 BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
1177 003722 005327 DEC (PC)+ ;;LOOP?
1178 003724 000001 $EOPCT: .WORD 1
1179 003726 003022 BGT $DOAGN ;;YES
1180 003730 012737 MOV (PC)+,@(PC)+ ;;RESTORE COUNTER
1181 003732 000001 $ENDCT: .WORD 1
1182 003734 003724 $EOPCT
1183 003736 104401 004003 TYPE ,SENDMG ;;TYPE 'END PASS #'
1184 003742 013746 001100 MOV $PASS,-(SP) ;;SAVE $PASS FOR TYPEOUT
1185 003746 104405 TYPDS ;;GO TYPE--DECIMAL ASCII WITH SIGN
1186 003750 104401 004000 TYPE ,SENULL ;;TYPE A NULL CHARACTER
1187 003754 013700 000042 $GET42: MOV @#42,R0 ;;GET MONITOR ADDRESS
1188 003760 001405 BEQ $DOAGN ;;BRANCH IF NO MONITOR
1189 003762 000005 RESET ;;CLEAR THE WORLD
1190 003764 004710 $ENDAD: JSR PC,(R0) ;;GO TO MONITOR
1191 003766 000240 NOP ;;SAVE ROOM
1192 003770 000240 NOP ;;FOR
1193 003772 000240 NOP ;;ACT11
1194 003774 $DOAGN:
1195 003774 000137 JMP @(PC)+ ;;RETURN
1196 003776 004020 $RTNAD: .WORD NOWEOP
1197 004000 377 377 000 $ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
1198 004003 015 042412 042116 $ENDMG: .ASCIIZ <15><12>/END PASS #/
1199 004010 050040 051501 020123
1200 004016 000043

```

1201	004020	005737	001250	NOWEOP:	TST	PRTCNT	
1202	004024	100002			BPL	11\$	
1203	004026	105337	001250		DECB	PRTCNT	
1204	004032	012737	002112	002110	MOV	#TST1,WHERE	
1205	004040	104401	004000		TYPE	\$ENULL	
1206	004044	000137	001750		JMP	SBEGIN	
1207					;*****		
1208					*TEST 12	J	FULL KEYBOARD CHARACTER TEST
1209					;*****		
1210	004050	000004			TST12:	SCOPE	
1211	004052	012703	012146		KRBTST:	MOV	#V52RW,R3
1212	004056	012706	001100		2\$:	MOV	#STACK,SP
1213	004062	004537	005340		A1:	JSR	R5,AMSG
1214	004066	011205				FRMFED	
1215	004070	004737	005266			JSR	PC,ADELAY
1216	004074	004537	005340			JSR	R5,AMSG
1217	004100	010350				MKBX	
1218	004102	C12302				MOV	(R3)+,R2
1219	004104	004537	005340			JSR	R5,AMSG
1220	004110	007375				MKBB2	
1221	004112	004737	005456		2\$:	JSR	PC,TSTROW
1222							
1223	004116	012302			B1:	MOV	(R3)+,R2
1224	004120	004537	005340			JSR	R5,AMSG
1225	004124	007463				MKBC	
1226	004126	004737	005456			JSR	PC,TSTROW
1227							
1228	004132	012302			C1:	MOV	(R3)+,R2
1229	004134	004537	005340			JSR	R5,AMSG
1230	004140	007524				MKBD2	
1231	004142	004737	005456		2\$:	JSR	PC,TSTROW
1232							
1233	004146	012302				MOV	(R3)+,R2
1234	004150	004537	005340			JSR	R5,AMSG
1235	004154	007614				MKBE	
1236	004156	004737	005456			JSR	PC,TSTROW
1237							
1238	004162	012702	012164			MOV	#ROW5,R2
1239	004166	004537	005340			JSR	R5,AMSG
1240	004172	007705				MKBF	
1241	004174	004737	005456			JSR	PC,TSTROW
1242							
1243						;TEST THE 'LEFT'-SHIFT KEY	
1244							
1245	004200	004537	005340		D:	JSR	R5,AMSG
1246	004204	007762				MKBG	
1247	004206	012302				MOV	(R3)+,R2
1248	004210	004537	005340			JSR	R5,AMSG
1249	004214	007375				MKBB2	
1250	004216	004737	005456		2\$:	JSR	PC,TSTROW
1251	004222	004537	005340			JSR	R5,AMSG
1252	004226	007253				MKB1	
1253							
1254						;TEST THE 'RIGHT-SHIFT' KEY	
1255							
1256	004230	004537	005340		E:	JSR	R5,AMSG

;SET UP FOR LOWER CASE CHAR.

;ERASE AND INIT SCREEN WITH FORM-FEED

;WAIT FOR FORM-FEED TO FINISH

;DISPLAY HEADER

;LOAD ROW #

;ISSUE ROW1 MESSAGE.

;CHECK THE ROW

;LOAD ROW #

;2ND ROW

;CHECK 2ND ROW

;LOAD ROW #

;ISSUE ROW 3 MESSAGE.

;CHECK ROW 3

;LOAD ROW #

;CHECK ROW 4

;CHECK ROW 4

;CHECK ROW 5

;CHECK ROW 5

;CHECK ROW 5

;CHECK ROW 5

;DEPRESS THE 'LEFT-SHIFT' KEY

;LOAD ROW 1 SHIFTED TABLE

;ISSUE ROW1 MESSAGE.

;TEST THE ROW

;RELEASE THE SHIFT KEY

;TEST THE ROW

;SET THE 'RIGHT-SHIFT' KEY

```

1257 004234 010031 MKBGA
1258 004236 012302 MOV (R3)+,R2 ;LOAD TABLE POINTER
1259 004240 004537 005340 JSR R5,AMSG ;ISSUE ROW1 MESSAGE.
1260 004244 007375 MKBB2
1261 004246 004737 005456 2$: JSR PC,TSTROW ;TEST THE ROW AGAIN WITH THE RIGHT-SHIFT SET
1262 004252 004537 005340 JSR R5,AMSG
1263 004256 007253 MKB1 ;RELEASE SK'FT KEY
1264 ;TEST THE CONTROL MODE
1265
1266 004260 004537 005340 F: JSR R5,AMSG ;SET CTRL
1267 004264 010101 MKBH
1268 004266 012302 MOV (R3)+,R2 ;LOAD ROW 1 CTRL TABLE
1269 004270 004537 005340 JSR R5,AMSG ;ISSUE ROW1 MESSAGE.
1270 004274 007375 MKBB2
1271 004276 004737 005456 2$: JSR PC,TSTROW ;TEST THE ROW
1272
1273 ;COMPLETION OF KEYBOARD TEST
1274
1275 004302 004537 005340 KRBON: JSR R5,AMSG ;END OF KEYBOARD TEST
1276 004306 010235 MKBR
1277 004310 000137 004052 JMP KRBST ;LOOP ON TEST
1278
1279
1280
1281 ;*****
1282 ;*TEST 13 K KEYBOARD OCTAL VALUE LOOP
1283 ;*****
1284 004314 000004 TST13: SCOPE
1285 004316 012706 001100 KRBECO: MOV #STACK,SP
1286 004322 004537 005340 JSR R5,AMSG ;DISPLAY HEADER
1287 004326 010460 MKE
1288 004330 004737 006334 1$: JSR PC,GETCHR ;GET CHAR
1289 004334 000775 BR 1$ ;:BR BACK IF NO INPUT
1290 004336 004737 005360 JSR PC,OCTAL ;CONVERT RO TO OCTAL
1291 004342 113737 005450 010703 MOVB DIG0,MKEB ;LOAD DIGIT
1292 004350 113737 005452 010704 MOVB DIG1,MKEB+1 ;LOAD DIGIT
1293 004356 113737 005454 010705 MOVB DIG2,MKEB+2 ;LOAD DIGIT
1294 004364 042700 177600 BIC #177600,R0
1295 004370 001001 BNE 10$
1296 004372 005200 INC R0
1297 004374 012701 004504 10$: MOV #BFCHR,R1 ;LOAD POINTER
1298 004400 121100 5$: CMPB (R1),R0 ;TEST IF = TO VALUE IN TABLE ?
1299 004402 001403 BEQ 3$ ;BR IF FOUND
1300 004404 005721 TST (R1)+ ;MOVE POINTER
1301 004406 001374 BNE 5$ ;BR IF MORE
1302 004410 000407 BR 2$ ;BR IF NOT IN LIST
1303 004412 062701 000036 3$: ADD #BFCHAR-BFCHR,R1 ;UPDATE POINTER
1304 004416 112137 010676 MOVB (R1)+,MKEA1 ;LOAD 1ST CHAR
1305 004422 112137 010677 MOVB (R1)+,MKEA1+1 ;LOAD 2ND
1306 004426 000420 BR 4$
1307 004430 120027 000040 2$: CMPB R0,#40 ;TEST IF LESS THAN 40
1308 004436 062700 000100 BHI 6$ ;BR IF ABOVE
1309 004442 110037 010677 ADD #100,R0 ;MAKE PRINTABLE
1310 004446 112737 000136 010676 MOVB R0,MKEA1+1 ;SAVE CHAR
1311 004454 000405 BR 4$ ;ADD A '^' BEFORE CHARACTER
1312 004456 112737 000040 010677 6$: MOVB #40,MKEA1+1 ;LOAD SPACE

```

```

1313 004464 110037 010676      MOVB  R0,MKEA1      ;LOAD CHARACTER
1314 004470 005237 005172      4$: INC  IGNORE      ;IGNORE DOUBLE CHARACTER FLAG
1315 004474 004537 005340      JSR   R5,AMSG      ;DISPLAY MESSAGE
1316 004500 010665                MKEA
1317 004502 000712                BR    1$          ;LOOP BACK
1318
1319
1320

```

;TABLE OF DEFINED CHARACTERS

```

1321 004504 000010      BFCHR: 10      ;BACKSPACE (CURSOR LEFT CODE)
1322 004506 000011      11      ;TAB CODE
1323 004510 000012      12      ;LINE FEED CODE
1324 004512 000013      13      ;VERTICAL TAB
1325 004514 000014      14      ;FORM FEED
1326 004516 000015      15      ;CARRIAGE RETURN CODE
1327 004520 000016      16      ;SO (^N)
1328 004522 000017      17      ;SI (^O)
1329 004524 000033      33      ;ESCAPE CODE
1330 004526 000040      40      ;SPACE CODE
1331 004530 000177      177     ;DELETE CODE
1332 004532 000000      0
1333 004534 000000      0
1334 004536 000000      0
1335 004540 000000      0
1336
1337
1338

```

;DEFINED CHARACTER EQUIVALENTS

```

1339 004542 051502      BFCHAR: .ASCII /BS/      ;BACKSPACE (CURSOR LEFT)
1340 004544 052110      .ASCII /HT/      ;H TAB
1341 004546 043114      .ASCII /LF/      ;LINE FEED
1342 004550 052126      .ASCII /VT/      ;VERTICAL TAB
1343 004552 043106      .ASCII /FF/      ;FORM-FEED
1344 004554 051103      .ASCII /CR/      ;CARRIAGE RETURN
1345 004556 047523      .ASCII /SO/      ;SO - ^N
1346 004560 044523      .ASCII /SI/      ;SI - ^O
1347 004562 051505      .ASCII /ES/      ;ESCAPE
1348 004564 050123      .ASCII /SP/      ;SPACE
1349 004566 042504      .ASCII /DE/      ;DELETE
1350 004570 000000 000000 000000      0,0,0,0,0
1351 004576 000000 000000 000000
1352
1353
1354
1355

```

.EVEN

```

;*****
;*TEST 14      L      KEYBOARD ECHO LOOP
;*****

```

```

1356 004604 000004      TST14: SCOPE
1357 004606 012706 001100      KRBECH: MOV  #STACK,SP
1358 004612 004537 005340      JSR   R5,AMSG      ;DISPLAY HEADER
1359 004616 010712      MKEH
1360
1361 004620 004737 006334      1$: JSR   PC,GETCHR      ;GET CHARACTER
1362 004624 000775      BR    1$
1363 004626 110077 174426      MOVB  R0,@VTOB      ;LOAD THE CHARACTER
1364 004632 105777 174420      2$: TSTB  @VTOS      ;WAIT FOR DONE
1365 004636 100375      BPL    2$
1366 004640 000767      BR    1$          ;LOOP BACK
1367
1368

```

```

1369
1370
1371
1372 004642 000004
1373
1374 004644 004537 005340
1375 004650 007124
1376 004652 005002
1377 004654 005037 001162
1378 004660 012700 011242
1379 004664 112001
1380 004666 120127 000377
1381 004672 001413
1382 004674 032702 000001
1383 004700 001002
1384 004702 052701 000200
1385 004706 110177 174346
1386 004712 105777 174340
1387 004716 100375
1388 004720 000761
1389
1390 004722 005237 001162
1391 004726 022737 000005 001162
1392 004734 001351
1393 004736 005202
1394 004740 022702 000005
1395 004744 001343
1396 004746 004737 005214
1397 004752 000734
1398
1399
1400
1401
1402
1403
1404
1405 004754 013702 001270
1406 004760 012700 022066
1407 004764 112720 000015
1408 004770 112720 000012
1409 004774 110120
1410 004776 005302
1411 005000 001375
1412 005002 112710 000377
1413 005006 000207
1414
1415
1416
1417
1418 005010 012700 022066
1419 005014 013502
1420 005016 112720 000015
1421 005022 112720 000012
1422 005026 110120
1423 005030 005201
1424 005032 023701 001264

;*****
;TEST 15 M CHARACTER ATTRIBUTE TEST (M.S.B. SELECT)
;*****
TST15: SCOPE

CHRATT: JSR R5,AMSG ;DISPLAY HEADER
M9222
CLR R2 ;CLEAR BLOCK COUNTER
1$: CLR $REGO ;CLEAR LINE COUNTER
6$: MOV #ATTR8,R0 ;SETUP MESSAGE POINTER
2$: MOVB (R0)+,R1 ;GET A CHARACTER
CMPB R1,#377 ;SEE IF REACHED END OF LINE YET
BEQ 5$
BIT #BIT0,R2 ;SEE IF ON AN EVEN OR ODD BLOCK
BNE 3$ ;IF ON EVEN BLOCK SET CHAR. ATTR.
BIS #BIT7,R1 ;SET M.S.B. OF THE CHAR. CODE
3$: MOVB R1,@VTOB ;LOAD THE CHAR.
4$: TSTB @VTOS ;WAIT UNTIL PRINTED
BPL 4$
BR 2$ ;GO GET NEXT CHAR.

5$: INC $REGO ;INCREMENT LINE COUNTER
CMP #5,$REGO ;SEE IF 5 LINES OF BLOCK PRINTED YET?
BNE 6$ ;CONTINUE IF BLOCK NOT DONE
INC R2 ;ELSE INCREMENT BLOCK COUNTER
CMP #5,R2 ;SEE IF SCREEN FULL (5 BLOCKS)?
BNE 1$ ;IF NO CONTINUE
JSR PC,DELAY ;WAIT A WHILE BEFORE LOOPING ON TEST
BR CHRATT

;***** PROGRAM SUBROUTINES *****
;LOAD A SINGLE CHARACTER ACROSS THE SCREEN WIDTH
;
FILBUF: MOV WIDTH,R2 ;LOAD WIDTH VALUE
FILBFB: MOV #BUFFER,R0 ;SET-UP BUFFER POINTER
MOVB #15,(R0)+ ;LOAD 'CR'
MOVB #12,(R0)+ ;LOAD 'FL'
FILBFA: MOVB R1,(R0)+ ;SAVE THE CHARACTER IN THE BUFFER
DEC R2 ;FINISHED?
BNE FILBFA ;BRANCH IF NOT COMPLETED
MOVB #377,(R0) ;LOAD TERM.
RTS PC ;EXIT

;LOAD A INCREMENTING CHARACTER ACROSS THE SCREEN WIDTH
;ONLY 40 THRU 177 ARE LEGAL CHARACTERS
LIC: MOV #BUFFER,R0 ;SET-UP BUFFER POINTER
MOV @5+,R2 ;SET-UP WIDTH
MOVB #15,(R0)+ ;LOAD 'CR'
MOVB #12,(R0)+ ;LOAD 'LF'
LICA: MOVB R1,(R0)+ ;SAVE A CHARACTER IN THE BUFFER
INC R1 ;UPDATE THE CHARACTER
CMP LASTCH,R1 ;TEST FOR

```

1425	005036	001002		BNE	LICB	;BRANCH IF NOT
1426	005040	012701	000040	MOV	#40,R1	;MAKE A LEGAL CHARACTER
1427	005044	005302		LICB: DEC	R2	;DECREMENT COUNT
1428	005046	001367		BNE	LICA	;BRANCH IF NOT COMPLETED
1429	005050	112710	000377	MOVB	#377,(R0)	;LOAD TERM
1430	005054	000205		RTS	R5	;EXIT

;DISPLAY SUBROUTINE

1434	005056	012700	022066	XPRNT: MOV	#BUFFER,R0	;SETUP BUFFER POINTER
1435	005062	105777	174170	XPRNTA: TSTB	@VTOS	;TEST READY
1436	005066	100404		BMI	XPRNTB	;BRANCH IF SET
1437	005070	005777	174162	TST	@VTOS	;TEST ERROR
1438	005074	100372		BPL	XPRNTA	;BRANCH IF RESET
1439	005076	104001		ERROR	1	;ERROR FLAG SET ON TRANSMITTER STATUS
1440	005100	112001		XPRNTB: MOVB	(R0)+,R1	
1441	005102	122701	000377	CMPB	#377,R1	
1442	005106	001413		BEQ	1\$	;BR IF END OF MSG.
1443	005110	122701	000033	5\$: CMPB	#33,R1	;TEST FOR ESC
1444	005114	001003		BNE	3\$	;BR IF NOT
1445	005116	005237	005170	INC	ANESC	;SET SOFT FLAG
1446	005122	000402		BR	4\$	
1447	005124	005037	005170	3\$: CLR	ANESC	;CLEAR SOFT FLAG
1448	005130	110177	174124	4\$: MOVB	R1,@VTOS	;LOAD CHAR
1449	005134	000752		BR	XPRNTA	;GO GET NEXT CHAR.

;NORMAL EXIT

1453	005136	005037	005160	1\$: CLR	XOFFOK	
1454	005142	005037	005172	CLR	IGNORE	
1455	005146	005037	005156	CLR	XOFFBR	
1456	005152	000207		RTS	PC	;EXIT

1458	005154	000000		LOOP: 0		
1459	005156	000000		XOFFBR: 0		
1460	005160	000000		XOFFOK: 0		
1461	005162	000000		AXOFF: 0		
1462	005164	000000		XOFFRC: 0		
1463	005166	000000		XONRC: 0		
1464	005170	000000		ANESC: 0		
1465	005172	000C00		IGNORE: 0		;WHEN SET IGNORE KEYBOARD FLAGS

;SUBROUTINE TO PRINT A BLOCK OF 5 LINES EACH (FOR CHAR. ATTR. TEST F - SO/SI)

1470	005174	012702	000005	BLKPRT: MOV	#5,R2	;SET UP LINE COUNTER
1471	005200	004537	005340	1\$: JSR	R5,AMSG	;TYPE CHAR. ATTR. MSG (SO/SI)
1472	005204	011366		ATTRS		
1473	005206	005302		DEC	R2	;DECREMENT LINE COUNTER
1474	005210	001373		BNE	1\$	;LOOP UNTIL MSG PRINTED 5 TIMES
1475	005212	000207		RTS	PC	;THEN RETURN

;PROGRAM DELAY ROUTINE

1480	005214	013737	001234	005262	DELAY: MOV	SUBTST,10\$	;LOAD COUNT
------	--------	--------	--------	--------	------------	-------------	-------------

```

1481 005222 005037 005264      CLR      11$
1482 005226 032777 010000 173704 2$:    BIT      #BIT12,ASWR      ;TEST SR
1483 005234 001006          BNE      3$      ;BR IF SET
1484 005236 005337 005264      DEC      11$      ;DELAY
1485 005242 001371          BNE      2$
1486 005244 005337 005262      DEC      10$
1487 005250 100366          BPL      2$      ;DELAY
1488 005252 000240          3$:    NOP
1489 005254 000240          NOP
1490 005256 000240          NOP
1491 005260 000207          RTS      PC      ;EXIT
1492
1493 005262 000002          10$:    2
1494 005264 000000          11$:    0
1495
1496 005266 013737 001234 005334 ADELAY: MOV      SUBTST,10$
1497 005274 005037 005336      CLR      11$
1498 005300 006237 005334      ASR      10$
1499 005304 006237 005334      ASR      10$
1500 005310 005337 005336          2$:    DEC      11$
1501 005314 001375          BNE      2$
1502 005316 005337 005334      DEC      10$
1503 005322 100372          BPL      2$
1504 005324 000240          NOP
1505 005326 000240          NOP
1506 005330 000240          NOP
1507 005332 000207          RTS      PC
1508 005334 000000          10$:    0
1509 005336 000000          11$:    0
1510
1511
1512      ;HEADER SUBROUTINE FOR M7142
1513
1514 005340 012537 005350      AMSG:    MOV      (R5)+,10$      ;GET POINTER
1515 005344 004537 006412      1$:    JSR      R5,MT0B      ;MOVE TO BUFFER
1516 005350 000000          10$:    0
1517 005352 004737 005056      11$:    JSR      PC,XPRNT      ;DISPLAY IT
1518 005356 000205          RTS      R5      ;EXIT
1519
1520
1521      ;OCTAL - 3 BIT CONVERSION
1522
1523 005360 010001          OCTAL:    MOV      R0,R1      ;LOAD R1
1524 005362 042701 177770      BIC      #177770,R1      ;MASK
1525 005366 062701 000060      ADD      #60,R1
1526 005372 110137 005454      MOVB     R1,DIG2      ;SAVE LSD
1527 005376 010001          MOV      R0,R1
1528 005400 006001          ROR      R1
1529 005402 006001          ROR      R1
1530 005404 006001          ROR      R1
1531 005406 042701 177770      BIC      #177770,R1
1532 005412 062701 000060      ADD      #60,R1
1533 005416 110137 005452      MOVB     R1,DIG1      ;SAVE IT
1534 005422 010001          MOV      R0,R1
1535 005424 006101          ROL      R1
1536 005426 006101          ROL      R1
  
```

```

1537 005430 000301          SWAB    R1
1538 005432 042701 177770    BIC     #177770,R1
1539 005436 062701 000060    ADD     #60,R1
1540 005442 110137 005450    MOVB    R1,DIG0      ;SAVE MSD
1541 005446 000207          RTS     PC           ;EXIT
1542
1543 005450 000000          DIG0:    0
1544 005452 000000          DIG1:    0
1545 005454 000000          DIG2:    0
1546
1547          ;SUBROUTINE FOR THE KEYBOARD CHARACTER TEST
1548 005456 004537 005340    TSTROW: JSR     R5,AMSG      ;DISPLAY HEADER
1549 005462 007310          MKBA
1550 005464 004737 006334    1$:    CALL    GETCHR      ;GET A CHARACTER
1551 005470 000775          BR        1$              ;WAIT FOR INPUT
1552 005472 004737 005716    CALL    PARITY      ;ADJUST CHAR IF PARITY
1553 005476 120027 000040    CMPB     RO,#40      ;SEE IF IT IS A PRINTING CHAR.
1554 005502 002410          BLT       12$          ;ECHO IF IT IS (40-176)
1555 005504 120027 000176    CMPB     RO,#176
1556 005510 003005          BGT       12$
1557 005512 110077 173542    MOVB     RO,@VTOB      ;ECHO THE CHAR.
1558 005516 105777 173534    11$:   TSTB     @VTOS      ;WAIT UNTIL ECHO'D
1559 005522 100375          BPL       11$
1560 005524 120037 005706    12$:   CMPB     RO,100$    ;CMP EXPCT AND RCVD
1561 005530 001003          BNE       2$              ;TYPE ERROR INFO
1562 005532 005722          TST       (R2)+
1563 005534 100353          BPL       1$              ;LOOP TILL DONE
1564 005536 000207          RETURN    ;EXIT TEST
1565
1566          ;COME HERE IF EXPECTED NOT EQUAL TO RECVD
1567          ;CONVERT RESULTS TO OCTAL FOR TYPEOUT
1568
1569 005540 010037 001126    2$:    MOV      RO,$BDDAT      ;LOAD BAD CHARACTER
1570 005544 004737 005360    JSR       PC,OCTAL      ;CONVERT TO OCTAL
1571 005550 113737 005450 010226    MOVB     DIG0,MKBQB      ;LOAD OCTAL #
1572 005556 113737 005452 010227    MOVB     DIG1,MKBQB+1
1573 005564 113737 005454 010230    MOVB     DIG2,MKBQB+2
1574 005572 042700 177600    BIC       #177600,RO
1575 005576 120027 000040    CMPB     RO,#40      ;TEST IF PRINTABLE
1576 005602 101002          BHI       10$          ;BR IF PRINTABLE
1577 005604 112700 000056    MOVB     #56,RO      ;CONVERT TO A '*' CHARACTER
1578 005610 110037 010222    10$:   MOVB     RO,MKBQ2      ;SAVE CHAR
1579 005614 011200          MOV       (R2),RO      ;GET GOOD CHAR
1580 005616 053700 005714    BIS       MASK2,RO
1581 005622 010037 001124    MOV       RO,$GDDAT      ;LOAD GOOD CHARACTER
1582 005626 004737 005360    JSR       PC,OCTAL      ;CONVERT IT
1583 005632 113737 005450 010207    MOVB     DIG0,MKBQA      ;LOAD DIGIT
1584 005640 113737 005452 010210    MOVB     DIG1,MKBQA+1
1585 005646 113737 005454 010211    MOVB     DIG2,MKBQA+2
1586 005654 042700 177600    BIC       #177600,RO
1587 005660 110037 010203    MOVB     RO,MKBQ1      ;SAVE CHAR
1588 005664 023737 001252 001144    CMP      VTIS,$TKS      ;TEST IF ON CTY
1589 005672 001403          BEQ       3$              ;BR IF YES
1590 005674 004537 005340    JSR       R5,AMSG      ;DISPLAY ERROR MESSAGE
1591 005700 010142          MKBQ
1592 005702 104003    3$:    ERROR    3              ;CHARACTER RECVD NOT EQUAL TO EXPECTED

```

```

1593 005704 000667          BR      1$          ;BR BACK AND TEST THE CHARACTER AGAIN
1594
1595 005706 000000          100$: 0
1596 005710 000000          101$: 0
1597 005712 177600          MASK1: 177600
1598 005714 000000          MASK2: 0
1599
1600          ;TEST TO ADD PARITY TO EXPCTD CHAR
1601 005716 012737 177600 005712 PARITY: MOV      #177600,MASK1
1602 005724 005037 005714          CLR      MASK2
1603 005730 032777 000004 173202          BIT      #BIT2,@SWR          ;TEST SWR
1604 005736 001416          BEQ      4$          ;DO NOT TEST PARITY BIT
1605 005740 042737 000200 005712          BIC      #BIT7,MASK1          ;ENABLE PARITY BIT
1606 005746 032777 000002 173164          BIT      #BIT1,@SWR          ;TEST IF FORCED PARITY
1607 005754 001417          BEQ      5$          ;BR IF NOT FORCED PARITY BIT
1608 005756 032777 000001 173154          BIT      #BIT0,@SWR          ;TEST FOR EVEN/ODD PARITY
1609 005764 001403          BEQ      4$          ;BR IF ALWAYS OFF
1610 005766 052737 000200 005714          BIS      #BIT7,MASK2          ;SET BIT 7
1611 005774 111237 005706          4$:  MOVB      (R2),100$          ;GET EXPECTED
1612 006000 053737 005714 005706          BIS      MASK2,100$          ;SET BIT 7 IF EXPECTED
1613 006006 043700 005712          BIC      MASK1,R0          ;MASK VALUE READ
1614 006012 000207          RETURN          ;EXIT ROUTINE
1615 006014 005037 005710          5$:  CLR      101$          ;CLEAR TEMP
1616 006020 111237 005706          MOVB      (R2),100$          ;CLEAR CHAR SAVE
1617 006024 006037 005706          20$:  ROR      100$          ;ROTATE CHAR
1618 006030 103002          BCC      21$          ;BR IF NO CARRY
1619 006032 005237 005710          INC      101$          ;UPDATE CNT
1620 006036 105737 005706          21$:  TSTB      100$          ;DONE ?
1621 006042 001370          BNE      20$          ;BR IF NOT
1622 006044 032777 000001 173066          BIT      #BIT0,@SWR          ;TEST EVEN/ODD
1623 006052 001407          BEQ      23$          ;BR IF OPER. SAYS EVEN
1624 006054 006037 005710          ROR      101$          ;
1625 006060 103403          BCS      22$          ;BR IF ODD ALREADY
1626 006062 052737 000200 005714          BIS      #BIT7,MASK2          ;SET PARITY BIT
1627 006070 000741          22$:  BR      4$          ;BR TO TEST CHAR
1628 006072 006037 005710          23$:  ROR      101$          ;
1629 006076 103003          BCC      24$          ;BR IF EVEN ALREADY
1630 006100 052737 000200 005714          BIS      #BIT7,MASK2
1631 006106 000732          24$:  BR      4$          ;BR TO TEST CHAR
1632
1633
1634
1635
1636          .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
1637
1638          ;*****
1639          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
1640          ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
1641          ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
1642          ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
1643          ;*REPLACED WITH SPACES.
1644          ;*CALL:
1645          ;*      MOV      NUM,-(SP)          ;;PUT THE BINARY NUMBER ON THE STACK
1646          ;*      TYPDS          ;;GO TO THE ROUTINE
1647
1648 006110          $TYPDS:

```

1649	006110	010046		MOV	R0,-(SP)	::PUSH R0 ON STACK
1650	006112	010146		MOV	R1,-(SP)	::PUSH R1 ON STACK
1651	006114	010246		MOV	R2,-(SP)	::PUSH R2 ON STACK
1652	006116	010346		MOV	R3,-(SP)	::PUSH R3 ON STACK
1653	006120	010546		MOV	R5,-(SP)	::PUSH R5 ON STACK
1654	006122	012746	020200	MOV	#20200,-(SP)	::SET BLANK SWITCH AND SIGN
1655	006126	016605	000020	MOV	20(SP),R5	::GET THE INPUT NUMBER
1656	006132	100004		BPL	1\$	::BR IF INPUT IS POS.
1657	006134	005405		NEG	R5	::MAKE THE BINARY NUMBER POS.
1658	006136	112766	000055 000001	MOVB	#'-,1(SP)	::MAKE THE ASCII NUMBER NEG.
1659	006144	005000		CLR	R0	::ZERO THE CONSTANTS INDEX
1660	006146	012703	006324	MOV	#SDBLK,R3	::SETUP THE OUTPUT POINTER
1661	006152	112723	000040	MOVB	#',(R3)+	::SET THE FIRST CHARACTER TO A BLANK
1662	006156	005002		CLR	R2	::CLEAR THE BCD NUMBER
1663	006160	016001	006314	MOV	\$DTBL(R0),R1	::GET THE CONSTANT
1664	006164	160105		SUB	R1,R5	::FORM THIS BCD DIGIT
1665	006166	002402		BLT	4\$	::BR IF DONE
1666	006170	005202		INC	R2	::INCREASE THE BCD DIGIT BY 1
1667	006172	000774		BR	3\$	
1668	006174	060105		ADD	R1,R5	::ADD BACK THE CONSTANT
1669	006176	005702		TST	R2	::CHECK IF BCD DIGIT=0
1670	006200	001002		BNE	5\$	::FALL THROUGH IF 0
1671	006202	105716		TSTB	(SP)	::STILL DOING LEADING 0'S?
1672	006204	100407		BMI	7\$	::BR IF YES
1673	006206	106316		ASLB	(SP)	::MSD?
1674	006210	103003		BCC	6\$	::BR IF NO
1675	006212	116663	000001 177777	MOVB	1(SP),-1(R3)	::YES--SET THE SIGN
1676	006220	052702	000060	BIS	#'0,R2	::MAKE THE BCD DIGIT ASCII
1677	006224	052702	000040	BIS	#',R2	::MAKE IT A SPACE IF NOT ALREADY A DIGIT
1678	006230	110223		MOVB	R2,(R3)+	::PUT THIS CHARACTER IN THE OUTPUT BUFFER
1679	006232	005720		TST	(R0)+	::JUST INCREMENTING
1680	006234	020027	000010	CMP	R0,#10	::CHECK THE TABLE INDEX
1681	006240	002746		BLT	2\$	::GO DO THE NEXT DIGIT
1682	006242	003002		BGT	8\$	::GO TO EXIT
1683	006244	010502		MOV	R5,R2	::GET THE LSD
1684	006246	000764		BR	6\$	::GO CHANGE TO ASCII
1685	006250	105726		TSTB	(SP)+	::WAS THE LSD THE FIRST NON-ZERO?
1686	006252	100003		BPL	9\$	::BR IF NO
1687	006254	116663	177777 177776	MOVB	-1(SP),-2(R3)	::YES--SET THE SIGN FOR TYPING
1688	006262	105013		CLRB	(R3)	::SET THE TERMINATOR
1689	006264	012605		MOV	(SP)+,R5	::POP STACK INTO R5
1690	006266	012603		MOV	(SP)+,R3	::POP STACK INTO R3
1691	006270	012602		MOV	(SP)+,R2	::POP STACK INTO R2
1692	006272	012601		MOV	(SP)+,R1	::POP STACK INTO R1
1693	006274	012600		MOV	(SP)+,R0	::POP STACK INTO R0
1694	006276	104401	006324	TYPE	\$DBLK	::NOW TYPE THE NUMBER
1695	006302	016666	000002 000004	MOV	2(SP),4(SP)	::ADJUST THE STACK
1696	006310	012616		MOV	(SP)+,(SP)	
1697	006312	000002		RTI		::RETURN TO USER
1698	006314	023420		\$DTBL:	10000.	
1699	006316	001750			1000.	
1700	006320	000144			100.	
1701	006322	000012			10.	
1702	006324	000004		\$DBLK:	.BLKW 4	
1703						
1704						

;SUBROUTINE TO GET A CHARACTER FROM THE KEYBOARD

```

1705
1706 006334 013737 001232 006406 GETCHR: MOV    TIME0,TIME1    ;LOAD TIME COUNTER
1707 006342 005037 006410          CLR    TIME2
1708
1709 006346 105777 172700          1$:   TSTB   @VTIS          ;TEST INPUT STATUS
1710 006352 100005          BPL    2$                ;BR IF CLEARED
1711 006354 017700 172674          MOV    @VTIB,R0        ;READ A CHAR
1712 006360 062716 000002          ADD    #2,(SP)        ;UPDATE RETURN
1713 006364 000207          RTS    PC                ;EXIT
1714
1715 006366 005337 006410          2$:   DEC    TIME2          ;DELAY
1716 006372 001365          BNE    1$
1717 006374 005337 006406          DEC    TIME1          ;FINISHED ?
1718 006400 100362          BPL    1$                ;LOOP TILL TIME EXPIRED
1719 006402 104002          ERROR  2                ;NO INPUT FLAG FROM DEVICE
1720 006404 000207          RTS    PC                ;EXIT
1721
1722 006406 000000          TIME1:  0
1723 006410 000000          TIME2:  0
1724
1725          ;MOVE TO THE OUTPUT BUFFER
1726
1727 006412 012500          MTOB:  MOV    (R5)+,R0        ;LOAD DEST.
1728 006414 012701 022066          MOV    #BUFFER,R1      ;LOAD R1
1729 006420 112021          1$:   MOVB   (R0)+,(R1)+      ;LOAD BYTE
1730 006422 100376          BPL    1$                ;BR UNTIL DONE
1731 006424 000205          RTS    R5                ;EXIT
1732

```

1733
1734
1735
1736
1737
1738

.SBTTL ASCII MESSAGES

:ASCII MESSAGES

: MESSAGES TO BE PRINTED BY "XPRNT" SUBROUTINE MUST BE TERMINATED
: BY A "377".

006426	006415	020012	055103	TITLE:	.ASCII	<15><15><12>\ CZVTOA M7142 SERIAL VIDEO BOARD ACCEPTANCE TEST\<15><12><0
006516	005015	005015	020040	M91:	.ASCII	<15><12><15><12>/ FULL SCREEN OF THE CHARACTER E/<15><12><377>
006566	005015	005015	020040	M93:	.ASCII	<15><12><15><12>/ SINGLE CHARACTER PER LINE /<15><12><377>
006632	005015	005015	020040	M94:	.ASCII	<15><12><15><12>/ ROTATING PATTERN /<15><12><377>
006665	015	006412	020012	M96:	.ASCII	<15><12><15><12>/ CURSOR MOTION TEST (CR,TAB,LF,BKSP,VT) /<15><12><377>
006746	005015	005015	020040	M97:	.ASCII	<15><12><15><12>/ SCROLL SPEED JUMP TEST/<15><12><377>
007006	005015	005015	020040	M99:	.ASCII	<15><12><15><12>/ COLUMN ADDRESS TEST/<15><12><377>
007043	015	006412	020012	M9221:	.ASCII	<15><12><15><12>/ CHARACTER ATTRIBUTE TEST (SO-SI SELECT)/<15><12><377>
007124	005015	005015	020040	M9222:	.ASCII	<15><12><15><12>/ CHARACTER ATTRIBUTE TEST (M.S.B. SELECT)/<15><12><377>
007206	005015	005015	020040	MKB:	.ASCII	<15><12><15><12>/ QUICK KEYBOARD CHARACTER TEST/<15><12>
007253	015	051012	046105	MKB1:	.ASCII	<15><12>/RELEASE THE "SHIFT" KEY/<15><12><377>
007310	042514	052106	052040	MKBA:	.ASCII	/LEFT TO RIGHT IN A ROW, DEPRESS ONE KEY AT A TIME/<15><12><377>
007375	015	006412	051412	MKBB2:	.ASCII	<15><12><15><12>/STARTING WITH THE TOP ROW EXCEPT THE BREAK KEY/<15><12>
007463	015	006412	051412	MKBC:	.ASCII	<15><12><15><12>/START WITH THE SECOND ROW/<15><12><377>
007524	005015	005015	052123	MKBD2:	.ASCII	<15><12><15><12>/START WITH THE THIRD ROW ,BEGIN ROW WITH "A" KEY/<15><1
007614	005015	005015	052123	MKBE:	.ASCII	<15><12><15><12>/START WITH THE FOURTH ROW EXCEPT SHIFT AND REPEAT/<15><
007705	015	006412	042012	MKBF:	.ASCII	<15><12><15><12>/DEPRESS THE SPACE BAR (THE FIFTH ROW)/<15><12><377>
007762	005015	047516	020127	MKBG:	.ASCII	<15><12>/NOW HOLD DOWN THE "LEFT-SHIFT" KEY/<15><12><377>
010031	015	047012	053517	MKBGA:	.ASCII	<15><12>/NOW HOLD DOWN THE "RIGHT-SHIFT" KEY/<15><12><377>
010101	015	047012	053517	MKBH:	.ASCII	<15><12>/NOW HOLD DOWN THE "CTRL" KEY/<15><12><377>
010142	005015	044103	051101	MKBQ:	.ASCII	<15><12>/CHARACTER WAS IN ERROR/<15><12>
010174	047507	042117	036440		.ASCII	/GOOD = /
010203	040	040	075	MKBQ1:	.BYTE	40,40,75,40
010207	040	040	040	MKBQA:	.BYTE	40,40,40,40,40
010214	040502	020104	020075		.ASCII	/BAD = /
010222	040	040	075	MKBQ2:	.BYTE	40,40,75,40
010226	040	040	040	MKBQB:	.BYTE	40,40,40,15,12,377,0
010235	015	045412	054505	MKBR:	.ASCII	<15><12>/KEYBOARD CHARACTER TEST COMPLETE/<15><12><377>
010303	015	044412	053116	INVLID:	.ASCII	<15><12>/INVALID BUS ADDRESS-PLEASE RETRY/<15><12>
010350	005015	005015	025040	MKBX:	.ASCII	<15><12><15><12>/ *** FULL KEYBOARD CHARACTER TEST ***/<15><12>
010423	015	051012	046105	MKBX1:	.ASCII	<15><12>/RELEASE THE "SHIFT" KEY/<15><12><377>
010460	005015	045440	054505	MKE:	.ASCII	<15><12>/ KEYBOARD ASCII AND OCTAL LOOP/<15><12>
010522	015	012			.BYTE	15,12
010524	044127	047105	040440		.ASCII	/WHEN A KEY IS DEPRESSED, THE ASCII CHARACTER AND/
010604	015	012			.BYTE	15,12
010606	052040	042510	052040		.ASCII	/ THE THREE DIGIT OCTAL CODE WILL BE ECHOED/
010661	015	012	377		.BYTE	15,12,377,0
010665	015	041412	040510	MKEA:	.ASCII	<15><12>/CHAR = /
010676	040	040	040	MKEA1:	.BYTE	40,40,40,75,40
010703	040	040	040	MKEB:	.BYTE	40,40,40,15,12,377,0
010712	005015	042513	041131	MKEH:	.ASCII	<15><12>/KEYBOARD ECHO LOOP/<15><12><377>
010742	056033	057433	005015	MQ0:	.ASCII	<33><134><33><137><15><12><12>/LOOP ON TEST PATTERN LETTER (A THRU M) ?
011027	033	015534	006537	MQ1:	.ASCII	<33><134><33><137><15><12><12>/START AT TEST PATTERN LETTER (A THRU M) ?
011115	015	005012	051124	MQ2:	.ASCII	<15><12><12>/TRY AGAIN /<15><12><377>
011136	005015	005015	020040	M98:	.ASCII	<15><12><15><12>/ DIRECT CURSOR ADDRESSING TEST /<377>
011205	014	000377		FRMFED:	.ASCII	<14><377>
011210	010	010	010	SCRL:	.BYTE	10,10,10,10,10,10,10,10,10,10
011223	040	051440	051103		.ASCII	/ SCROLL # /
011236	060	000	377	SCRNO:	.BYTE	60,0,377,0

```

011242 005015 044124 026505 ATTR8: .ASCII <15><12>/THE-CHARACTER-ATTRIBUTE(S)-ON-THIS-LINE-IS/
011316 040450 042522 026451 .ASCII2 /(ARE)-SELECTED(DESELECTED)-BY-'M.S.B'./<377>
011366 005015 044124 026505 ATTRS: .ASCII <15><12>/THE-CHARACTER-ATTRIBUTE(S)-ON-THIS-LINE-IS/
011442 040450 042522 026451 .ASCII2 /(ARE)-SELECTED(DESELECTED)-BY-'SO-SI'./<377>
011512 177416 000 SO: .ASCII2 <16><377>
011515 017 000377 SI: .ASCII2 <17><377>
011520 005015 043012 051111 WHAT0: .ASCII2 <15><12><12>/FIRST LINE DEVICE ADDRESS ? = /<377>
011563 015 005012 040514 WHAT1: .ASCII2 <15><12><12>/LAST LINE DEVICE ADDRESS (CR IF NONE) ? = /<377>
011642 005015 020040 020040 PASHED: .ASCII2 <15><12>/ PASS #/
011665 105 051122 051117 EM1: .ASCII2 /ERROR FLAG SET ON TRANSMITTER STATUS/
011732 047516 044440 050116 EM2: .ASCII2 /NO INPUT FLAG DETECTED/
011761 125 042516 050130 EM3: .ASCII2 /UNEXPECTED OR INCORRECT INPUT CHAR/
012024 051105 050122 020103 DH1: .ASCII2 /ERRPC VTNOW TSTNUM/
012053 105 051122 041520 DH3: .ASCII2 /ERRPC VTNOW TSTNUM EXPCT RECV/
012122 001116 001226 001230 DT1: $ERRPC,VTNOW,TSTNUM,0
012132 001116 001226 001230 DT3: $ERRPC,VTNOW,TSTNUM,$GDDAT,$BDDAT,0

```

.SBTTL KEYBOARD CHARACTER CODE TABLES

;THE ACTUAL KEYBOARD LAYOUT IS REQUIRED

```

012146 012166 012224 012262 V52RW: ROW12,ROW22,ROW32,ROW42,ROW12S,ROW12S,ROW12C

```

```

012164 100040 ROW5: .WORD 100040

```

;M7142 KEYBOARD EQUIVALENCES(LOWER CASE CHAR.)

```

012166 000033 000061 000062 ROW12: .WORD 33,61,62,63,64,65,66,67,70,71,60,55,75,140,100010
012224 000011 000161 000167 ROW22: .WORD 11,161,167,145,162,164,171,165,151,157,160,133,134,12,100177
012262 000141 000163 000144 ROW32: .WORD 141,163,144,146,147,150,152,153,154,73,47,173,100015
012314 000172 000170 000143 ROW42: .WORD 172,170,143,166,142,156,155,54,56,100057

```

;SHIFTED ROW CODES

```

012340 000033 000041 000100 ROW12S: .WORD 33,41,100,43,44,45,136,46,52,50,51,137,53,176,100010

```

;CONTROL ROW CODES

```

012376 000033 000021 000022 ROW12C: .WORD 33,21,22,23,24,25,26,27,30,31,20,15,35,0,100010

```

```

012434 005015 042513 020131 ERCHR: .ASCII <15><12>/KEY CODE ERROR/<377>
012455 123 040524 052122 INSTR: .ASCII /START 1ST ROW LEFT TO RIGHT/<15><12>
012512 051120 041517 042505 .ASCII /PROCEED TO LAST ROW/<15><12><377>

```

```

012540 005015 020012 040524 MTAB: .ASCII <15><12><12>/ TAB OVER-NO CHARACTER ERASE <CR>/<15>
012605 011 011 011 .BYTE 11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11,11
012631 011 011 011 .BYTE 11,11,11,11,11,11,11,11,11,11,11,11,11
012646 041474 037122 177415 .ASCII /<CR>/<15><377>
012654 036012 043114 005076 MBKSP: .ASCII <12>/<LF>/<12>/<LF>/<12>/<LF>/<12>/<LF>/
012700 041040 041501 051513 .ASCII / BACKSPACE OVER-NO CHAR. ERASE/
012736 010 010 010 .BYTE 10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10
012755 010 010 010 .BYTE 10,10,10,10,10,10,10,10,10,10,10,10,10,10,10,10
012774 010 010 010 .BYTE 10,10,10,10,10,10,10,10,10,10,377

```

013006	036015	051103	006476	MVT:	.ASCII	<15>/<CR>/<15><13>/<VT>/<13>/<VT>/
013026	036013	052126	005476		.ASCII	<13>/<VT>/<13>/<VT>/<13>/<VT><CR>/<15><377>
013053	015	005012	044124	MTEXT0:	.ASCII2	<15><12><12>/THIS SOFTWARE IS FURNISHED TO PURCHASER UNDER A LICENSE FOR
013157	015	047412	020116	MTEXT1:	.ASCII2	<15><12>/ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION/<
013260	005015	043117	042040	MTEXT2:	.ASCII2	<15><12>/OF DEC'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT/
013362	005015	051501	046440	MTEXT3:	.ASCII2	<15><12>/AS MAY OTHERWISE BE PROVIDED IN WRITING BY DEC./<377>
013445	015	005012	044124	MTEXT4:	.ASCII2	<15><12><12>/THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHO
013547	015	047012	052117	MTEXT5:	.ASCII2	<15><12>/NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL/<
013650	005015	050505	044525	MTEXT6:	.ASCII2	<15><12>/EQUIPMENT CORPORATION./<377>
013702	005015	042012	041505	MTEXT7:	.ASCII2	<15><12><12>/DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF
014002	005015	052111	020123	MTEXT8:	.ASCII	<15><12>/ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC./
014073	015	012	377		.BYTE	15,12,377,0

.SBTTL TTY INPUT ROUTINE

.ENABL LSB

.DSABL LSB

*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY

*CALL:

* RDCHR	:: INPUT A SINGLE CHARACTER FROM THE TTY
* RETURN HERE	:: CHARACTER IS ON THE STACK
	:: WITH PARITY BIT STRIPPED OFF

SRDCHR: MOV	(SP),-(SP)	:: PUSH DOWN THE PC
MOV	4(SP),2(SP)	:: SAVE THE PS
1\$: TSTB	@\$TKS	:: WAIT FOR
BPL	1\$	:: A CHARACTER
MOVB	@\$TKB,4(SP)	:: READ THE TTY
BIC	#^C<177>,4(SP)	:: GET RID OF JUNK IF ANY
CMP	4(SP),#23	:: IS IT A CONTROL-S?
BNE	3\$	:: BRANCH IF NO
2\$: TSTB	@\$TKS	:: WAIT FOR A CHARACTER
BPL	2\$	:: LOOP UNTIL ITS THERE
MOVB	@\$TKB,-(SP)	:: GET CHARACTER
BIC	#^C177,(SP)	:: MAKE IT 7-BIT ASCII
CMP	(SP)+,#21	:: IS IT A CONTROL-Q?
BNE	2\$	:: IF NOT DISCARD IT
BR	1\$	:: YES, RESUME
3\$: CMP	4(SP),#140	:: IS IT UPPER CASE?
BLT	4\$	:: BRANCH IF YES
CMP	4(SP),#175	:: IS IT A SPECIAL CHAR?
BGT	4\$	:: BRANCH IF YES
BIC	#40,4(SP)	:: MAKE IT UPPER CASE
4\$: RTI		:: GO BACK TO USER

*THIS ROUTINE WILL INPUT A STRING FROM THE TTY

*CALL:

* RDLIN	:: INPUT A STRING FROM THE TTY
* RETURN HERE	:: ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK

(1)
 1739
 1740
 1741
 1742
 1743
 1744
 1745
 1746
 1747
 1748
 1749
 1750
 1751
 1752
 1753
 1754
 1755
 1756
 1757
 1758
 1759
 1760
 1761
 1762
 1763
 1764
 1765
 1766
 1767
 1768
 1769
 1770
 1771
 1772
 1773
 1774
 1775
 1776
 1777
 1778
 1779

```

1780 ;* ;:TERMINATOR WILL BE A BYTE OF ALL 0'S
1781
1782 014220 010346 $RDLIN: MOV R3,-(SP) ;:SAVE R3
1783 014222 012703 014326 1$: MOV #TTYIN,R3 ;:GET ADDRESS
1784 014226 022703 014336 2$: CMP #TTYIN+8.,R3 ;:BUFFER FULL?
1785 014232 101405 BLOS 4$ ;:BR IF YES
1786 014234 104406 RDCHR ;:GO READ ONE CHARACTER FROM THE TTY
1787 014236 112613 MOVB (SP)+,(R3) ;:GET CHARACTER
1788 014240 122713 000177 10$: CMPB #177,(R3) ;:IS IT A RUBOUT
1789 014244 001003 BNE 3$ ;:SKIP IF NOT
1790 014246 104401 001166 4$: TYPE ,8QUES ;:TYPE A '?'
1791 014252 000763 BR 1$ ;:CLEAR THE BUFFER AND LOOP
1792 014254 111337 014324 3$: MOVB (R3),9$ ;:ECHO THE CHARACTER
1793 014260 104401 014324 TYPE ,9$
1794 014264 122723 000015 CMPB #15,(R3)+ ;:CHECK FOR RETURN
1795 014270 001356 BNE 2$ ;:LOOP IF NOT RETURN
1796 014272 105063 177777 CLRB -1(R3) ;:CLEAR RETURN (THE 15)
1797 014276 104401 001170 TYPE ,8LF ;:TYPE A LINE FEED
1798 014302 012603 MOV (SP)+,R3 ;:RESTORE R3
1799 014304 011646 MOV (SP),-(SP) ;:ADJUST THE STACK AND PUT ADDRESS OF THE
1800 014306 016666 000004 000002 MOV 4(SP),2(SP) ;: FIRST ASCII CHARACTER ON IT
1801 014314 012766 014326 000004 MOV #TTYIN,4(SP)
1802 014322 000002 RTI ;:RETURN
1803 014324 000 9$: .BYTE 0 ;:STORAGE FOR ASCII CHAR. TO TYPE
1804 014325 000 .BYTE 0 ;:TERMINATOR
1805 014326 000010 $TTYIN: .BLKB 8. ;:RESERVE 8 BYTES FOR TTY INPUT
1806 014336 052536 005015 000 $CNTLU: .ASCIZ /^U/<15><12> ;:CONTROL 'U'
1807 014343 136 006507 000012 $CNTLG: .ASCIZ /^G/<15><12> ;:CONTROL 'G'
1808 014350 005015 053523 020122 $MSWR: .ASCIZ <15><12>/SWR = /
1809 014356 020075 000
1810 014361 040 047040 053505 $MNEW: .ASCIZ / NEW = /
1811 014366 036440 000040
1812 .SBTTL READ AN OCTAL NUMBER FROM THE TTY
1813
1814 ;:*****
1815 ;:THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
1816 ;:CHANGE IT TO BINARY.
1817 ;:THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
1818 ;:OCTAL DIGITS. IF AN ILLEGAL CHARACTER IS READ A '?' WILL BE TYPED
1819 ;:FOLLOWED BY A CARRIAGE RETURN-LINE FEED. THE COMPLETE NUMBER MUST
1820 ;:THEN BE RETYPED. THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN.
1821 ;:CALL:
1822 ;* RDOCT ;:READ AN OCTAL NUMBER
1823 ;* RETURN HERE ;:LOW ORDER BITS ARE ON TOP OF THE STACK
1824 ;* ;:HIGH ORDER BITS ARE IN $HI OCT
1825
1826 014372 011646 $RDOCT: MOV (SP),-(SP) ;:PROVIDE SPACE FOR THE
1827 014374 016666 000004 000002 MOV 4(SP),2(SP) ;:INPUT NUMBER
1828 014402 010046 MOV R0,-(SP) ;:PUSH R0 ON STACK
1829 014404 010146 MOV R1,-(SP) ;:PUSH R1 ON STACK
1830 014406 010246 MOV R2,-(SP) ;:PUSH R2 ON STACK
1831 014410 104407 1$: RDLIN ;:READ AN ASCII LINE
1832 014412 01260C MOV (SP)+,R0 ;:GET ADDRESS OF 1ST CHARACTER
1833 014414 010037 014520 MOV R0,5$ ;:AND SAVE IT
1834 014420 005001 CLR R1 ;:CLEAR DATA WORD
1835 014422 005002 CLR R2

```

```

1836 014424 112046      2$:  MOVB  (R0)+, -(SP)  ;; PICKUP THIS CHARACTER
1837 014426 001420      BEQ    3$          ;; IF ZERO GET OUT
1838 014430 122716 000060  CMPB  #'0,(SP)    ;; MAKE SURE THIS CHARACTER
1839 014434 003026      BGT    4$          ;; IS AN OCTAL DIGIT
1840 014436 122716 000067  CMPB  #'7,(SP)
1841 014442 002423      BLT    4$
1842 014444 006301      ASL    R1            ;; *2
1843 014446 006102      ROL    R2
1844 014450 006301      ASL    R1            ;; *4
1845 014452 006102      ROL    R2
1846 014454 006301      ASL    R1            ;; *8
1847 014456 006102      ROL    R2
1848 014460 042716 177770  BIC    #'^C7,(SP)    ;; STRIP THE ASCII JUNK
1849 014464 062601      ADD    (SP)+, R1    ;; ADD IN THIS DIGIT
1850 014466 000756      BR     2$          ;; LOOP
1851 014470 005726      3$:  TST    (SP)+    ;; CLEAN TERMINATOR FROM STACK
1852 014472 010166 000012  MOV    R1, 12(SP)    ;; SAVE THE RESULT
1853 014476 010237 014530  MOV    R2, $HIOCT
1854 014502 012602      MOV    (SP)+, R2    ;; POP STACK INTO R2
1855 014504 012601      MOV    (SP)+, R1    ;; POP STACK INTO R1
1856 014506 012600      MOV    (SP)+, R0    ;; POP STACK INTO R0
1857 014510 000002      RTI          ;; RETURN
1858 014512 005726      4$:  TST    (SP)+    ;; CLEAN PARTIAL FROM STACK
1859 014514 105010      CLRB  (R0)        ;; SET A TERMINATOR
1860 014516 104401      TYPE          ;; TYPE UP THRU THE BAD CHAR.
1861 014520 000000      5$:  .WORD  0
1862 014522 104401 001166  TYPE          $QUES    ;; '?' 'CR' & 'LF'
1863 014526 000730      BR     1$          ;; TRY AGAIN
1864 014530 000000      $HIOCT: .WORD  0    ;; HIGH ORDER BITS GO HERE
1865 014532 032777 001000 164400 MSCOPE: BIT  #BIT9, @SWR    ;; TEST BIT 7
1866 014540 001403      BEQ    $SCOPE    ;; NOT SET
1867 014542 005737 005154      TST    LOOP    ;; TEST LOOP
1868 014546 001041      BNE    $OVER    ;; SET LOOP ON TEST
1869
1870      .SBTTL  SCOPE HANDLER ROUTINE
1871
1872      ;; *****
1873      ;; *THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
1874      ;; *AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
1875      ;; *AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
1876      ;; *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
1877      ;; *SW14=1      LOOP ON TEST
1878      ;; *SW08=1      LOOP ON TEST IN SWR<7:0>
1879      ;; *CALL
1880      ;; *      SCOPE          ;; SCOPE=10T
1881
1882 014550      $SCOPE:
1883 014550 032777 040000 164362 1$:  BIT  #BIT14, @SWR    ;; LOOP ON PRESENT TEST?
1884 014556 001035      BNE    $OVER    ;; YES IF SW14=1
1885      ;; ***** START OF CODE FOR THE XOR TESTER *****
1886 014560 000416      $XTSTR: BR     6$    ;; IF RUNNING ON THE 'XOR' TESTER CHANGE
1887      ;; THIS INSTRUCTION TO A 'NOP' (NOP=240)
1888 014562 013746 000004      MOV    @#ERRVEC, -(SP)    ;; SAVE THE CONTENTS OF THE ERROR VECTOR
1889 014566 012737 014606 000004      MOV    #55, @#ERRVEC    ;; SET FOR TIMEOUT
1890 014574 005737 177060      TST    @#177060    ;; TIME OUT ON XOR?
1891 014600 012637 000004      MOV    (SP)+, @#ERRVEC    ;; RESTORE THE ERROR VECTOR

```

```

1892 014604 000414          BR    $SVLAD          ;;GO TO THE NEXT TEST
1893 014606 022626          5$:  CMP    (SP)+,(SP)+      ;;CLEAR THE STACK AFTER A TIME OUT
1894 014610 012637 000004    MOV    (SP)+,@#ERRVEC      ;;RESTORE THE ERROR VECTOR
1895 014614 000416          BR    $OVER           ;;LOOP ON THE PRESENT TEST
1896 014616                6$:;#####END OF CODE FOR THE XOR TESTER#####
1897 014616 032777 000400 164314  BIT    #BIT08,@SWR      ;;LOOP ON SPEC. TEST?
1898 014624 001404          BEQ    $SVLAD          ;;BR IF NO
1899 014626 127737 164306 001102  CMPB   @SWR,$STNM      ;;ON THE RIGHT TEST?   SWR<7:0>
1900 014634 001406          BEQ    $OVER           ;;BR IF YES
1901 014636 105237 001102    $SVLAD: INCB   $STNM          ;;COUNT TEST NUMBERS
1902 014642 011637 001106    MOV    (SP),$LPADR       ;;SAVE SCOPE LOOP ADDRESS
1903 014646 105037 001103    CLRB   $ERFLG          ;;ZERO THE ERROR FLAG
1904 014652 013777 001102 164262 $OVER:  MOV    $STNM,@DISPLAY  ;;DISPLAY TEST NUMBER
1905 014660 013716 001106    MOV    $LPADR,(SP)       ;;FUDGE RETURN ADDRESS
1906 014664 000002          RTI                    ;;FIXES PS

```

.SBTTL ERROR HANDLER ROUTINE

```

1907
1908
1909
1910
1911
1912
1913
1914
1915
1916
1917
1918
1919
1920
1921
1922
1923
1924
1925
1926
1927
1928
1929
1930
1931
1932
1933
1934
1935
1936
1937
1938
1939
1940
1941
1942
1943
1944
1945
1946
1947

```

*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
*AND GO TO \$ERRTYP ON ERROR
*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
*SW15=1 HALT ON ERROR
*SW13=1 INHIBIT ERROR TYPEOUTS
*CALL
* ERROR N ;;ERROR=EMT AND N=ERROR ITEM NUMBER

```

1920 014666 113737 001102 001230 $ERROR:
1921 014666 105237 001103 7$:  MOVB   $STNM,$TSTNUM
1922 014674 001775          INCB   $ERFLG          ;;SET THE ERROR FLAG
1923 014700 013777 001102 164232 BEQ    7$          ;;DON'T LET THE FLAG GO TO ZERO
1924 014702 005237 001112          MOV    $STNM,@DISPLAY  ;;DISPLAY TEST NUMBER AND ERROR FLAG
1925 014710 011637 001116          INC    $ERTTL          ;;INC THE ERROR COUNT
1926 014714 162737 000002 001116 MOV    (SP),$ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
1927 014720 117737 164164 001114 SUB    #2,$ERRPC
1928 014734 032777 020000 164176 MOVB   @ERRPC,$ITEMB    ;;STRIP AND SAVE THE ERROR ITEM CODE
1929 014742 001004          BIT    #BIT13,@SWR      ;;SKIP TYPEOUT IF SET
1930 014744 004737 015000          BNE    20$          ;;SKIP TYPEOUTS
1931 014750 104401 001167          JSR    PC,$ERRTYP      ;;GO TO USER ERROR ROUTINE
1932 014754          TYPE    ,SCRLF
1933 014754 005777 164160 20$:  TST    @SWR          ;;HALT ON ERROR
1934 014760 100001          BPL    3$          ;;SKIP IF CONTINUE
1935 014762 000000          HALT                    ;;HALT ON ERROR!
1936 014764 022737 003764 000042 3$:  CMP    #SENDAD,@#42      ;;ACT-11 AUTO-ACCEPT?
1937 014772 001001          BNE    6$          ;;BRANCH IF NO
1938 014774 000000          HALT                    ;;YES
1939 014776          6$:  RTI                    ;;RETURN

```

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

```

*****
*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH

```

```

1948 ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE 'ERROR TABLE' ($ERRTB),
1949 ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
1950
1951 $ERRTYP:
1952 TYPE ,SCLF ;:'CARRIAGE RETURN' & 'LINE FEED'
1953 MOV RO,-(SP) ;:SAVE RO
1954 CLR RO ;:PICKUP THE ITEM INDEX
1955 BISB @#$ITEMB,RO
1956 BNE 1$ ;:IF ITEM NUMBER IS ZERO, JUST
1957 ;:TYPE THE PC OF THE ERROR
1958 MOV $ERRPC,-(SP) ;:SAVE $ERRPC FOR TYPEOUT
1959 ;:ERROR ADDRESS
1960 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1961 BR 6$ ;:GET OUT
1962 1$: DEC RO ;:ADJUST THE INDEX SO THAT IT WILL
1963 ASL RO ;:WORK FOR THE ERROR TABLE
1964 ASL RO
1965 ASL RO
1966 ADD #$ERRTB,RO ;:FORM TABLE POINTER
1967 MOV (RO)+,2$ ;:PICKUP 'ERROR MESSAGE' POINTER
1968 BEQ 3$ ;:SKIP TYPEOUT IF NO POINTER
1969 TYPE ;:TYPE THE 'ERROR MESSAGE'
1970 2$: .WORD 0 ;:'ERROR MESSAGE' POINTER GOES HERE
1971 TYPE ,SCLF ;:'CARRIAGE RETURN' & 'LINE FEED'
1972 3$: MOV (RO)+,4$ ;:PICKUP 'DATA HEADER' POINTER
1973 BEQ 5$ ;:SKIP TYPEOUT IF 0
1974 TYPE ;:TYPE THE 'DATA HEADER'
1975 4$: .WORD 0 ;:'DATA HEADER' POINTER GOES HERE
1976 TYPE ,SCLF ;:'CARRIAGE RETURN' & 'LINE FEED'
1977 5$: MOV (RO),RO ;:PICKUP 'DATA TABLE' POINTER
1978 BNE 7$ ;:GO TYPE THE DATA
1979 6$: MOV (SP)+,RO ;:RESTORE RO
1980 TYPE ,SCLF ;:'CARRIAGE RETURN' & 'LINE FEED'
1981 RTS PC ;:RETURN
1982 7$:
1983 MOV @ (RO)+,-(SP) ;:SAVE @ (RO)+ FOR TYPEOUT
1984 TYPOC ;:GO TYPE--OCTAL ASCII(ALL DIGITS)
1985 TST (RO) ;:IS THERE ANOTHER NUMBER?
1986 BEQ 6$ ;:BR IF NO
1987 TYPE ,8$ ;:TYPE TWO(2) SPACES
1988 BR 7$ ;:LOOP
1989 8$: .ASCIZ / / ;:TWO(2) SPACES
1990 .EVEN
1991
1992 .SBTTL TYPE ROUTINE
1993
1994 ;:*****
1995 ;:*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
1996 ;:*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
1997 ;:*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
1998 ;:*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
1999 ;:*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
2000 ;*
2001 ;*CALL:
2002 ;*1) USING A TRAP INSTRUCTION
2003 ;* TYPE ,MESADR ;:MESADR IS FIRST ADDRESS OF AN ASCIZ STRING

```

```

2004      ;*OR
2005      ;*      TYPE
2006      ;*      MESADR
2007      ;*
2008
2009 015134 105737 001157 $TYPE: TSTB $TPFLG      ;;IS THERE A TERMINAL?
2010 015140 100002      BPL 1$      ;;BR IF YES
2011 015142 000000      HALT      ;;HALT HERE IF NO TERMINAL
2012 015144 000407      BR 3$      ;;LEAVE
2013 015146 010046 1$: MOV R0,-(SP)      ;;SAVE R0
2014 015150 017600 000002 MOV @2(SP),R0      ;;GET ADDRESS OF ASCIZ STRING
2015 015154 112046 2$: MOVB (R0)+,-(SP)      ;;PUSH CHARACTER TO BE TYPED ONTO STACK
2016 015156 001005      BNE 4$      ;;BR IF IT ISN'T THE TERMINATOR
2017 015160 005726      TST (SP)+      ;;IF TERMINATOR POP IT OFF THE STACK
2018 015162 012600 60$: MOV (SP)+,R0      ;;RESTORE R0
2019 015164 062716 000002 3$: ADD #2,(SP)      ;;ADJUST RETURN PC
2020 015170 000002      RTI      ;;RETURN
2021 015172 122716 000011 4$: CMPB #HT,(SP)      ;;BRANCH IF <HT>
2022 015176 001430      BEQ 8$
2023 015200 122716 000200      CMPB #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
2024 015204 001006      BNE 5$
2025 015206 005726      TST (SP)+      ;;POP <CR><LF> EQUIV
2026 015210 104401      TYPE      ;;TYPE A CR AND LF
2027 015212 001167      $CRLF
2028 015214 105037 015350      CLRB $CHARCNT      ;;CLEAR CHARACTER COUNT
2029 015220 000755      BR 2$      ;;GET NEXT CHARACTER
2030 015222 004737 015304 5$: JSR PC,$TYPEC      ;;GO TYPE THIS CHARACTER
2031 015226 123726 001156 6$: CMPB $FILLC,(SP)+      ;;IS IT TIME FOR FILLER CHARS.?
2032 015232 001350      BNE 2$      ;;IF NO GO GET NEXT CHAR.
2033 015234 013746 001154      MOV $NULL,-(SP)      ;;GET # OF FILLER CHARS. NEEDED
2034      ;;AND THE NULL CHAR.
2035 015240 105366 000001 7$: DECB 1(SP)      ;;DOES A NULL NEED TO BE TYPED?
2036 015244 002770      BLT 6$      ;;BR IF NO--GO POP THE NULL OFF OF STACK
2037 015246 004737 015304      JSR PC,$TYPEC      ;;GO TYPE A NULL
2038 015252 105337 015350      DECB $CHARCNT      ;;DO NOT COUNT AS A COUNT
2039 015256 000770      BR 7$      ;;LOOP
2040
2041      ;HORIZONTAL TAB PROCESSOR
2042
2043 015260 112716 000040 8$: MOVB #'',(SP)      ;;REPLACE TAB WITH SPACE
2044 015264 004737 015304 9$: JSR PC,$TYPEC      ;;TYPE A SPACE
2045 015270 132737 000007 015350      BITB #7,$CHARCNT      ;;BRANCH IF NOT AT
2046 015276 001372      BNE 9$      ;;TAB STOP
2047 015300 005726      TST (SP)+      ;;POP SPACE OFF STACK
2048 015302 000724      BR 2$      ;;GET NEXT CHARACTER
2049 015304 105777 163640 $TYPEC: TSTB @2$TPS      ;;WAIT UNTIL PRINTER IS READY
2050 015310 100375      BPL $TYPEC
2051 015312 116677 000002 163632      MOVB 2(SP),@2$TPB      ;;LOAD CHAR TO BE TYPED INTO DATA REG.
2052 015320 122766 000015 000002      CMPB #CR,2(SP)      ;;IS CHARACTER A CARRIAGE RETURN?
2053 015326 001003      BNE 1$      ;;BRANCH IF NO
2054 015330 105037 015350      CLRB $CHARCNT      ;;YES--CLEAR CHARACTER COUNT
2055 015334 000406      BR $TYPEX      ;;EXIT
2056 015336 122766 000012 000002 1$: CMPB #LF,2(SP)      ;;IS CHARACTER A LINE FEED?
2057 015344 001402      BEQ $TYPEX      ;;BRANCH IF YES
2058 015346 105227      INCB (PC)+      ;;COUNT THE CHARACTER
2059 015350 000000      $CHARCNT: .WORD 0      ;;CHARACTER COUNT STORAGE

```

```

2060 015352 000207 $TYPEX: RTS PC
2061
2062
2063 .SBTTL BINARY TO OCTAL (ASCII) AND TYPE
2064
2065 *****
2066 *THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
2067 *OCTAL (ASCII) NUMBER AND TYPE IT.
2068 *$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
2069 *CALL:
2070 *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
2071 *      TYPOS      ;;CALL FOR TYPEOUT
2072 *      .BYTE      N      ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
2073 *      .BYTE      M      ;;M=1 OR 0
2074 *                               ;;1=TYPE LEADING ZEROS
2075 *                               ;;0=SUPPRESS LEADING ZEROS
2076
2077 *$TYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
2078 *$TYPOS OR $TYPOC
2079 *CALL:
2080 *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
2081 *      TYPON      ;;CALL FOR TYPEOUT
2082
2083 *$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
2084 *CALL:
2085 *      MOV      NUM,-(SP)      ;;NUMBER TO BE TYPED
2086 *      TYPOC      ;;CALL FOR TYPEOUT
2087
2088 015354 017646 000000 015577 $TYPOS: MOV      @ (SP),-(SP)      ;;PICKUP THE MODE
2089 015360 116637 000001 $TYPOS: MOVB      1(SP),%OFILL      ;;LOAD ZERO FILL SWITCH
2090 015366 112637 015601 $TYPOS: MOVB      (SP)+,%SOMODE+1      ;;NUMBER OF DIGITS TO TYPE
2091 015372 062716 000002 $TYPOS: ADD      #2,(SP)      ;;ADJUST RETURN ADDRESS
2092 015376 000406 $TYPOS: BR      $TYPON
2093 015400 112737 000001 015577 $TYPOC: MOVB      #1,%OFILL      ;;SET THE ZERO FILL SWITCH
2094 015406 112737 000006 015601 $TYPOC: MOVB      #6,%SOMODE+1      ;;SET FOR SIX(6) DIGITS
2095 015414 112737 000005 015576 $TYPON: MOVB      #5,%SOCNT      ;;SET THE ITERATION COUNT
2096 015422 010346 $TYPON: MOV      R3,-(SP)      ;;SAVE R3
2097 015424 010446 $TYPON: MOV      R4,-(SP)      ;;SAVE R4
2098 015426 010546 $TYPON: MOV      R5,-(SP)      ;;SAVE R5
2099 015430 113704 015601 $TYPON: MOVB      %SOMODE+1,R4      ;;GET THE NUMBER OF DIGITS TO TYPE
2100 015434 005404 $TYPON: NEG      R4
2101 015436 062704 000006 $TYPON: ADD      #6,R4      ;;SUBTRACT IT FOR MAX. ALLOWED
2102 015442 110437 015600 $TYPON: MOVB      R4,%SOMODE      ;;SAVE IT FOR USE
2103 015446 113704 015577 $TYPON: MOVB      %OFILL,R4      ;;GET THE ZERO FILL SWITCH
2104 015452 016605 000012 $TYPON: MOV      12(SP),R5      ;;PICKUP THE INPUT NUMBER
2105 015456 005003 $TYPON: CLR      R3      ;;CLEAR THE OUTPUT WORD
2106 015460 006105 1$: $TYPON: ROL      R5      ;;ROTATE MSB INTO 'C'
2107 015462 000404 $TYPON: BR      3$      ;;GO DO MSB
2108 015464 006105 2$: $TYPON: ROL      R5      ;;FORM THIS DIGIT
2109 015466 006105 $TYPON: ROL      R5
2110 015470 006105 $TYPON: ROL      R5
2111 015472 010503 $TYPON: MOV      R5,R3
2112 015474 006103 3$: $TYPON: ROL      R3      ;;GET LSB OF THIS DIGIT
2113 015476 105337 015600 $TYPON: DECB      %SOMODE      ;;TYPE THIS DIGIT?
2114 015502 100016 $TYPON: BPL      7$      ;;BR IF NO
2115 015504 042703 177770 $TYPON: BIC      #177770,R3      ;;GET RID OF JUNK

```

2116	015510	001002		BNE	4\$	::TEST FOR 0
2117	015512	005704		TST	R4	::SUPPRESS THIS 0?
2118	015514	001403		BEQ	5\$	::BR IF YES
2119	015516	005204		4\$: INC	R4	::DON'T SUPPRESS ANYMORE 0'S
2120	015520	052703	000060	BIS	#'0,R3	::MAKE THIS DIGIT ASCII
2121	015524	052703	000040	5\$: BIS	#',R3	::MAKE ASCII IF NOT ALREADY
2122	015530	110337	015574	MOVB	R3,8\$	::SAVE FOR TYPING
2123	015534	104401	015574	TYPE	,8\$	::GO TYPE THIS DIGIT
2124	015540	105337	015576	7\$: DECB	\$OCNT	::COUNT BY 1
2125	015544	003347		BGT	2\$	::BR IF MORE TO DO
2126	015546	002402		BLT	6\$	::BR IF DONE
2127	015550	005204		INC	R4	::INSURE LAST DIGIT ISN'T A BLANK
2128	015552	000744		BR	2\$	::GO DO THE LAST DIGIT
2129	015554	012605		6\$: MOV	(SP)+,R5	::RESTORE R5
2130	015556	012604		MOV	(SP)+,R4	::RESTORE R4
2131	015560	012603		MOV	(SP)+,R3	::RESTORE R3
2132	015562	016666	000002 000004	MOV	2(SP),4(SP)	::SET THE STACK FOR RETURNING
2133	015570	012616		MOV	(SP)+,(SP)	
2134	015572	000002		RTI		::RETURN
2135	015574	000		8\$: .BYTE	0	::STORAGE FOR ASCII DIGIT
2136	015575	000		.BYTE	0	::TERMINATOR FOR TYPE ROUTINE
2137	015576	000		\$OCNT: .BYTE	0	::OCTAL DIGIT COUNTER
2138	015577	000		\$OFILL: .BYTE	0	::ZERO FILL SWITCH
2139	015600	000000		\$OMODE: .WORD	0	::NUMBER OF DIGITS TO TYPE
2140				.SBTIL	RANDOM NUMBER GENERATOR ROUTINE	
2141						
2142						
2143						
2144						
2145						
2146						
2147						
2148						
2149						
2150						
2151	015602			\$RAND:		
2152	015602	010046		MOV	R0,-(SP)	::PUSH R0 ON STACK
2153	015604	010146		MOV	R1,-(SP)	::PUSH R1 ON STACK
2154	015606	010246		MOV	R2,-(SP)	::PUSH R2 ON STACK
2155	015610	013700	015702	MOV	\$LONUM,R0	::SET R0 WITH LOW
2156	015614	013701	015700	MOV	\$HINUM,R1	::SET R1 WITH HIGH
2157	015620	012702	177771	MOV	#-7,R2	::SET SHIFT COUNT
2158	015624	006300		1\$: ASL	R0	::SHIFT R0 LEFT AND
2159	015626	006101		ROL	R1	::ROTATE CARRY INTO R1 AND
2160	015630	005202		INC	R2	::CHECK FOR DONE
2161	015632	001374		BNE	1\$	::CONTINUE SHIFT LOOP
2162	015634	063700	015702	ADD	\$LONUM,R0	::ADD NUMBER TO MAKE X 129
2163	015640	005501		ADC	R1	::PROPOGATE CARRY
2164	015642	063701	015700	ADD	\$HINUM,R1	::ADD NUMBER TO MAKE X 129
2165	015646	062700	001057	ADD	#1057,R0	::ADD LOW CONSTANT
2166	015652	005501		ADC	R1	::PROPOGATE CARRY
2167	015654	062701	047401	ADD	#47401,R1	::ADD HIGH CONSTANT
2168	015660	010037	015702	MOV	R0,\$LONUM	::SAVE R0
2169	015664	010137	015700	MOV	R1,\$HINUM	::SAVE R1
2170	015670	012602		MOV	(SF)+,R2	::POP STACK INTO R2
2171	015672	012601		MOV	(SF)+,R1	::POP STACK INTO R1

```

2172 015674 012600          MOV      (SP)+,R0          ;;POP STACK INTO R0
2173 015676 000207          RTS      PC              ;;RETURN
2174 015700 176543          $HINUM: .WORD 176543
2175 015702 123456          $LONUM: .WORD 123456
2176                          .SBTTL  TRAP DECODER
2177
2178                          ;;*****
2179                          ;;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE 'TRAP' INSTRUCTION
2180                          ;;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
2181                          ;;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
2182                          ;;*GO TO THAT ROUTINE.
2183
2184 015704 010046          $TRAP: MOV      R0,-(SP)          ;;SAVE R0
2185 015706 016600 000002    MOV      2(SP),R0          ;;GET TRAP ADDRESS
2186 015712 005740          TST      -(R0)              ;;BACKUP BY 2
2187 015714 111000          MOVB     (R0),R0            ;;GET RIGHT BYTE OF TRAP
2188 015716 006300          ASL      R0                  ;;POSITION FOR INDEXING
2189 015720 016000 015740    MOV      $TRPAD(R0),R0      ;;INDEX TO TABLE
2190 015724 000200          RTS      R0                  ;;GO TO ROUTINE
2191
2192
2193                          ;;THIS IS USE TO HANDLE THE 'GETPRI' MACRO
2194
2195 015726 011646          $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
2196 015730 016666 000004 000002    MOV     4(SP),2(SP)    ;;MOVE THE PSW DOWN
2197 015736 000002          RTI                          ;;RESTORE THE PSW
2198
2199                          .SBTTL  TRAP TABLE
2200
2201                          ;;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
2202                          ;;*BY THE 'TRAP' INSTRUCTION.
2203
2204                          ;      ROUTINE
2205                          ;      -----
2206 015740 015726          $TRPAD: .WORD  $TRAP2
2207 015742 015134          $TYPE  ;;CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
2208 015744 015400          $TYPOC ;;CALL=TYPOC     TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
2209 015746 015354          $TYPOS ;;CALL=TYPOS     TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
2210 015750 015414          $TYPON ;;CALL=TYPON     TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
2211 015752 006110          $TYPDS ;;CALL=TYPDS     TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
2212
2213
2214 015754 014100          $RDCHR  ;;CALL=RDCHR     TRAP+6(104406)  TTY TYPEIN CHARACTER ROUTINE
2215 015756 014220          $RDLIN  ;;CALL=RDLIN     TRAP+7(104407)  TTY TYPEIN STRING ROUTINE
2216 015760 014372          $RDOCT  ;;CALL=RDOCT     TRAP+10(104410) READ AN OCTAL NUMBER FROM TTY
2217
2218                          .SBTTL  POWER DOWN AND UP ROUTINES
2219
2220                          ;;*****
2221 015762 012737 016122 000024    $PWDRN: MOV     # $ILLUP, @#PWRVEC ;;SET FOR FAST UP
2222 015770 012737 000340 000026    MOV     #340, @#PWRVEC+2 ;;PRIO:7
2223 015776 010046          MOV     R0,-(SP)            ;;PUSH R0 ON STACK
2224 016000 010146          MOV     R1,-(SP)            ;;PUSH R1 ON STACK
2225 016002 010246          MOV     R2,-(SP)            ;;PUSH R2 ON STACK
2226 016004 010346          MOV     R3,-(SP)            ;;PUSH R3 ON STACK
2227 016006 010446          MOV     R4,-(SP)            ;;PUSH R4 ON STACK

```

```

2228 016010 010546      MOV      R5,-(SP)      ;;PUSH R5 ON STACK
2229 016012 017746 163122  MOV      @SWR,-(SP)      ;;PUSH @SWR ON STACK
2230 016016 010637 016126  MOV      SP,$SAVR6      ;;SAVE SP
2231 016022 012737 016034 000024  MOV      $SPWRUP,@#PWRVEC ;;SET UP VECTOR
2232 016030 000000      HALT
2233 016032 000776      BR       .-2      ;;HANG UP
2234
2235      ;*****
2236      ;POWER UP ROUTINE
2237 016034 012737 016122 000024  $PWRUP: MOV      $SILLUP,@#PWRVEC ;;SET FOR FAST DOWN
2238 016042 013706 016126      MOV      $SAVR6,SP      ;;GET SP
2239 016046 005037 016126      CLR      $SAVR6      ;;WAIT LOOP FOR THE TTY
2240 016052 005237 016126 1$:      INC      $SAVR6      ;;WAIT FOR THE INC
2241 016056 001375      BNE      1$      ;;OF WORD
2242 016060 012677 163054      MOV      (SP)+,@SWR      ;;POP STACK INTO @SWR
2243 016064 012605      MOV      (SP)+,R5      ;;POP STACK INTO R5
2244 016066 012604      MOV      (SP)+,R4      ;;POP STACK INTO R4
2245 016070 012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
2246 016072 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
2247 016074 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
2248 016076 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
2249 016100 012737 015762 000024  MOV      $SPWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
2250 016106 012737 000340 000026  MOV      #340,@#PWRVEC+2 ;;PRIO:7
2251 016114 104401      TYPE      ;;REPORT THE POWER FAILURE
2252 016116 016130  $PWRMG: .WORD    $POWER      ;;POWER FAIL MESSAGE POINTER
2253 016120 000002      RTI
2254 016122 000000  $SILLUP: HALT      ;;THE POWER UP SEQUENCE WAS STARTED
2255 016124 000776      BR       .-2      ;; BEFORE THE POWER DOWN WAS COMPLETE
2256 016126 000000  $SAVR6: 0      ;;PUT THE SP HERE
2257 016130 005015 047520 042527  $POWER: .ASCIIZ  <15><12>'POWER'
2258 016136 000122
2259
2260 016140 000000  RETRY:  .WORD    0
2261
2262      ; THIS TEXT IS THE SOURCE OF THE CHARACTERS TO BE PLACED IN THE MESSAGE
2263      ; AREA 'BUFFER' FOR THE DIRECT CURSOR ADDRESSING TEST (E).
2264      ; THE BUFFER IS 2000. CHARS. LONG FOR TEST 'E' AND SHOULD BE TERMINATED
2265      ; BY 3 OR 4 '377''S AS A TERMINATOR.
2266
2267      .NLIST  BEX
MSGTXT:
(1) 016142 033515 032061 026462 .ASCII  \M7142-DIRECT-CURSOR-ADDRESSING-TEST-DIGI\
(1) 016212 040524 026514 050505 .ASCII  \TAL-EQUIPMENT-CORP.-MERRIMACK-N.H.-M7142-\
(1) 016263 115 030467 031064 .ASCII  \M7142-DIRECT-CURSOR-ADDRESSING-TEST-DIGI\
(1) 016333 124 046101 042455 .ASCII  \TAL-EQUIPMENT-CORP.-MERRIMACK-N.H.-M7142-\
(1) 016404 033515 032061 026462 .ASCII  \M7142-DIRECT-CURSOR-ADDRESSING-TEST-DIGI\
(1) 016454 040524 026514 050505 .ASCII  \TAL-EQUIPMENT-CORP.-MERRIMACK-N.H.-M7142-\
(1) 016525 115 030467 031064 .ASCII  \M7142-DIRECT-CURSOR-ADDRESSING-TEST-DIGI\
(1) 016575 124 046101 042455 .ASCII  \TAL-EQUIPMENT-CORP.-MERRIMACK-N.H.-M7142-\
(1) 016646 033515 032061 026462 .ASCII  \M7142-DIRECT-CURSOR-ADDRESSING-TEST-DIGI\
(1) 016716 040524 026514 050505 .ASCII  \TAL-EQUIPMENT-CORP.-MERRIMACK-N.H.-M7142-\
(1) 016767 115 030467 031064 .ASCII  \M7142-DIRECT-CURSOR-ADDRESSING-TEST-DIGI\
(1) 017037 124 046101 042455 .ASCII  \TAL-EQUIPMENT-CORP.-MERRIMACK-N.H.-M7142-\
(1) 017110 033515 032061 026462 .ASCII  \M7142-DIRECT-CURSOR-ADDRESSING-TEST-DIGI\
(1) 017160 040524 026514 050505 .ASCII  \TAL-EQUIPMENT-CORP.-MERRIMACK-N.H.-M7142-\
(1) 017231 115 030467 031064 .ASCII  \M7142-DIRECT-CURSOR-ADDRESSING-TEST-DIGI\

```

(1)	017301	124	046101	042455	.ASCII	\TAL-EQUIPMENT-CORP.-MERRIMACK-N.H.-M7142-\
(1)	017352	033515	032061	026462	.ASCII	\M7142-DIRECT-CURSOR-ADDRESSING-TEST-DIGI\
(1)	017422	040524	026514	050505	.ASCII	\TAL-EQUIPMENT-CORP.-MERRIMACK-N.H.-M7142-\
(1)	017473	115	030467	031064	.ASCII	\M7142-DIRECT-CURSOR-ADDRESSING-TEST-DIGI\
(1)	017543	124	046101	042455	.ASCII	\TAL-EQUIPMENT-CORP.-MERRIMACK-N.H.-M7142-\
(1)	017614	033515	032061	026462	.ASCII	\M7142-DIRECT-CURSOR-ADDRESSING-TEST-DIGI\
(1)	017664	040524	026514	050505	.ASCII	\TAL-EQUIPMENT-CORP.-MERRIMACK-N.H.-M7142-\
(1)	017735	115	030467	031064	.ASCII	\M7142-DIRECT-CURSOR-ADDRESSING-TEST-DIGI\
(1)	020005	124	046101	042455	.ASCII	\TAL-EQUIPMENT-CORP.-MERRIMACK-N.H.-M7142-\
(1)	020056	033515	032061	026462	.ASCII	\M7142-DIRECT-CURSOR-ADDRESSING-TEST-DIGI\
(1)	020126	040524	026514	050505	.ASCII	\TAL-EQUIPMENT-CORP.-MERRIMACK-N.H.-M7142-\
(1)	020177	115	030467	031064	.ASCII	\M7142-DIRECT-CURSOR-ADDRESSING-TEST-DIGI\
(1)	020247	124	046101	042455	.ASCII	\TAL-EQUIPMENT-CORP.-MERRIMACK-N.H.-M7142-\
(1)	020320	033515	032061	026462	.ASCII	\M7142-DIRECT-CURSOR-ADDRESSING-TEST-DIGI\
(1)	020370	040524	026514	050505	.ASCII	\TAL-EQUIPMENT-CORP.-MERRIMACK-N.H.-M7142-\
(1)	020441	115	030467	031064	.ASCII	\M7142-DIRECT-CURSOR-ADDRESSING-TEST-DIGI\
(1)	020511	124	046101	042455	.ASCII	\TAL-EQUIPMENT-CORP.-MERRIMACK-N.H.-M7142-\
(1)	020562	033515	032061	026462	.ASCII	\M7142-DIRECT-CURSOR-ADDRESSING-TEST-DIGI\
(1)	020632	040524	026514	050505	.ASCII	\TAL-EQUIPMENT-CORP.-MERRIMACK-N.H.-M7142-\
(1)	020703	115	030467	031064	.ASCII	\M7142-DIRECT-CURSOR-ADDRESSING-TEST-DIGI\
(1)	020753	124	046101	042455	.ASCII	\TAL-EQUIPMENT-CORP.-MERRIMACK-N.H.-M7142-\
(1)	021024	033515	032061	026462	.ASCII	\M7142-DIRECT-CURSOR-ADDRESSING-TEST-DIGI\
(1)	021074	040524	026514	050505	.ASCII	\TAL-EQUIPMENT-CORP.-MERRIMACK-N.H.-M7142-\
(1)	021145	115	030467	031064	.ASCII	\M7142-DIRECT-CURSOR-ADDRESSING-TEST-DIGI\
(1)	021215	124	046101	042455	.ASCII	\TAL-EQUIPMENT-CORP.-MERRIMACK-N.H.-M7142-\
(1)	021266	033515	032061	026462	.ASCII	\M7142-DIRECT-CURSOR-ADDRESSING-TEST-DIGI\
(1)	021336	040524	026514	050505	.ASCII	\TAL-EQUIPMENT-CORP.-MERRIMACK-N.H.-M7142-\
(1)	021407	115	030467	031064	.ASCII	\M7142-DIRECT-CURSOR-ADDRESSING-TEST-DIGI\
(1)	021457	124	046101	042455	.ASCII	\TAL-EQUIPMENT-CORP.-MERRIMACK-N.H.-M7142-\
(1)	021530	033515	032061	026462	.ASCII	\M7142-DIRECT-CURSOR-ADDRESSING-TEST-DIGI\
(1)	021600	040524	026514	050505	.ASCII	\TAL-EQUIPMENT-CORP.-MERRIMACK-N.H.-M7142-\
(1)	021651	115	030467	031064	.ASCII	\M7142-DIRECT-CURSOR-ADDRESSING-TEST-DIGI\
(1)	021721	124	046101	042455	.ASCII	\TAL-EQUIPMENT-CORP.-MERRIMACK-N.H.-M7142-\
	021772	033515	032061	026462	.ASCII	\M7142-DIRECT-CURSOR-ADDRESSING-TEST-DIGI\
	022042	040524	026514	050505	.ASCII	\TAL-EQUIPMENT-CO\
	022062	377	377	377	MSGTND:	.BYTE 377,377,377,377
2268	022066	001762			.LIST	BEX
2269	000001				BUFFER:	.BLKW 1010.
					.END	

A	003460	1105#												
ADELAY	005266	824	903	945	1016	1045	1172	1215	1496#					
AMSG	005340	822	825	837	856	879	901	904	943	946	1014	1017	1019	1022
		1025	1028	1031	1034	1043	1046	1055	1068	1078	1105	1108	1113	1118
		1123	1128	1134	1213	1216	1219	1224	1229	1234	1239	1245	1248	1251
		1256	1259	1262	1266	1269	1275	1285	1315	1358	1374	1471	1514#	1548
		1590												
ANESC	005170	1445*	1447*	1464#										
ATTRS	011366	1472	1738#											
ATTR8	011242	1378	1738#											
AXOFF	005162	1461#												
A1	004062	1213#												
B	003502	1112#												
BEGIN	001274	595	717#											
BEGIN1	001340	597	726#											
BEGIN2	001360	598	730#											
BEGIN3	001330	599	724#											
BEGIN4	001350	600	728#											
BFCCHAR	004542	1302	1339#											
BFCMR	004504	1296	1302	1321#										
BIT0 =	000001	557#	1382	1608	1622									
BIT00 =	000001	547#	557											
BIT01 =	000002	546#	556											
BIT02 =	000004	545#	555											
BIT03 =	000010	544#	554											
BIT04 =	000020	543#	553											
BIT05 =	000040	542#	552											
BIT06 =	000100	541#	551											
BIT07 =	000200	540#	550											
BIT08 =	000400	539#	549	1897										
BIT09 =	001000	538#	548											
BIT1 =	000002	556#	1606											
BIT10 =	002000	537#	1100											
BIT11 =	004000	536#												
BIT12 =	010000	535#	1482											
BIT13 =	020000	534#	1929											
BIT14 =	040000	533#	1883											
BIT15 =	100000	532#												
BIT2 =	000004	555#	1603											
BIT3 =	000010	554#												
BIT4 =	000020	553#												
BIT5 =	000040	552#												
BIT6 =	000100	551#												
BIT7 =	000200	550#	1384	1605	1610	1626	1630							
BIT8 =	000400	549#												
BIT9 =	001000	548#	774	1865										
BLKPRT	005174	1021	1024	1027	1030	1033	1470#							
BPTVEC=	000014	564#												
BUFFER	022066	907	914	921	948*	949*	954	959*	970	989*	990*	991*	1406	1418
		1434	1728	2268#										
B1	004116	1223#												
C	003516	1117#												
CHRATT	004644	728	1374#	1397										
CR =	000015	472#	2052	2062										
CRLF =	000200	473#	2023	2062										
C1	004132	1228#												

MKB	007206	1106	1738#					
MKBA	007310	1549	1738#					
MKBB2	007375	1109	1220	1249	1260	1270	1738#	
MKBC	007463	1114	1225	1738#				
MKBD2	007524	1119	1230	1738#				
MKBE	007614	1124	1235	1738#				
MKBF	007705	1129	1240	1738#				
MKBG	007762	1246	1738#					
MKBGA	010031	1257	1738#					
MKBH	010101	1267	1738#					
MKBQ	010142	1591	1738#					
MKBQA	010207	1583*	1584*	1585*	1738#			
MKBQB	010226	1571*	1572*	1573*	1738#			
MKBQ1	010203	1587*	1738#					
MKBQ2	010222	1578*	1738#					
MKBR	010235	1135	1276	1738#				
MKBX	010350	1217	1738#					
MKBX1	010423	1738#						
MKB1	007253	1252	1263	1738#				
MKE	010460	1286	1738#					
MKEA	010665	1316	1738#					
MKEA1	010676	1303*	1304*	1309*	1310*	1312*	1313*	1738#
MKEB	010703	1290*	1291*	1292*	1738#			
MKEH	010712	1359	1738#					
MOVBS	002440	914#						
MOVAB	002414	907#						
MOVVT	002464	921#						
MQ0	010742	1738#						
MQ1	011027	1738#						
MQ2	011115	1738#						
MSCOPE	014532	769	1865#					
MSGTND	022062	2267#						
MSGTXT	016142	955	2267#					
MTAB	012540	908	1738#					
MTEXT0	013053	1137	1738#					
MTEXT1	013157	1138	1738#					
MTEXT2	013260	1139	1738#					
MTEXT3	013362	1140	1738#					
MTEXT4	013445	1141	1738#					
MTEXT5	013547	1142	1738#					
MTEXT6	013650	1143	1738#					
MTEXT7	013702	1144	1738#					
MTEXT8	014002	1145	1738#					
MT08	006412	1515	1727#					
MVT	013006	922	1738#					
M91	006516	838	1738#					
M9221	007043	1018	1738#					
M9222	007124	1375	1738#					
M93	006566	857	1738#					
M94	006632	880	1738#					
M96	006665	905	1738#					
M97	006746	1047	1738#					
M98	011136	947	1738#					
M99	007006	1079	1738#					
NOWEOP	004020	1196	1201#					
OCTAL	005360	1289	1523#	1570	1582			

[illegible]

[illegible]

[illegible]

CZVTOAO MACY11 30A(1052) 09-JAN-79 08:55 PAGE 58

E 5

CZVTOA.P11 09-JAN-79 08:54

CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0056

STRP = 000011	2199#	2208#	2209#	2210#	2211#	2212#	2214	2215#	2216#	2217#							
STRPAD 015740	2189	2206#															
STSTNM 001102	620#	1174*	1875	1899	1901*	1904	1907	1921	1924	1943							
STTYIN 014326	1783	1784	1801	1805#													
STYPBN= ***** U	2212																
STYPDS 006110	1648#	2211															
STYPE 015134	2009#	2199	2207														
STYPEC 015304	2030	2037	2044	2049#	2050												
STYPEX 015352	2055	2057	2060#														
STYPOC 015400	2093#	2208															
STYPON 015414	2092	2095#	2210														
STYPOS 015354	2088#	2209															
\$XTSTR 014560	1886#																
\$GET4= 000000	1189#																
\$OFILL 015577	2089*	2093*	2103	2138#													
\$4OCAT= ***** U	1883	1931															
.	587#	591#	605	606#	608#	610#	617#	655	738	750	1197	1201	1702#				
= 026032	1738#	1741	1805#	1806	1812	1865	1907	1943	1990#	2062	2233	2255	2268#				

.SERRT	1#	452#	1944
.SMULT	1#		
.SPARM	452#		
.SPOWE	1#	452#	2217
.SRAND	1#	452#	2140
.SRDDE	1#		
.SRDOC	1#	452#	1812
.SREAD	1#	452#	1738
.SR2AZ	1#		
.SSAVE	1#	452#	
.SSB2D	1#		
.SSB2O	1#		
.SSCOP	1#	452#	1870
.SSIZE	1#		
.SSPAC	452#		
.SSUPR	1#		
.SSWDO	452#		
.STRAP	1#	452#	2176
.STYPB	1#		
.STYPD	1#	452#	1636
.STYPE	1#	452#	1992
.STYPO	1#	452#	2063
.S4OCA	1#		
.1170	1#		

. ABS. 026032 000

ERRORS DETECTED: 0

CZVTOA/1,CZVTOA.SEQ/CRF/SOL=CZVTOA.SML,CZVTOA.P11
RUN-TIME: 14 11 .8 SECONDS
RUN-TIME RATIO: 30/26=1.1
CORE USED: 32K (63 PAGES)