

IP11/IP300

PROCESS CONTROL SUBSYSTEMS
MD-11-DVPCA-A

EP-DVPCA-A-DL-A
COPYRIGHT © 1977
FICHE 1 OF 1

DEC 1977
digital
MADE IN USA

The main body of the document consists of a grid of 48 small, faint diagrams or data tables arranged in 8 rows and 6 columns. Each cell contains a small schematic or data set, likely related to process control subsystems. The diagrams are too faint to read clearly but appear to be organized into a structured layout.

IDENTIFICATION

SEQ 0001

PRODUCT CODE MAINDEC-11-DVPCA-A-D
PRODUCT NAME PROCESS CONTROL SUBSYSTEMS - FIELD TEST
PRODUCT DATE NOVEMBER 1977
MAINTAINER DIAGNOSTIC ENGINEERING

The information in this document is subject to change without notice and should not be construed as a commitment by Digital Equipment Corporation. Digital Equipment Corporation assumes no responsibility for any errors that may appear in this manual.

The software described in this document is furnished to the purchaser under a license for use on a single computer system and can be copied (with inclusion of Digital's copyright notice) only for use in such system, except as may otherwise be provided in writing by Digital.

Digital Equipment Corporation assumes no responsibility for the use or reliability of its software on equipment that is not supplied by Digital.

Copyright (C) 1977 Digital Equipment Corporation

| | |
|------|--|
| 1 | DOCUMENT |
| 340 | TRAP CATCHER |
| 349 | STARTING ADDRESS(ES) |
| 360 | BASIC DEFINITIONS |
| 470 | BIT DEFINITIONS |
| 481 | COMMON TAGS |
| 602 | ERROR POINTER TABLE |
| 702 | INITIALIZE THE COMMON TAGS |
| 736 | DIAGNOSTICS MONITOR |
| 737 | TYPE PROGRAM NAME |
| 818 | DIGITAL MODULE MONITOR |
| 880 | SET SOFTWARE SWITCH REGISTER |
| 888 | AUTO TEST MONITOR |
| 981 | END OF PASS ROUTINE |
| 1008 | LOOP TEST |
| 1063 | MODULE TEST |
| 1112 | MAP OF DBUS |
| 1172 | IOCM TEST |
| 1426 | DIGITAL MODUL TEST ,M5010 |
| 1453 | DIGITAL MODULE TEST ,M5013 |
| 1541 | DIGITAL MODULE TEST ,M6010 |
| 1589 | DIGITAL MODULE TEST , M6012,M6013 |
| 1632 | DIGITAL MODULE TEST ,M6011 |
| 1692 | DIGITAL MODULE TEST ,M5011 |
| 1804 | DIGITAL MODULE TEST ,M5012 |
| 1924 | SUBROUTINES |
| 2231 | SCOPE HANDLER ROUTINE |
| 2287 | TTY INPUT ROUTINE |
| 2361 | READ AN OCTAL NUMBER FROM THE TTY |
| 2414 | ERROR HANDLER ROUTINE |
| 2455 | ERROR MESSAGE TYPEOUT ROUTINE |
| 2502 | TYPE ROUTINE |
| 2572 | CONVERT BINARY TO DECIMAL AND TYPE ROUTINE |
| 2639 | BINARY TO OCTAL (ASCII) AND TYPE |
| 2716 | SAVE AND RESTORE R0-R5 ROUTINES |
| 2761 | TRAP DECODER |
| 2784 | TRAP TABLE |
| 2804 | POWER DOWN AND UP ROUTINES |

1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56

SBTTL DOCUMENT

REM -
1 0 ABSTRACT

THIS PROGRAM IS A DIAGNOSTIC TOOL FOR TESTING THE ENTIRE FAMILY OF PCS MODULES. THE PROGRAM IS INTENDED TO BE USE BY FIELD ENGINEERING AND MANUFACTURING BY USING THIS PROGRAM AN OPERATOR IS ABLE TO CHECK THE IOCM, AND ANY DIGITAL OR ANALOG MODULE. SINCE THE ASSUMPTION IS MADE THAT ALL OUTPUT MODULES ARE CONNECTED TO THE CUSTOMERS WIRING DURING THE TEST OF OUTPUT MODULES THE DISABLE BIT IS SET. THIS DIAGNOSTIC WILL NOT CHECK THE PART OF HARDWARE THAT IS INTERFACING DIGITAL OR ANALOG MODULES WITH CUSTOMERS WIRING (DRIVING TRANSISTOR, ISOLATING TRANSFORMERS, PHOTO-COUPPLERS ECT) THIS DIAGNOSTIC DOES NOT REQUIRE ANY EXTERNAL DEVICES OR SPECIAL CONNECTIONS

2 0 HARDWARE REQUIREMENTS

1. LSI 11 WITH 16K OF MEMORY OR ANY PDP11 CPU WITH UNIBUS BRIDGE INTERFACE
- 2 CONSOLE TERMINAL
- 3 FLOPPY DISC OR SOME OTHER INPUT DEVICE
- 4 IOCM (M7458)

3 0 PROGRAM CONSIDERATION

3 1 CPU COMPATIBILITY
THIS PROGRAM CAN BE USED BY THE LSI 11 OR BY THE PDP-11 /04,05,10,20,34,35,40,45,50 & 70 IF UNIBUS BRIDGE MODULE IS USED

3 2 XXDP
THIS PROGRAM CAN BE CHAINED BY XXDP & WILL NOT OVERLAY THE LOADER

CHAIN MODE OPERATION

- 1 THE INPUT DIALOGUE WITH AN OPERATOR IS BYPASSED
- 2 THE SUBSYSTEM IS MAPPED IN MEMORY
- 3 THE SYSTEM TEST IS AUTOMATICALLY EXECUTED

NORMAL OPERATION

- 1 START AT LOC 204
2. SELECT TEST
- 3 PROGRAM RUNS AND GOES BACK TO MONITOR

3 3 ACT/APT

THIS PROGRAM IS ACT COMPATIBLE TO THE EXTENT THAT APT HOOKS

57 WILL BE IN THE PROGRAM & WILL WORK THRU THE "UPTON INTERFACE"

58

59

60

61

3 4 EXECUTION TIME

62

63

64

65

66

67

68

69

70

71

72

73

74

75

76

77

78

79

80

81

82

83

84

85

86

87

88

89

90

91

92

93

94

95

96

97

98

99

100

101

102

103

104

105

106

107

108

109

110

111

112

MOST INDIVIDUAL TESTS TAKE LESS THEN 1 SEC THE ONESHOT MODULE TEST TAKES 15 SEC. THE SYSTEM TEST EXECUTION TIME DEPENDS ON THE NUMBER AND TYPE OF I/O MODULES

3 5 DEFAULT ADDRESSES & VECTORS

THE FOLLOWING IS A LIST OF THE DEFAULT ADDRESSES & VECTORS OF ALL HARDWARE TO BE USED AND THEIR COPRESPONDING MEMORY LOCATIONS IN THE CASE THAT THE IOCM IS SET TO AN ADDRESS OTHER THAN 171000 THESE LOCATIONS MUST BE CHANGED

| | | | |
|---------|------|--------|-------------------------------------|
| BASE | WORD | 171000 | , FIRST ADDRESS OF I/OADDRESS BLOCK |
| CSR | WORD | 171377 | , ADDRESS OF IOCM CSR REGISTER |
| IAR | WORD | 171376 | , " IAR REGISTER |
| VECTO. | WORD | 234 | , INTERRUPT VECTOR OF IOCM |
| VECTOR. | WORD | 236 | , INTERRUPT VECTOR+2 OF IOCM |

4 0 OPERATING PROCEDURE

4 1 PROGRAM LOADING

THE PROGRAM CAN BE LOADED FROM PAPER TAPE OR FLOPPY DISK USING STANDARD PROCEDURES

4 2 PROGRAM STARTING

LOCATION 200 - STARTING ADDRESS TO RUN THE DIAGNOSTICS MONITOR THE PROGRAM WILL PRINT OPTIONS AVAILABLE TO THE OPERATOR

LOCATION 204 - STARTING ADDRESS OF MONITOR WITHOUT PRINTING OPERATOR'S OPTIONS

4 3 INPUT DIALOGUE

IF AN OPERATOR STARTS THE PROGRAM AT LOCATION 204 HE CAN SELECT ONE OF FOLLOWING OPTIONS

4 3 1 S-SYSTEM TEST

THIS OPTION OF THE PROGRAM WILL MAP ALL THE ANALOG AND DIGITAL MODULES CONNECTED TO THE IOCM AND BUILD A TABLE OF THEM IN MEMORY. THEN IT WILL PICK UP EACH ONE AND RUN THE APPROPRIATE TESTS IF SWITCH 14 IN THE SOFTWARE SWITCH REGISTER IS SET, IT WILL LOOP ON CURRENT TEST UNTIL OPERATOR TYPES CONTROL C AFTER THE SELECTED # OF PASSES, THE PROGRAM WILL PRINT PASS COUNT UNLESS SWITCH 13 IS SET

4 3 2 M - MAP OF DBUS

113
114 THIS OPTION ALLOWS AN OPERATOR TO CHECK WHICH I/O MODULE
115 ARE CONNECTED TO THE IOCM. IT CHECKS ALL ADDRESSES BETWEEN
116 171000 AND 171375. FOR ADDRESSES THAT ANSWER, IT CHECKS THE
117 GENERIC CODE AND TYPES THE ADDRESS AND INTERFACE TYPE
118

119 4 3 3 D - TEST DIGITAL MODULE

120
121 THIS TEST WILL EXERCISE THE DIGITAL MODULE AT THE ADDRESS
122 SPECIFIED BY THE OPERATOR. FOLLOWING IS A DESCRIPTION OF
123 THE DIGITAL MODULE TESTS FOR EACH MODULE TYPE
124 EACH TEST IS RUN AS MANY TIMES AS SELECTED BY THE L OPTION
125 AND THE PASS COUNT IS PRINTED

126 M5010

- 127 1 SET D BIT
- 128 2 CHECK THAT ALL BITS ARE ZERO
- 129 3 SET T BIT
- 130 4 CHECK THAT ALL BITS ARE ONES
- 131 5 SET C BIT

132 M5011

- 133 1 SET DBIT AND TBIT
- 134 2 CHECK IF INPUTS ARE ALL ONES
- 135 3 CLEAR TBIT
- 136 4 CHECK IF INPUTS ARE ALL ZEROS
- 137 5 CLEAR ALL INTERRUPTS
- 138 6 SET TBIT
- 139 7 ENABLE INTERRUPT
- 140 8 CHECK IF INTERRUPT OCCURRED
- 141 9 CHECK IF INPUTS ARE ALL ONES
- 142 10 CHECK IF COS REGISTERS ARE ALL ONES
- 143 11 SET RIF BIT
- 144 12 CLEAR ALL COS REGISTERS
- 145 13 CHECK IF THEY ARE CLEAR
- 146 14 CLEAR T BIT
- 147 15 CHECK IF INTERRUPT OCCURED
- 148 16 CHECK IF ALL INPUTS ARE ZERO
- 149 17 RU, STEP 10 TO 13
- 150 18 CLEAR ALL INTERRUPTS

151 M5012

- 152 1 SET DBIT
- 153 2 CHECK IF ALL BITS ARE ZERO
- 154 3 SET TBIT
- 155 4 CHECK IF ALL BITS ARE ONE
- 156 5 CLEAR T BIT
- 157 6 CLEAR ALL INTERRUPTS
- 158 7 SET T BIT
- 159 8 ENABLE INTERRUPT
- 160 9 CHECK IF INTERRUPT OCCUPRED

161
162
163
164
165
166
167
168

169 10 SET RIF BIT
170 11. CLEAR INTERRUPT
171 12 CHECK IF INTERRUPTS ARE CLEAR
172 13 CLEAR T BIT
173 14. RUN STEP 9 TO 12
174 15 CLEAR ALL INTERRUPTS

5013

180 1 SET DBIT
181 2 CHECK IF ALL BITS ARE ZERO
182 3 SET TBIT
183 4 TEST IF ALL BITS ARE ONE
184 5 CLEAR T BIT
185 6 CLEAR ALL INTERRUPTS
186 7 SET T BIT
187 8 ENABLE INTERRUPT
188 9 CHECK IF INTERRUPT OCCURED
189 10 SET RIF BIT
190 11 CLEAR INTERRUPT
191 12 CHECK IF INTERRUPT IS CLEAR
192 13 CLEAR T BIT
193 14 RUN STEP 9 TO 12
194 15 CLEAR ALL INTERRUPTS

M6010:

195 1 SET D BIT
196 2 CLEAR ALL 4 BYTES OF I/O REGISTER
197 3 SET DATA PATTERN 125 IN FIRST BYTE
198 4 CHECK IF IT IS SET
199 5 CHECK IF OTHER BYTES ARE ZERO
200 6 DO THE SAME WITH DATA 252 , 377 , 000
201 7 DO THE SAME WITH OTHER BYTES

M6011

THE LINE CLOCK MUST BE ENABLE

202 1 SET D BIT
203 2 SET DATA PATTERN 252 AT MUT
204 3 CHECK IF IT IS SET
205 4 WAIT UP TO 10 SEC
206 5 DATA SHOULD BE CLEAR
207 6 REPEAT STEP 2 TO 5 WITH DATA EQUAL TO 125

M6012
M6013

208 1 SET D BIT
209 2 SET OUTPUT REGISTER TO FOLLOWING DATA PATTERN
210 0, 377 252, 125
211 3 CHECK IF IT IS SET
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
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

4 3 4 I - TEST IOCM

THIS TEST WILL EXERCISE ALL THE FEATURES OF THE IOCM THE FOLLOWING
IS A DESCRIPTION OF ALL THE TESTS
THIS TEST IS RUN AS MANY TIMES AS SELECTED AND THE PASS COUNT IS TYPED

- TST1 CHECKS IF EACH BIT OF THE IOCM IS CLEARED BY THE CBIT
- TST2 CHECKS IF THE INTERRUPT ENABLE BIT (EBIT) CAN BE SET AND CLEARED
- TST3 CHECKS IF THE MAINTENANCE BIT (MBIT) CAN BE SET AND CLEARED
- TST4 CHECKS IF THE DISABLE BIT (DBIT) CAN BE SET AND CLEARED
CHECKS IF DBIT GENERATES CLEAR
- TST5 CHECKS IF THE TEST BIT (TBIT) CAN BE SET AND CLEARED
- TST6 CHECKS IF THE GENERIC CODE BIT (GBIT) CAN BE SET AND CLEARED
- TST7 CHECKS IF THE RIF BIT (RBIT) CAN BE SET AND CLEARED
- TST10 CHECKS DBUS DATA PATHS IN A MAINTENANCE
MODE IF THE MBIT IS SET AND THE CPU ADDRESSES ANY LOCATION
BETWEEN 171000 AND 171375 IT SHOULD READ BACK THE LOWER BYTE
OF THE MODULE ADDRESS
- TST11 CHECKS MAINTENANCE INTERRUPT IF THE MBIT & EBIT
ARE SET, THE IOCM WILL GENERATE AN INTERRUPT WITH VECTOR 234
THE IAR (171376) HAS THE LOWER BYTE OF CSR ADDRESS (377)

4 3 5 W -WRAP AROUND TEST

THIS OPTION ALLOWS THE FIELD ENGINEER TO CONNECT MODULE M6010 TO
EITHER M5010 OR M5011 THEN IT WRAPS AROUND A SET OF DATA PATTERNS(125, 252, 377, 0
IF SWITCH 14 IN THE SOFTWARE SWITCH REGISTER IS SET, IT WILL
LOOP ON THIS TEST UNTIL THE OPERATOR TYPES CONTROL C. EVERY SELECTED # OF
PASSES IT WILL PRINT PASS COUNT UNLESS SWITCH 13 IS SET
ERRORS WILL BE PRINTED INDICATING GOOD DATA, BAD DATA AND THE PASS # AT WHICH
IT OCCURRED

4 3 6 F -FIELD TEST

THE PURPOSE OF THIS TEST IS
TO OUTPUT ANY SELECTED DATA PATTERN TO AN OUTPUT
MODULE SPECIFIED BY OPERATOR
B TO MONITOR DATA FROM AN INPUT MODULE SPECIFIED BY OPERATOR

WARNING

THE FIELD ENGINEER MUST REALIZE THAT THIS IS THE ONLY TEST
THAT PERMITS HIM TO OUTPUT DATA TO THE CUSTOMER'S WIRING
THERE WILL BE VOLTAGE APPLIED TO CUSTOMER EQUIPMENT CONNECTED
TO THE OUTPUT MODULE UNDER TEST
THEREFORE PRECAUTIONS MUST BE TAKEN THAT AN ERRONEOUS OUTPUT WILL
NOT CAUSE DAMAGE TO THE FIELD EQUIPMENT AND THAT ALL TESTING IS
CONDUCTED WITH THE CUSTOMERS KNOWLEDGE

281 PROCEDURE
282 USER SELECTS THE ADDRESS OF THE MODULE UNDER TEST.
283 IF IT IS AN OUTPUT MODULE THEN.
284
285 USER SELECTS AND OUTPUTS A DATA PATTERN, ONE BYTE AT A TIME,
286 TO THE MODULE. TYPE CONTROL/C TO ABORT TEST
287 AFTER ALL BYTES OF MODULE HAS BEEN SELECTED, USER MUST TYPE CONTROL/C
288 TO GO BACK TO MONITOR

289
290 EXEMPLE
291
292 TYPE ADDRESS OF MUT 171XXX
293 TYPE ONE BYTE DATA PATTERN YYY
294 (WHERE YYY IS A 8 BIT DATA PATTERN TO OUTPUT
295 ALSO, THE REQUEST FOR DATA FOR OUTPUT WILL BE REPEATED UP TO
296 4 TIMES DEPENDENT UPON MODULE TYPE)
297
298 IF IT IS AN INPUT MODULE THEN
299
300 THE TEST MONITORS AND PRINTS DATA FROM MODULE INPUTS
301 IT PRINTS EVERY BYTE ADDRESS OF THE MODULE WITH ITS
302 CONTENT DURING THE FIRST PASS AND IT CONTINUES TO
303 MONITOR THE DATA WITHOUT PRINTING IT UNLESS A CHANGE
304 IN THE DATA OCCURED MUST TYPE CONTROL/C TO RETURN TO MONITOR

305
306 EXEMPLE
307
308 TYPE ADDRESS OF MUT 171XXX
309 171XXX YYY
310 (WHERE YYY IS THE 8 BITS OF DATA READ FROM THE MODULE
311 THE SECOND LINE OF THE MESSAGE WILL BE REPEATED
312 UP TO 4 TIMES, WITH INCREMENTING ADDRESS,
313 DEPENDING UPON MODULE TYPE

314
315 4 3 7 L - SET PASS COUNT
316
317 TYPE THE OCTAL NUMBER OF ITERATIONS FOR ALL TESTS
318 DEFAULT NUMBER IS 1
319 MAXIMUM NUMBER IS 177777

320
321 4 3 7 T - SET SOFTWARE SWITCH REGISTER
322
323 TYPE THE OCTAL NUMBER TO BE LOADED TO SWREG
324
325 100000 - HALT ON ERROR
326 40000 - LOOP ON TEST
327 20000 - INHIBIT ERROR AND END OF PASS PRINT
328 10000 - LOOP ON ERROR

329
330 4 3 8 CONTROL C
331
332 TYPING CONTROL C CAUSES THE PRESENT TEST TO BE ABORTED
333 AND PROGRAM RETURNS TO THE MONITOR
334 IN SOME CASES IT MAY TAKE UP TO 20 SECONDS FOR THIS TO
335 HAPPEN
336

```

337
338
339
340      SBTTL  TRAP CATCHER
341
342      000000      =0
343      ,*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ". +2,HALT"
344      ,*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
345      ,*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
346      000174      =174
347 000174 000000  DISPREG: WORD 0      ; SOFTWARE DISPLAY REGISTER,
348 000176 000000  SWREG   WORD 0      ; SOFTWARE SWITCH REGISTER
349      SBTTL  STARTING ADDRESS(ES)
350 000200 000137 002362  JMP      @#START ; JUMP TO STARTING ADDRESS OF PROGRAM
351      =100
352 000100 012350  NOCLK
353      =102
354 000102 000340  #PR7
355      =42
356 000042 000000  HALT
357      =204
358 000204 000137 002640  JMP      SETCLK
359      =1100
360      SBTTL  BASIC DEFINITIONS
361
362      ,*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
363      001100  STACK= 1100
364      EQUIV  EMT,ERROR      ; BASIC DEFINITION OF ERROR CALL
365      EQUIV  IOT,SCOPE     ; BASIC DEFINITION OF SCOPE CALL
366
367      ,*MISCELLANEOUS DEFINITIONS
368      000011  HT= 11      ; CODE FOR HORIZONTAL TAB
369      000012  LF= 12      ; CODE FOR LINE FEED
370      000015  CR= 15      ; CODE FOR CARRIAGE RETURN
371      000200  CRLF= 200    ; CODE FOR CARRIAGE RETURN-LINE FEED
372      177776  PS= 177776  ; PROCESSOR STATUS WORD
373      EQUIV  PS,PSW
374      177774  STKLMT= 177774 ; STACK LIMIT REGISTER
375      177772  PIRQ= 177772 ; PROGRAM INTERRUPT REQUEST REGISTER
376      177570  DSWR= 177570 ; HARDWARE SWITCH REGISTER
377      177570  ODISP= 177570 ; HARDWARE DISPLAY REGISTER
378
379      ,*GENERAL PURPOSE REGISTER DEFINITIONS
380      000000  R0= %0      ; GENERAL REGISTER
381      000001  R1= %1      ; GENERAL REGISTER
382      000002  R2= %2      ; GENERAL REGISTER
383      000003  R3= %3      ; GENERAL REGISTER
384      000004  R4= %4      ; GENERAL REGISTER
385      000005  R5= %5      ; GENERAL REGISTER
386      000006  R6= %6      ; GENERAL REGISTER
387      000007  R7= %7      ; GENERAL REGISTER
388      000006  SP= %6      ; STACK POINTER
389      000007  PC= %7      ; PROGRAM COUNTER
390
391      ,*PRIORITY LEVEL DEFINITIONS
392      000000  PRO= 0      ; PRIORITY LEVEL 0
  
```

BASIC DEFINITIONS

| | | | | |
|-----|--------|------|-----|---------------------|
| 393 | 000040 | PR1= | 40 | :: PRIORITY LEVEL 1 |
| 394 | 000100 | PR2= | 100 | :: PRIORITY LEVEL 2 |
| 395 | 000140 | PR3= | 140 | :: PRIORITY LEVEL 3 |
| 396 | 000200 | PR4= | 200 | :: PRIORITY LEVEL 4 |
| 397 | 000240 | PR5= | 240 | :: PRIORITY LEVEL 5 |
| 398 | 000300 | PR6= | 300 | :: PRIORITY LEVEL 6 |
| 399 | 000340 | PR7= | 340 | :: PRIORITY LEVEL 7 |

; *"SWITCH REGISTER" SWITCH DEFINITIONS

| | | | | |
|-----|--------|-------|-----------|--|
| 401 | | SW15= | 100000 | |
| 402 | 100000 | SW14= | 40000 | |
| 403 | 040000 | SW13= | 20000 | |
| 404 | 020000 | SW12= | 10000 | |
| 405 | 010000 | SW11= | 4000 | |
| 406 | 004000 | SW10= | 2000 | |
| 407 | 002000 | SW09= | 1000 | |
| 408 | 001000 | SW08= | 400 | |
| 409 | 000400 | SW07= | 200 | |
| 410 | 000200 | SW06= | 100 | |
| 411 | 000100 | SW05= | 40 | |
| 412 | 000040 | SW04= | 20 | |
| 413 | 000020 | SW03= | 10 | |
| 414 | 000010 | SW02= | 4 | |
| 415 | 000004 | SW01= | 2 | |
| 416 | 000002 | SW00= | 1 | |
| 417 | 000001 | EQUIV | SW09, SW9 | |
| 418 | | EQUIV | SW08, SW8 | |
| 419 | | EQUIV | SW07, SW7 | |
| 420 | | EQUIV | SW06, SW6 | |
| 421 | | EQUIV | SW05, SW5 | |
| 422 | | EQUIV | SW04, SW4 | |
| 423 | | EQUIV | SW03, SW3 | |
| 424 | | EQUIV | SW02, SW2 | |
| 425 | | EQUIV | SW01, SW1 | |
| 426 | | EQUIV | SW00, SW0 | |
| 427 | | | | |
| 428 | | | | |

; *DATA BIT DEFINITIONS (BIT00 TO BIT15)

| | | | | |
|-----|--------|--------|-------------|--|
| 429 | | BIT15= | 100000 | |
| 430 | 100000 | BIT14= | 40000 | |
| 431 | 040000 | BIT13= | 20000 | |
| 432 | 020000 | BIT12= | 10000 | |
| 433 | 010000 | BIT11= | 4000 | |
| 434 | 004000 | BIT10= | 2000 | |
| 435 | 002000 | BIT09= | 1000 | |
| 436 | 001000 | BIT08= | 400 | |
| 437 | 000400 | BIT07= | 200 | |
| 438 | 000200 | BIT06= | 100 | |
| 439 | 000100 | BIT05= | 40 | |
| 440 | 000040 | BIT04= | 20 | |
| 441 | 000020 | BIT03= | 10 | |
| 442 | 000010 | BIT02= | 4 | |
| 443 | 000004 | BIT01= | 2 | |
| 444 | 000002 | BIT00= | 1 | |
| 445 | 000001 | EQUIV | BIT09, BIT9 | |
| 446 | | EQUIV | BIT08, BIT8 | |
| 447 | | EQUIV | BIT07, BIT7 | |
| 448 | | | | |

BASIC DEFINITIONS

```
449 .EQUIV BIT06,BIT6
450 .EQUIV BIT05,BIT5
451 .EQUIV BIT04,BIT4
452 .EQUIV BIT03,BIT3
453 .EQUIV BIT02,BIT2
454 .EQUIV BIT01,BIT1
455 .EQUIV BIT00,BIT0
456
457 ;*BASIC "CPU" TRAP VECTOR ADDRESSES
458 000004 ERRVEC= 4 ;,TIME OUT AND OTHER ERRORS
459 000010 RESVEC= 10 ;,RESERVED AND ILLEGAL INSTRUCTIONS
460 000014 TBITVEC=14 ;,"T" BIT
461 000014 TRTVEC= 14 ;,TRACE TRAP
462 000014 BPTVEC= 14 ;,BREAKPOINT TRAP (BPT)
463 000020 IOTVEC= 20 ;,INPUT/OUTPUT TRAP (IOT) **SCOPE**
464 000024 PWRVEC= 24 ;,POWER FAIL
465 000030 EMTVEC= 30 ;,EMULATOR TRAP (EMT) **ERROR**
466 000034 TRAPVEC=34 ;,"TRAP" TRAP
467 000060 TKVEC= 60 ;,TTY KEYBOARD VECTOR
468 000064 TPVEC= 64 ;,TTY PRINTER VECTOR
469 000240 PIRQVEC=240 ;,PROGRAM INTERRUPT REQUEST VECTOR
470 SBTTL BIT DEFINITIONS
471 000200 FBIT=BIT7 ;,FLAG BIT
472 000100 EBIT=BIT6 ;,INTERRUPT ENABLE
473 000040 MBIT=BIT5 ;,MAINTENANCE INTERRUPT
474 000020 DBIT=BIT4 ;,I/O DISABLE BIT
475 000010 TBIT=BIT3 ;,TEST BIT, INVERTS I/O BITS
476 000004 GBIT=BIT2 ;,GENERIC CODE ENABLE
477 000002 CBIT=BIT1 ;,CLEAR I/O
478 000001 RBIT=BIT0 ;,RIF BIT, CLEARS I/O INTERRUPT
479
480
```

```

481 . SBTTL COMMON TAGS
482
483 ; *****
484 ; *THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
485 ; *USED IN THE PROGRAM
486
487         001100          =1100
488 001100    $CMTAG:      . =1100          ; START OF COMMON TAGS
489 001100    $PASS      . WORD 0          ; CONTAINS PASS COUNT
490 001102    $TSTNM:    . BYTE 0          ; CONTAINS THE TEST NUMBER
491 001103    $ERFLG:    . BYTE 0          ; CONTAINS ERROR FLAG
492 001104    $ICNT:     . WORD 0          ; CONTAINS SUBTEST ITERATION COUNT
493 001106    $LPADR:    . WORD 0          ; CONTAINS SCOPE LOOP ADDRESS
494 001110    $LPERR:    . WORD 0          ; CONTAINS SCOPE RETURN FOR ERRORS
495 001112    $ERTTL     . WORD 0          ; CONTAINS TOTAL ERRORS DETECTED
496 001114    $ITEMB:    . BYTE 0          ; CONTAINS ITEM CONTROL BYTE
497 001115    $ERMAX:    . BYTE 1          ; CONTAINS MAX. ERRORS PER TEST
498 001116    $ERRPC:    . WORD 0          ; CONTAINS PC OF LAST ERROR INSTRUCTION
499 001120    $GDADR:    . WORD 0          ; CONTAINS ADDRESS OF 'GOOD' DATA
500 001122    $BDADR:    . WORD 0          ; CONTAINS ADDRESS OF 'BAD' DATA
501 001124    $GDDAT     . WORD 0          ; CONTAINS 'GOOD' DATA
502 001126    $BDDAT     . WORD 0          ; CONTAINS 'BAD' DATA
503 001130    .          . WORD 0          ; RESERVED--NOT TO BE USED
504 001132    .          . WORD 0
505 001134    $AUTOB     . BYTE 0          ; AUTOMATIC MODE INDICATOR
506 001135    $INTAG     . BYTE 0          ; INTERRUPT MODE INDICATOR
507 001136    .          . WORD 0
508 001140    $SWR       . WORD 0          ; ADDRESS OF SWITCH REGISTER
509 001142    $DISPLAY   . WORD 0          ; ADDRESS OF DISPLAY REGISTER
510 001144    $TKS       . WORD 177560    ; TTY KBD STATUS
511 001146    $TKB       . WORD 177562    ; TTY KBD BUFFER
512 001150    $TPS       . WORD 177564    ; TTY PRINTER STATUS REG. ADDRESS
513 001152    $TPB       . WORD 177566    ; TTY PRINTER BUFFER REG. ADDRESS
514 001154    $NULL      . BYTE 0          ; CONTAINS NULL CHARACTER FOR FILLS
515 001155    $FILLS     . BYTE 2          ; CONTAINS # OF FILLER CHARACTERS REQUIRED
516 001156    $FILLC     . BYTE 12         ; INSERT FILL CHARS AFTER A "LINE FEED"
517 001157    $TPFLG     . BYTE 0          ; "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
518 001160    $REGAD     . WORD 0          ; CONTAINS THE ADDRESS FROM
519          .          . WORD 0          ; WHICH ($REGO) WAS OBTAINED
520 001162    $REGO      . WORD 0          ; CONTAINS (($REGAD)+0)
521 001164    $REG1      . WORD 0          ; CONTAINS (($REGAD)+2)
522 001166    $REG2      . WORD 0          ; CONTAINS (($REGAD)+4)
523 001170    $REG3      . WORD 0          ; CONTAINS (($REGAD)+6)
524 001172    $ESCAPE    . WORD 0          ; ESCAPE ON ERROR ADDRESS
525 001174    $QUES      . ASCII 77       ; QUESTION MARK
526 001175    $CRLF      . ASCII <15>    ; CARRIAGE RETURN
527 001176    $LF        . ASCII <12>    ; LINE FEED
528 ; *****
529 001200    $ATABL      . BLKW 80         ; TABLE A=ADDRESSES OF I/O MODULES
530 001440    $BTABL      . BLKW 80         ; TABLE B=GENERIC CODES OF THIS MODULES
531 001700    $GENER      . WORD 141        ; GENERIC CODE FOR NONISOLATED DC 32 BITS IN
532 001702    .          . WORD 121        ; GENERIC CODE FOR NONISOLATED DC 16 BITS IN
533 001704    .          . WORD 122        ; GENERIC CODE FOR ISOLATED DC 16 BITS IN
534 001706    .          . WORD 101        ; GENERIC CODE FOR AC 8 BITS IN
535 001710    .          . WORD 41         ; GENERIC CODE FOR NONISOLATED DC 32 BITS OUT
536 001712    .          . WORD 21         ; GENERIC CODE FOR ONESHOT 16 BITS OUT

```

| | | | | | |
|-----|--------|--------|------|--------|--|
| 537 | 001714 | 000001 | WORD | 1 | ; GENERIC CODE FOR ISOLATED DC 8 BITS OUT |
| 538 | 001716 | 000002 | WORD | 2 | ; GENERIC CODE FOR AC 8 BITS OUT |
| 539 | 001720 | 000000 | WORD | 0 | |
| 540 | 001722 | 005010 | WORD | 5010 | |
| 541 | 001724 | 005011 | WORD | 5011 | |
| 542 | 001726 | 005012 | WORD | 5012 | |
| 543 | 001730 | 005013 | WORD | 5013 | |
| 544 | 001732 | 006010 | WORD | 6010 | |
| 545 | 001734 | 006011 | WORD | 6011 | |
| 546 | 001736 | 006012 | WORD | 6012 | |
| 547 | 001740 | 006013 | WORD | 6013 | |
| 548 | 001742 | 006672 | WORD | M5010 | ; THIS TABLE HAS ADDRESSES OF I/O SUBROUTINE |
| 549 | 001744 | 010322 | WORD | M5011 | |
| 550 | 001746 | 011152 | WORD | M5012 | |
| 551 | 001750 | 006774 | WORD | M5013 | |
| 552 | 001752 | 007440 | WORD | M6010 | |
| 553 | 001754 | 010024 | WORD | M6011 | |
| 554 | 001756 | 007632 | WORD | M6012 | |
| 555 | 001760 | 007632 | WORD | M6013 | |
| 556 | 001762 | 000000 | WORD | 0 | |
| 557 | 001764 | 000000 | WORD | 0 | |
| 558 | 001766 | 000000 | WORD | 0 | |
| 559 | 001770 | 000000 | WORD | 0 | |
| 560 | 001772 | 000000 | WORD | 0 | |
| 561 | 001774 | 125 | BYTE | 125 | |
| 562 | 001775 | 252 | BYTE | 252 | |
| 563 | 001776 | 377 | BYTE | 377 | |
| 564 | 001777 | 000 | BYTE | 0 | |
| 565 | 002000 | 000042 | WORD | 42 | |
| 566 | 002002 | 000000 | WORD | 0 | |
| 567 | 002004 | 000000 | WORD | 0 | ; NUMBER OF BITES OF I/O |
| 568 | 002006 | 000000 | WORD | 0 | ; ADDRESS OF MUT |
| 569 | 002010 | 000000 | WORD | 0 | |
| 570 | 002012 | 000000 | WORD | 0 | |
| 571 | 002014 | 000000 | WORD | 0 | ; SECOND ADDRESS OF COS MODULE |
| 572 | 002016 | 000000 | WORD | 0 | ; DIGITAL MODULE UNDER TEST |
| 573 | 002020 | 000000 | WORD | 0 | |
| 574 | 002022 | 000000 | WORD | 0 | |
| 575 | 002024 | 177546 | WORD | 177546 | |
| 576 | 002026 | 171377 | WORD | 171377 | |
| 577 | 002030 | 171376 | WORD | 171376 | |
| 578 | 002032 | 000234 | WORD | 234 | |
| 579 | 002034 | 000236 | WORD | 236 | |
| 580 | 002036 | 000000 | WORD | 0 | |
| 581 | 002040 | 000000 | WORD | 0 | |
| 582 | 002042 | 171000 | WORD | 171000 | ; FIRST I/O |
| 583 | 002044 | 000000 | WORD | 0 | |
| 584 | 002046 | 000000 | WORD | 0 | |
| 585 | 002050 | 000000 | WORD | 0 | |
| 586 | 002052 | 000000 | WORD | 0 | |
| 587 | 002054 | 000000 | WORD | 0 | |
| 588 | 002056 | 000000 | WORD | 0 | |
| 589 | 002060 | 000001 | WORD | 1 | |
| 590 | 002062 | 000100 | WORD | 100 | |
| 591 | 002064 | 000102 | WORD | 102 | |
| 592 | 002066 | 011640 | WORD | F5010 | |

| | | | | |
|-----|--------|--------|------------|-------|
| 593 | 002070 | 011640 | . WORD | F5011 |
| 594 | 002072 | 011652 | WORD | F5012 |
| 595 | 002074 | 011664 | . WORD | F5013 |
| 596 | 002076 | 011676 | WORD | F6010 |
| 597 | 002100 | 011732 | WORD | F6011 |
| 598 | 002102 | 011716 | WORD | F6012 |
| 599 | 002104 | 011716 | . WORD | F6013 |
| 600 | 002106 | 000000 | WORD | 0 |
| 601 | 002110 | 000060 | KBVEC WORD | 60 |

ERROR POINTER TABLE

SEQ 0015

602 SBTTL ERROR POINTER TABLE
 603
 604 *THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
 605 *THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
 606 *LOCATION \$ITEMB THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
 607 *NOTE1 IF \$ITEMB IS 0 THE ONLY PERTINENT DATA IS (\$ERRPC).
 608 *NOTE2 EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS
 609
 610 * EM ;, POINTS TO THE ERROR MESSAGE
 611 * DH ;, POINTS TO THE DATA HEADER
 612 * DT ;, POINTS TO THE DATA
 613 * DF ;, POINTS TO THE DATA FORMAT
 614
 615

| Line | Item | Pointer | Pointer |
|------|--------|---------|---------|
| 616 | 002112 | | \$ERRTB |
| 617 | 002112 | 016230 | EM1 |
| 618 | 002114 | 016340 | DH1 |
| 619 | 002116 | 022024 | DT1 |
| 620 | 002120 | 000000 | 0 |
| 621 | 002122 | 016257 | EM2 |
| 622 | 002124 | 016340 | DH1 |
| 623 | 002126 | 022024 | DT1 |
| 624 | 002130 | 000000 | 0 |
| 625 | 002132 | 016371 | EM3 |
| 626 | 002134 | 000000 | 0 |
| 627 | 002136 | 000000 | 0 |
| 628 | 002140 | 000000 | 0 |
| 629 | 002142 | 016422 | EM4 |
| 630 | 002144 | 016340 | DH1 |
| 631 | 002146 | 022024 | DT1 |
| 632 | 002150 | 000000 | 0 |
| 633 | 002152 | 016450 | EM5 |
| 634 | 002154 | 016470 | DH5 |
| 635 | 002156 | 022040 | DT5 |
| 636 | 002160 | 000000 | 0 |
| 637 | 002162 | 016523 | EM6 |
| 638 | 002164 | 016340 | DH1 |
| 639 | 002166 | 022024 | DT1 |
| 640 | 002170 | 000000 | 0 |
| 641 | 002172 | 016572 | EM7 |
| 642 | 002174 | 016607 | DH7 |
| 643 | 002176 | 022004 | DT7 |
| 644 | 002200 | 000000 | 0 |
| 645 | 002202 | 016657 | EM10 |
| 646 | 002204 | 000000 | 0 |
| 647 | 002206 | 000000 | 0 |
| 648 | 002210 | 000000 | 0 |
| 649 | 002212 | 016712 | EM11 |
| 650 | 002214 | 016607 | DH7 |
| 651 | 002216 | 022004 | DT7 |
| 652 | 002220 | 000000 | 0 |
| 653 | 002222 | 016750 | EM12 |
| 654 | 002224 | 016470 | DH5 |
| 655 | 002226 | 022040 | DT5 |
| 656 | 002230 | 000000 | 0 |
| 657 | 002232 | 017000 | EM13 |

| | | | |
|-----|--------|--------|------|
| 658 | 002234 | 000000 | 0 |
| 659 | 002236 | 000000 | 0 |
| 660 | 002240 | 000000 | 0 |
| 661 | 002242 | 017046 | EM14 |
| 662 | 002244 | 000000 | 0 |
| 663 | 002246 | 000000 | 0 |
| 664 | 002250 | 000000 | 0 |
| 665 | 002252 | 017111 | EM15 |
| 666 | 002254 | 000000 | 0 |
| 667 | 002256 | 000000 | 0 |
| 668 | 002260 | 000000 | 0 |
| 669 | 002262 | 017154 | EM16 |
| 670 | 002264 | 000000 | 0 |
| 671 | 002266 | 000000 | 0 |
| 672 | 002270 | 000000 | 0 |
| 673 | 002272 | 016572 | EM17 |
| 674 | 002274 | 016607 | DH7 |
| 675 | 002276 | 021734 | DT17 |
| 676 | 002300 | 000000 | 0 |
| 677 | 002302 | 017217 | EM20 |
| 678 | 002304 | 016607 | DH7 |
| 679 | 002306 | 021754 | DT20 |
| 680 | 002310 | 000000 | 0 |
| 681 | 002312 | 017251 | EM21 |
| 682 | 002314 | 017313 | DH21 |
| 683 | 002316 | 022054 | DT21 |
| 684 | 002320 | 000000 | 0 |
| 685 | 002322 | 017337 | EM22 |
| 686 | 002324 | 000000 | 0 |
| 687 | 002326 | 000000 | 0 |
| 688 | 002330 | 000000 | 0 |
| 689 | 002332 | 017413 | EM23 |
| 690 | 002334 | 017450 | DH23 |
| 691 | 002336 | 021774 | DT23 |
| 692 | 002340 | 000000 | 0 |
| 693 | 002342 | 017541 | EM24 |
| 694 | 002344 | 000000 | 0 |
| 695 | 002346 | 000000 | 0 |
| 696 | 002350 | 000000 | 0 |
| 697 | 002352 | 020356 | EM25 |
| 698 | 002354 | 000000 | 0 |
| 699 | 002356 | 000000 | 0 |
| 700 | 002360 | 000000 | 0 |

MAIN MACY11 30(1046) 25-OCT-77 10 10 PAGE 15
 DVPLAR P11 25-OCT-77 10 02 ERROR POINTER TABLE

SEQ 0017

```

701 002362          START.
702          SBTTL  INITIALIZE THE COMMON TAGS
703          ;; CLEAR THE COMMON TAGS (%CMTAG) AREA
704 002362 012706 001100      MOV    #%CMTAG, R6      ;; FIRST LOCATION TO BE CLEARED
705 002366 005026          CLR    (R6)+          ;; CLEAR MEMORY LOCATION
706 002370 022706 001140      CMP    #SWR, R6      ;; DONE?
707 002374 001374          BNE    -6              ;; LOOP BACK IF NO
708 002376 012706 001100      MOV    #STACK, SP     ;; SETUP THE STACK POINTER
709          ;; INITIALIZE A FEW VECTORS
710 002402 012737 013670 000020  MOV    %%SCOPE, @%IOTVEC ;; IOT VECTOR FOR SCOPE ROUTINE
711 002410 012737 000340 000022  MOV    #340, @%IOTVEC+2 ;; LEVEL 7
712 002416 012737 014514 000030  MOV    #%ERROR, @%EMTVEC ;; EMT VECTOR FOR ERROR ROUTINE
713 002424 012737 000340 000032  MOV    #340, @%EMTVEC+2 ;; LEVEL 7
714 002432 012737 015770 000034  MOV    #%TRAP, @%TRAPVEC ;; TRAP VECTOR FOR TRAP CALLS
715 002440 012737 000340 000036  MOV    #340, @%TRAPVEC+2; LEVEL 7
716 002446 013737 004264 004256  MOV    %SENDCT, %SEOPCT ;; SETUP END-OF-PROGRAM COUNTER
717 002454 005037 001172          CLR    %ESCAPE        ;; CLEAR THE ESCAPE ON ERROR ADDRESS
718 002460 112737 000001 001115  MOVB  #1, %SERMAX     ;; ALLOW ONE ERROR PER TEST
719 002466 012737 002466 001110  MOV    # , %SLPERR    ;; SETUP THE ERROR LOOP ADDRESS
720          ;; SIZE FOR A HARDWARE SWITCH REGISTER IF NOT FOUND OR IT IS
721          ;; EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER
722 002474 013746 000004          MOV    @%ERRVEC, -(SP)  ;; SAVE ERROR VECTOR
723 002500 012737 002534 000004  MOV    #64$, @%ERRVEC  ;; SET UP ERROR VECTOR
724 002506 012737 177570 001140  MOV    #DSWR, SWR      ;; SETUP FOR A HARDWARE SWICH REGISTER
725 002514 012737 177570 001142  MOV    #DDISP, DISPLAY ;; AND A HARDWARE DISPLAY REGISTER
726 002522 022777 177777 176410  CMP    #-1, @SWR      ;; TRY TO REFERENCE HARDWARE SWR
727 002530 001012          BNE    66$          ;; BRANCH IF NO TIMEOUT TRAP OCCURRED
728          ;; AND THE HARDWARE SWR IS NOT = -1
729 002532 000403          BR     65$          ;; BRANCH IF NO TIMEOUT
730 002534 012716 002542          64$  MOV    #65$, (SP)      ;; SET UP FOR TRAP RETURN
731 002540 000002          RTI
732 002542 012737 000176 001140  65$  MOV    #SWREG, SWR     ;; POINT TO SOFTWARE SWR
733 002550 012737 000174 001142  MOV    #DISPREG, DISPLAY
734 002556 012637 000004          66$  MOV    (SP)+, @%ERRVEC ;; RESTORE ERROR VECTOR
735

```

DIAGNOSTICS MONITOR

```

736          SBTTL  DIAGNOSTICS MONITOR
737          SBTTL  TYPE PROGRAM NAME
738          ,,TYPE THE NAME OF THE PROGRAM IF FIRST PASS
739 002562 005227 177777          INC      #-1          ;;FIRST TIME?
740 002566 001024          BNE      67$          ;;BRANCH IF NO
741 002570 022737 004300 000042  CMP      #$ENDAD, @#42  ;;ACT-11?
742 002576 001420          BEQ      67$          ;;BRANCH IF YES
743 002600 104401 002606          TYPE     ,68$          ;;TYPE ASCIZ STRING
744 002604 000415          BR       67$          ;;GET OVER THE ASCIZ
745          ,,68$  ASCIZ  <CRLF>/DIGITAL IO SYSTEM TEST/<CRLF>
746 002640          67$
747 002640          SETCLK
748 002640 013746 000004          MOV      ERRVEC, -(SP)  ;;PUSH ERRVEC ON STACK
749 002644 012737 002662 000004  MOV      #1$, ERRVEC
750 002652 052777 000100 177144  BIS      #BIT6, @CLKADR  .ENABLE LINE CLOCK INTERRUPT FOR UNIBUS COMPUTERS
751 002660 000401          BR       2$
752 002662 022626          1$  CMP      (SP)+, (SP)+  .TRAP IF LS111
753 002664          2$
754 002664 012637 000004          MOV      (SP)+, ERRVEC  ;;POP STACK INTO ERRVEC
755 002670 012706 001100          MOV      #1100, R6
756 002674 005777 177100          TST      @XXDP
757 002700 001402          BEQ      4$
758 002702 000137 003612          JMP      AUTO
759 002706          4$
760 002706 012777 012550 177174  MOV      #KBINT, @KBVEC
761 002714 152777 000100 176222  BISB     #BIT6, @STKS  .ENABLE KB INTERRUPT
762 002722 012746 000000          MOV      #PRO, -(SP)  .SET PSW TO PRIORITY 0
763 002726 012746 002734          MOV      #64$, -(SP)
764 002732 000002          RTI
765 002734 000240          64$  NOP
766 002736 104401 017607          TYPE     ,M13          .TEST OPTIONS
767 002742 104401 017631          TYPE     ,M14          .S - SYSTEM TEST
768 002746 104401 017652          TYPE     ,M15          .D - DIGITAL MODULE TEST
769 002752 104401 017721          TYPE     ,M17          .M - MAP OF DBUS DEVICES
770 002756 104401 017774          TYPE     ,M19          .I - IOCM TEST
771 002762 104401 020022          TYPE     ,M21          .T - SET SWREG
772 002766 104401 020230          TYPE     ,M24          .F - FIELD TEST
773 002772 104401 020250          TYPE     ,M25          .W - WRAP AROUND TEST
774 002776 104401 020267          TYPE     ,M26          .L - LOAD LOOP COUNT
775

```

| | | | | | | | | |
|-----|--------|--------|--------|--------|-------|-------|--------------|---------------------------------|
| 776 | 003002 | 012706 | 001100 | | MONIT | MOV | #1100,R6 | ,SET STACK |
| 777 | 003006 | 142777 | 000100 | 176130 | | BICB | #BIT6,STKS | ,DISABLE KB INTERRUPT |
| 778 | 003014 | 104401 | 021114 | | | TYPE | ,MASS12 | ,TYPE TEST OPTION TO RUN |
| 779 | 003020 | 104406 | | | | RDCHR | | |
| 780 | 003022 | 012600 | | | | MOV | (SP)+,RO | ::POP STACK INTO RO |
| 781 | 003024 | 110037 | 021315 | | | MOVB | RO,MASS18 | |
| 782 | 003030 | 104401 | 021315 | | | TYPE | ,MASS18 | ,ECHO CHARACTER |
| 783 | 003034 | 022700 | 000123 | | | CMP | #123,RO | ,IS IT S? |
| 784 | 003040 | 001002 | | | | BNE | 1\$ | ,STAR AUTO TEST |
| 785 | 003042 | 000137 | 003612 | | | JMP | AUTO | |
| 786 | 003046 | 022700 | 000104 | 1\$ | | CMP | #104,RO | ,D ? |
| 787 | 003052 | 001002 | | | | BNE | 2\$ | |
| 788 | 003054 | 000137 | 003222 | | | JMP | DIGIT | ,START DIGITAL MODULE TEST |
| 789 | 003060 | 022700 | 000101 | 2\$ | | CMP | #101,RO | ,A ? |
| 790 | 003064 | 001002 | | | | BNE | 3\$ | |
| 791 | 003066 | 000137 | 011150 | | | JMP | ANALOG | ,START ANALOG MODULE TEST |
| 792 | 003072 | 022700 | 000115 | 3\$ | | CMP | #115,RO | ,M ? |
| 793 | 003076 | 001002 | | | | BNE | 4\$ | |
| 794 | 003100 | 000137 | 005054 | | | JMP | MAPE | ,MAP THE SYSTEM |
| 795 | 003104 | 022700 | 000130 | 4\$ | | CMP | #130,RO | ,X ? |
| 796 | 003110 | 001002 | | | | BNE | 5\$ | |
| 797 | 003112 | 000137 | 011634 | | | JMP | EXERC | ,START EXERCISER |
| 798 | 003116 | 022700 | 000111 | 5\$ | | CMP | #111,RO | ,I ? |
| 799 | 003122 | 001002 | | | | BNE | 6\$ | |
| 800 | 003124 | 000137 | 005302 | | | JMP | IOCM | ,TEST IOCM |
| 801 | 003130 | 022700 | 000124 | 6\$ | | CMP | #124,RO | ,T ? |
| 802 | 003134 | 001002 | | | | BNE | 7\$ | |
| 803 | 003136 | 000137 | 003564 | | | JMP | SWREGS | ,SET SOFTWARE SWITCH REG |
| 804 | 003142 | 022700 | 000106 | 7\$ | | CMP | #106,RO | ,F ? |
| 805 | 003146 | 001002 | | | | BNE | 8\$ | |
| 806 | 003150 | 000137 | 004610 | | | JMP | FIELD | ,START FIELD TEST |
| 807 | 003154 | 022700 | 000127 | 8\$ | | CMP | #127,RO | ,W ? |
| 808 | 003160 | 001002 | | | | BNE | 9\$ | |
| 809 | 003162 | 000137 | 004314 | | | JMP | LOOP | ,WRAP AROUND TEST |
| 810 | 003166 | 022700 | 000114 | 9\$ | | CMP | #114,RO | ,LOAD LOOP COUNT |
| 811 | 003172 | 001007 | | | | BNE | 10\$ | |
| 812 | 003174 | 104401 | 020321 | | | TYPE | ,M27 | ,NUMBER OF ITERATIONS = |
| 813 | 003200 | 104410 | | | | RDOCT | | |
| 814 | 003202 | 012637 | 002060 | | | MOV | (SP)+,PASCNT | ::POP STACK INTO PASCNT |
| 815 | 003206 | 000137 | 003002 | | | JMP | MONIT | |
| 816 | 003212 | 104401 | 020013 | 10\$ | | TYPE | ,M20 | ,? |
| 817 | 003216 | 000137 | 003002 | | | JMP | MONIT | ,OPERATOR TYPED WRONG CHARACTER |

| | | | | SBTTL | DIGITAL MODULE MONITOR | | | |
|-----|--------|--------|--------|--------|------------------------|---------------|-----------------------------------|--|
| | | | | DIGIT | | | | |
| 818 | | | | | | | | |
| 819 | 003222 | | | | MOV | #KBINT,@KBVEC | | |
| 820 | 003222 | 012777 | 012550 | 176660 | BISB | #BIT6,@STKS | ;ENABLE KB INTERRUPT | |
| 821 | 003230 | 152777 | 000100 | 175706 | TYPE | ,MASS13 | ;TYPE ADDRESS OF MUT | |
| 822 | 003236 | 104401 | 021144 | | RDOCT | | | |
| 823 | 003242 | 104410 | | | MOV | (SP)+,TADDR | ;POP STACK INTO TADDR | |
| 824 | 003244 | 012637 | 002006 | | MOV | ERRVEC,-(SP) | ;SAVE ERROR VECTOR | |
| 825 | 003250 | 013746 | 000004 | | MOV | #5\$,ERRVEC | ;SET LOC 4 | |
| 826 | 003254 | 012737 | 003550 | 000004 | TSTB | @CSR | ;MAKE SURE IOCM IS OK | |
| 827 | 003262 | 105777 | 176540 | | BEQ | 17\$ | | |
| 828 | 003266 | 001404 | | | ERROR | 10 | ;TEST ABORTED,IOCM ERROR | |
| 829 | 003270 | 104010 | | | MOV | (SP)+,ERRVEC | ;POP STACK INTO ERRVEC | |
| 830 | 003272 | 012637 | 000004 | | RTS | PC | | |
| 831 | 003276 | 000207 | | | MOVB | #GBIT,@CSR | ;SET GENERIC BIT | |
| 832 | 003300 | 112777 | 000004 | 176520 | MOVB | @TADDR,R0 | ;R0=GENERIC CODE OF MUT | |
| 833 | 003306 | 117700 | 176474 | | MOV | #MODUL,R1 | ;ADDRESS OF TEST | |
| 834 | 003312 | 012701 | 001742 | | MOV | #MUT,R3 | | |
| 835 | 003316 | 012703 | 001722 | | MOV | #GENER,R2 | ;TABLE OF GENERIC CODES | |
| 836 | 003322 | 012702 | 001700 | | CMR | R0,(R2)+ | | |
| 837 | 003326 | 020022 | | 3\$ | BNE | 1\$ | | |
| 838 | 003330 | 001067 | | | MOV | (R3),\$MUT | | |
| 839 | 003332 | 011337 | 002002 | | BISB | #CBIT,@CSR | ;SET C BIT | |
| 840 | 003336 | 152777 | 000002 | 176462 | CLR | YLOOP | ;WAIT | |
| 841 | 003344 | 005037 | 001766 | | INC | YLOOP | | |
| 842 | 003350 | 005237 | 001766 | 64\$ | CMR | YLOOP,#7 | | |
| 843 | 003354 | 023727 | 001766 | 000007 | BNE | 64\$ | | |
| 844 | 003362 | 001372 | | | BISB | #CBIT,@CSR | ;DO IT AGAIN | |
| 845 | 003364 | 152777 | 000002 | 176434 | CLR | YLOOP | ;WAIT | |
| 846 | 003372 | 005037 | 001766 | | INC | YLOOP | | |
| 847 | 003376 | 005237 | 001766 | 65\$ | CMR | YLOOP,#7 | | |
| 848 | 003402 | 023727 | 001766 | 000007 | BNE | 65\$ | | |
| 849 | 003410 | 001372 | | | MOV | (R1),DMUT | | |
| 850 | 003412 | 011137 | 002016 | | CLR | PASS | | |
| 851 | 003416 | 005037 | 002056 | 9\$ | JSR | PC,@DMUT | ;TEST MODULE | |
| 852 | 003422 | 004777 | 176370 | 4\$ | JSR | PC,CNTRC | ;CONTROL C | |
| 853 | 003426 | 004737 | 012446 | | INC | PASS | | |
| 854 | 003432 | 005237 | 002056 | 2\$ | CMR | PASCNT,PASS | | |
| 855 | 003436 | 023737 | 002060 | 002056 | BNE | 4\$ | | |
| 856 | 003444 | 001366 | | | BIT | #BIT13,SWREG | ;INHIBIT PRINT ? | |
| 857 | 003446 | 032737 | 020000 | 000176 | BNE | 7\$ | | |
| 858 | 003454 | 001005 | | | MOV | PASCNT,-(SP) | ;SAVE PASCNT FOR TYPEOUT | |
| 859 | 003456 | 013746 | 002060 | | TYPOC | | ;GO TYPE--OCTAL ASCII(ALL DIGITS) | |
| 860 | 003462 | 104402 | | | TYPE | ,M11 | ;# OF PASSES | |
| 861 | 003464 | 104401 | 017515 | | BIT | #BIT14,SWREG | ;LOOP ? | |
| 862 | 003470 | 032737 | 040000 | 000176 | BNE | 9\$ | | |
| 863 | 003476 | 001347 | | | TYPE | ,MASS15 | ;END OF MODULE TEST | |
| 864 | 003500 | 104401 | 021221 | | JMP | MONIT | | |
| 865 | 003504 | 000137 | 003002 | | ADD | #2,R1 | | |
| 866 | 003510 | 062701 | 000002 | 1\$ | ADD | #2,R3 | | |
| 867 | 003514 | 062703 | 000002 | | TST | (R2) | | |
| 868 | 003520 | 005712 | | | BNE | 3\$ | | |
| 869 | 003522 | 001301 | | | TYPE | ,MASS11 | ;UNKNOWN GENERIC CODE | |
| 870 | 003524 | 104401 | 021060 | | MOV | R0,-(SP) | ;SAVE R0 FOR TYPEOUT | |
| 871 | 003530 | 010046 | | | TYPOC | | ;GO TYPE--OCTAL ASCII(ALL DIGITS) | |
| 872 | 003532 | 104402 | | | TYPE | ,MASS0 | ;CR,LF | |
| 873 | 003534 | 104401 | 020417 | | | | | |

DIGITAL MODULE MONITOR

SEQ 0021

| | | | | | | | |
|-----|--------|--------|--------|----|-------|--------------|------------------------|
| 874 | 003540 | 012637 | 000004 | | MOV | (SP)+,ERRVEC | ,RESTORE ERRVEC |
| 875 | 003544 | 000137 | 003002 | | JMP | MONIT | |
| 876 | 003550 | 022626 | | 55 | CMP | (SP)+,(SP)+ | ,RESTORE STACK POINTER |
| 877 | 003552 | 104024 | | | ERROR | 24 | ,MODULE NOT RESPONDING |
| 878 | 003554 | 012637 | 000004 | | MOV | (SP)+,ERRVEC | |
| 879 | 003560 | 000137 | 003002 | | JMP | MONIT | |

SET SOFTWARE SWITCH REGISTER

SEQ 0022

| | | | | | | | |
|-----|--------|--------|--------|--------|------------------------------|-------------|------------------------|
| 880 | | | | SBTTL | SET SOFTWARE SWITCH REGISTER | | |
| 881 | | | | | | | |
| 882 | 003564 | 104401 | 020041 | SWREGS | TYPE | ,M22 | ,SWICHES OPTIONS |
| 883 | 003570 | 104401 | 020216 | | TYPE | ,M23 | ,SW REG = |
| 884 | 003574 | 104410 | | | RDOCT | | |
| 885 | 003576 | 012637 | 000176 | | MOV | (SP)+,SWREG | ,,POP STACK INTO SWPEG |
| 886 | 003602 | 104401 | 020417 | | TYPE | ,MASSO | ,CR,LF |
| 887 | 003606 | 000137 | 003002 | | JMP | MONIT | |

```

888          SBTTL  AUTO TEST MONITOR
889
890
891          , , *****
892          , THIS PART OF A PROGRAM IS A MONITOR FOR AUTOMATIC TEST
893          , FIRST IT TEST IOCM. THEN IT GENERATES A TABLE AND B TABLE
894          , THEN IT TESTS EACH INDIVIDUAL I/O MODULE CONNECTED TO DBUS
895          , REGISTERS RO & R1 SERVE AS THE POINTERS FOR MUT
896
897          , , *****
898 003612 005037 002056 AUTO CLR PASS
899 003616 012777 012550 176264 MOV #KBINT, @KBVEC
900 003624 152777 000100 175312 BISB #BIT6, @STKS , ENABLE KB INTERRUPT
901 003632 004737 005414 JSR PC, TIOCM
902 003636 105777 176164 TSTB @CSR , MAKE SURE IOCM IS OK
903 003642 001402 BEQ 7$
904 003644 104010 ERROR 10 , TEST ABORTED
905 003646 000207 RTS PC
906 003650 013746 000004 7$ MOV @#ERRVEC, -(SP) , SAVE ERROR VECTOR
907 003654 012737 003766 000004 MOV #1$, ERRVEC , SET ERROR VECTOR
908 003662 013700 002042 MOV BASE, RO , RO=171000
909 003666 112777 000004 176132 MOVB #GBIT, @CSR , SET GENERIC CODE ENABLE
910 003674 012702 001200 MOV #ATABL, R2
911 003700 012705 001440 MOV #BTABL, R5
912 003704 111001 5$ MOVB (RO), R1 , R1=GENERIC CODE, READ GENERIC CODE
913 003706 042701 177400 BIC #177400, R1
914 003712 004737 012200 JSR PC, TABLE , FIND WHAT I/O IT IS
915 003716 C90164 003722 JMP 3$(R4)
916 003722 000410 7$ BR 10$
917 003724 000407 BR 10$
918 003726 000410 BR 12$
919 003730 000410 BR 13$
920 003732 000404 BR 10$
921 003734 000405 BR 12$
922 003736 000405 BR 13$
923 003740 000404 BR 13$
924 003742 000403 BR 13$
925 003744 005200 10$ INC RO
926 003746 005200 11$ INC RO
927 003750 005200 12$ INC RO
928 003752 005200 13$ INC RO
929 003754 020037 002030 CMP RO, IAR
930 003760 001351 BNE 5$
931 003762 005012 CLR (R2)
932 003764 000406 BR 6$
933 003766 022626 1$ CMP (SP)+, (SP)+
934 003770 005200 INC R0
935 003772 020037 002030 CMP RO, IAR
936 003776 001342 BNE 5$
937 004000 005015 CLR (R5)
938 004002 012637 000004 6$ MOV (SP)+, @#ERRVEC , START TESTING INDIVIDUAL I/O
939 004006 142777 000004 176012 BICB #GBIT, @CSR , CLEAR G BIT
940 004014 004737 005414 25$ JSR PC, TIOCM
941 004020 005000 CLR R0
942 004022 005001 22$ CLR R1
943 004024 026061 001440 001700 23$ CMP BTABL(RO), GENER(R1)

```



```

944 004032 001416          BEQ      21$
945 004034 062701 000002    ADD      #2,R1
946 004040 005761 001700    TST     GENER(R1)
947 004044 001367          BNE     23$
948 004046 104401 021060    TYPE    ,MASS11          ;UNKNOWN GENERIC CODE
949 004052 016046 001440    MOV     BTABL(RO),-(SP) ;,SAVE BTABL(RO) FOR TYPEOUT
950 004056 104402          TYPOC   ;,GO TYPE--OCTAL ASCII(ALL DIGITS)
951 004060 104401 020417    TYPE    ,MASS0
952 004064 000137 003002    JMP     MONIT
953 004070 016137 001742 002016 21$  MOV     MODUL(R1),DMUT
954 004076 016037 001200 002006    MOV     ATABL(RO),TADDR
955 004104 016137 001722 002002    MOV     MUT(R1),$MUT
956 004112 010146          MOV     R1,-(SP)        ;,PUSH R1 ON STACK
957 004114 010046          MOV     RO,-(SP)        ;,PUSH RO ON STACK
958 004116 004777 175674    JSR     PC,@DMUT        ;,TEST THIS MODULE
959 004122 012600          MOV     (SP)+,RO        ;,POP STACK INTO RO
960 004124 012601          MOV     (SP)+,R1        ;,POP STACK INTO R1
961 004126 062700 000002    ADD     #2,RO
962 004132 005760 001440    TST     BTABL(RO)
963 004136 001331          BNE     22$
964 004140 005237 002056    INC     PASS            ;,INC PASS COUNT
965 004144 004737 012446    JSR     PC,CNTRC        ;,CONTROL C ?
966 004150 023737 002060 002056    CMP     PASCNT,PASS    ;,IS IT 1000 PASSES
967 004156 001316          BNE     25$
968 004160 005037 002056    CLR     PASS
969 004164 032737 020000 000176    BIT     #BIT13,SWREG    ;,INHIBIT PRINT
970 004172 001005          BNE     20$
971 004174 013746 002060    MOV     PASCNT,-(SP)    ;,SAVE PASCNT FOR TYPEOUT
972 004200 104402          TYPOC   ;,GO TYPE--OCTAL ASCII(ALL DIGITS)
973 004202 104401 017515    TYPE    ,M11           ;,# OF PASSES COMPLETED
974 004206 032737 040000 000176 20$  BIT     #BIT14,SWREG    ;,DO WE LOOP ON TESTS
975 004214 001277          BNE     25$
976 004216 005777 175556    TST     @XXDP
977 004222 001004          BNE     27$
978 004224 104401 021271 24$  TYPE    ,MASS17        ;,END OF SYSTEM TEST
979 004230 006137 003002    JMP     MONIT
980 004234          27$
981          SBTTL  END OF PASS ROUTINE
982
983          ;,*****
984          ;*INCREMENT THE PASS NUMBER ($PASS)
985          ;*IF THERES A MONITOR GO TO IT
986          ;*IF THERE ISN'T JUMP TO 200
987
988          SEOP
989          SCOPE
990 004234 000004          CLR     $TSTNM          ;,ZERO THE TEST NUMBER
991 004236 005037 001102    INC     $PASS           ;,INCREMENT THE PASS NUMBER
992 004242 005237 001100    BIC     #10000,$PASS    ;,DON'T ALLOW A NEG NUMBER
993 004246 042737 100000 001130    DEC     (PC)+          ;,LOOP?
994 004254 005327          SEOPCT  WORD  1
995 004256 000001          BGT     $DOAGN          ;,YES
996 004260 003013          MOV     (PC)+,@(PC)+    ;,RESTORE COUNTER
997 004262 012737          SENDCT  WORD  1
998 004264 000001          SEOPCT
999 004266 004256          SGET42  MOV     @#42,RO    ;,GET MONITOR ADDRESS
1000 004270 013700 000042

```

END OF PASS ROUTINE

SEQ 0025

| | | | | | | |
|------|--------|--------|---------|-------|---------|------------------------|
| 1000 | 004274 | 001405 | | BEQ | \$DOAGN | ..BRANCH IF NO MONITOR |
| 1001 | 004276 | 000005 | | RESET | | ..CLEAR THE WORLD |
| 1002 | 004300 | 004710 | SENDAD | JSR | PC.(R0) | ..GO TO MONITOR |
| 1003 | 004302 | 000240 | | NOP | | ..SAVE ROOM |
| 1004 | 004304 | 000240 | | NOP | | ..FOR |
| 1005 | 004306 | 000240 | | NOP | | ..ACT11 |
| 1006 | 004310 | | \$DOAGN | | | |
| 1007 | 004310 | 000137 | 000200 | JMP | @#200 | ..RETURN |

```

1008          SBTTL LOOP TEST
1009
1010          , THIS TEST ENABLE FIELD ENGINEER TO CONNECT OUTPUT MODULE TO INPUT MODULE
1011          ; OUTPUT TEST PATTERNS AND READ THEM BACK
1012          ,
1013          ,
1014 004314 104401 021515          LOOP TYPE , MASS23          , CONNECT OUTPUT TO INPUT MODULE
1015 004320 012777 012550 175562 MOV      #KBINT, @KBVEC
1016 004326 152777 000100 174610 BISB    #BIT6, @STKS          , ENABLE KB INTERRUPT
1017 004334 104401 021433          TYPE , MASS21          , TYPE ADDRESS OF OUTPUT MUT
1018 004340 104410          RDOCT
1019 004342 012637 002006          MOV      (SP)+, TADDR          , POP STACK INTO TADDR
1020 004346 104401 021463          TYPE , MASS22          , TYPE ADDRESS OF INPUT MUT
1021 004352 104410          RDOCT
1022 004354 012637 002050          MOV      (SP)+, TEMP1          , POP STACK INTO TEMP1
1023 004360 112777 000004 175440 MOVB   #GBIT, @CSR          , SET GENERIC BIT
1024 004366 117700 175414          MOVB   @TADDR, RO          , GENERIC CODE OF OUTPUT MUT
1025 004372 020027 000041          CMP     RO, #41          , IS IT M6010?
1026 004376 001047          BNE    1$
1027 004400 117701 175444          MOVB   @TEMP1, R1          , GENERIC CODE OF INPUT MUT
1028 004404 020127 000141          CMP     R1, #141          , IS IT M5010?
1029 004410 001403          BEQ    2$
1030 004412 020127 000121          CMP     R1, #121          , IS IT M5011?
1031 004416 001042          BNE    6$
1032 004420 005037 002056          2$    CLR     PASS
1033 004424 004737 012042          3$    JSR    PC, LOPTST          , TEST SUBROUTINE
1034 004430 004737 012360          JSR    PC, CLRINT          , CLEAR ALL INTERRUPTS
1035 004434 004737 012446          JSR    PC, CNTRC          , CONTROL-C?
1036 004440 005237 002056          INC     PASS
1037 004444 023737 002060 002056 CMP     PASCNT, PASS
1038 004452 001364          BNE    3$
1039 004454 032737 020000 000176 BIT     #BIT13, SWREG          , INHIBIT PRINTOUT
1040 004462 001005          BNE    5$
1041 004464 013746 002060          MOV     PASCNT, -(SP)          , SAVE PASCNT FOR TYPEOUT
1042 004470 104402          TYPOC          , GO TYPE--OCTAL ASCII (ALL DIGITS)
1043 004472 104401 017515          TYPE , M11          , PASSES COMPLETED
1044 004476 032737 040000 000176 5$ BIT     #BIT14, SWREG          , LOOP
1045 004504 001345          BNE    2$
1046 004506 104401 021556          TYPE , MASS25          , END OF LOOP TEST
1047 004512 000137 003002          JMP     MONIT
1048 004516 104401 021601          1$    TYPE , MASS26          , WRONG MODULE -OUTPUT MUST BE M6010
1049 004522 000402          BR     4$
1050 004524 104401 021650          6$    TYPE , MASS28          , WRONG MODULE- INPUT MUST BE EITHER M5010 OR M5011
1051 004530          4$
1052 004530 152777 000002 175270 BISB   #CBIT, @CSR          , SET C BIT
1053 004536 005037 001766          CLR     YLOOP          , WAIT
1054 004542 005237 001766          64$   INC     YLOOP
1055 004546 023727 001766 000007 CMP     YLOOP, #7
1056 004554 001372          BNE    64$
1057 004556 152777 000002 175242 BISB   #CBIT, @CSR          , DO IT AGAIN
1058 004564 005037 001766          CLR     YLOOP          , WAIT
1059 004570 005237 001766          65$   INC     YLOOP
1060 004574 023727 001766 000007 CMP     YLOOP, #7
1061 004602 001372          BNE    65$
1062 004604 000137 003002          JMP     MONIT
  
```

```

1063          .SETTL  MODULE TEST
1064
1065          , THIS TEST ENABLE FIELD ENGINEER TO SELECT AND OUTPUT ANY DATA
1066          , PATTERN TO OUTPUT MODULES, MONITOR AND PRINT DATA FROM INPUT MODULES
1067
1068
1069 004610 104401 021144          FIELD  TYPE  , MASS13          ; TYPE ADDRESS OF MUT
1070 004614 012777 012550 175266  MOV      #KBINT, @KBVEC
1071 004622 152777 000100 174314  BISB    #BIT6, @STKS          ; ENABLE KB INTERRUPT
1072 004630 104410
1073 004632 012637 002006          MOV      (SP)+, TADDR          ; , POP STACK INTO TADDR
1074 004636 013746 000004          MOV      ERRVEC, -(SP)        ; SAVE ERROR VECTOR
1075 004642 012737 005040 000004  MOV      #5$, ERRVEC          ; SET LOC 4
1076 004650 112777 000004 175150  MOVB    #GBIT, @CSR          ; SET GENERIC BIT
1077 004656 117700 175124          MOVB    @TADDR, R0           ; GENERIC CODE OF MUT
1078 004662 042700 177400          BIC     #177400, R0
1079 004666 012701 002066          MOV      #MOD, R1            ; ADDRESS OF TEST
1080 004672 012702 001700          MOV      #GENER, R2          ; TABLE OF GENERIC CODES
1081 004676 020022          3$     CMP      R0, (R2)+
1082 004700 001037          BNE     2$
1083 004702 152777 000002 175116  BISB    #CBIT, @CSR          ; SET C BIT
1084 004710 005037 001766          CLR     YLOOP                ; WAIT
1085 004714 005237 001766          64$   INC     YLOOP
1086 004720 023727 001766 000007  CMP     YLOOP, #7
1087 004726 001372          BNE     64$
1088 004730 152777 000002 175070  BISB    #CBIT, @CSR          ; DO IT AGAIN
1089 004736 005037 001766          CLR     YLOOP                ; WAIT
1090 004742 005237 001766          65$   INC     YLOOP
1091 004746 023727 001766 000007  CMP     YLOOP, #7
1092 004754 001372          BNE     65$
1093 004756 011137 002016          MOV     (R1), DMUT
1094 004762 004777 175030          JSR     PC, @DMUT            ; TEST MODULE
1095 004766 104401 021362          TYPE    , MASS20            ; TYPE CONTROL-C TO RETURN TO MONITOR
1096 004772 004737 012446          1$     JSR     PC, CNTPC          ; CONTROL-C ?
1097 004776 000775          BR      1$
1098 005000 062701 000002          2$     ADD     #2, R1
1099 005004 062703 000002          ADD     #2, R3
1100 005010 005712          TST    (R2)
1101 005012 001331          BNE     3$
1102 005014 104401 021060          TYPE    , MASS11            ; UNKNOWN GENERIC CODE
1103 005020 010046          MOV     RD, -(SP)            ; , SAVE RD FOR TYPEOUT
1104 005022 104402          TYPOC          ; , GO TYPE--OCTAL ASCII(ALL DIGITS)
1105 005024 104401 020417          TYPE    , MASS0             ; CR.LF
1106 005030 012637 000004          MOV     (SP)+, ERRVEC        ; RESTORE ERRVEC
1107 005034 000137 003002          JMP     MONIT
1108 005040 022626          5$     CMP     (SP)+, (SP)+      ; RESTORE STACK POINTER
1109 005042 104024          ERROR 24          ; MODULE NOT RESPONDING
1110 005044 012637 000004          MOV     (SP)+, ERRVEC
1111 005050 000137 003002          JMP     MONIT

```

```

1112 . SBTTL MAP OF DBUS
1113
1114 ,, *****
1115
1116 , THIS TEST WILL LIST ALL I/O INTERFACES CONNECTED TO IOCM
1117 , IT WILL ALSO LIST QBUS ADDRESSES
1118
1119 ,, *****
1120
1121
1122
1123
1124
1125 005054 013746 000004 MAPE MOV @#ERRVEC, -(SP) ;SAVE ERROR VECTOR
1126 005060 012737 005260 000004 MOV #1$, ERRVEC ;SET ERROR VECTOR
1127 005066 012777 012550 175014 MOV #KBINT, @KBVEC
1128 005074 152777 000100 174042 BLSB #BIT6, @STKS ;ENABLE KB INTERRUPT
1129 005102 013700 002042 MOV BASE, RO ;RO=171000
1130 005106 112777 000004 174712 MOVB #GBIT, @CSR ;SET GENERIC CODE ENABLE
1131 005114 111001 5S MOVB (RO), R1 ;R1=GENERIC CODE, READ GENERIC CODE
1132 005116 042701 177400 BIC #177400, R1
1133 005122 004737 012154 JSR PC, GENCOD ;FIND WHAT I/O IT IS
1134 005126 104401 020422 TYPE , MASS2 ;ADDRESS
1135 005132 010046 MOV RO, -(SP) ;SAVE RO FOR TYPEOUT
1136 005134 104402 TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
1137 005136 000164 00E142 JMP 3$(R4)
1138 005142 104401 020434 3$ TYPE , MASS3 ;M5010
1139 005146 000434 BR 10$
1140 005150 104401 020503 TYPE , MASS4 ;M5011
1141 005154 000431 BR 10$
1142 005156 104401 020552 TYPE , MASS5 ;M5012
1143 005162 000430 BR 12$
1144 005164 104401 020616 TYPE , MASS6 ;M5013
1145 005170 000426 BR 13$
1146 005172 104401 020650 TYPE , MASS7 ;M6010
1147 005176 000420 BR 10$
1148 005200 104401 020720 TYPE , MASS8 ;M6011
1149 005204 000417 BR 12$
1150 005206 104401 020761 TYPE , MASS9 ;M6012
1151 005212 000415 BR 13$
1152 005214 104401 021025 TYPE , MASS10 ;M6013
1153 005220 000412 BR 13$
1154 005222 104401 021060 TYPE , MASS11 ;UNKNOWN GENERIC CODE
1155 005226 010146 MOV R1, -(SP) ;SAVE R1 FOR TYPEOUT
1156 005230 104402 TYPOC ;GO TYPE--OCTAL ASCII(ALL DIGITS)
1157 005232 104401 020417 TYPE , MASS0 ;CR, LF
1158 005236 000403 BR 13$
1159 005240 005200 10$ INC RO ;INC ADDRESS BY APPROPRIATE NUMBER OF BYTES
1160 005242 005200 11$ INC RO
1161 005244 005200 12$ INC RO
1162 005246 005200 13$ INC RO
1163 005250 020037 002030 CMP RO, IAR ;LAST ADDRESS ?
1164 005254 001317 BNE 5$ ;NO, DO IT AGAIN
1165 005256 000405 BR 6$
1166 005260 022626 1$ CMP (SP)+, (SP)+ ;HERE IF ADDRESS IS NOT RESPONDING
1167 005262 005200 INC RO ;INC ADDRESS
  
```

MAIN MACY11 30(1046) 25-OCT-77 10 10 PAGE 27
DVPCAA P11 25-OCT-77 10 02 MAP OF DBUS

SEQ 0029

| | | | | | | | |
|------|--------|--------|--------|----|-----|-----------------|------------------------|
| 1168 | 005264 | 020037 | 002030 | | CMP | RD, IAR | , LAST ONE ? |
| 1169 | 005270 | 001311 | | | BNE | SS | |
| 1170 | 005272 | 012637 | 000004 | 65 | MOV | (SP)+, @#ERRVEC | , RESTORE ERROR VECTOR |
| 1171 | 005276 | 000137 | 003002 | | JMP | MONIT | |

1172
1173
1174
1175
1176
1177
1178
1179
1180

.SBTTL IOCM TEST

,,*****

, THIS PART OF DIAGNOSTIC TEST THE IOCM
, IT HAS 9 TESTS IT CHECKS IF ALL THE BITS OF THE IOCM
, CAN BE SET AND CLEAR , CHECKS MAINTENANCE
, INTERRUPT AND CHECKS ALL ADDRESSES IN MAINTENANCE MODE

```

1181                                     , , *****
1182 005302 005037 002056                IOCM CLR PASS
1183 005306 013746 000004                MOV ERRVEC, -(SP) ; ; PUSH ERRVEC ON STACK
1184 005312 012777 012550 174570         MOV #KBINT, @KBVEC
1185 005320 152777 000100 173616         BISB #BIT6, @STKS ; ; ENABLE KB INTERRUPT
1186 005326 004737 005414                JSR PC, TIOCM ; ; TEST IOCM
1187 005332 005237 002056                INC PASS
1188 005336 023737 002060 002056         CMP PASCNT, PASS ; ; 1000 PASSES DONE ?
1189 005344 001370                        BNE 1$
1190 005346 032737 020000 000176         BIT #BIT13, SWREG ; ; INHIBIT PRINT
1191 005354 001005                        BNE 2$
1192 005356 013746 002060                MOV PASCNT, -(SP) ; ; SAVE PASCNT FOR TYPEOUT
1193 005362 104402                        TYPOC ; ; GO TYPE--OCTAL ASCII(ALL DIGITS)
1194 005364 104401 017515                TYPE , M11 ; ; # OF PASSES
1195 005370 032737 040000 000176 2$     BIT #BIT14, SWREG ; ; LOOP ON TEST?
1196 005376 001353                        BNE 1$
1197 005400 104401 021246                TYPE , MASS16 ; ; IOCM TEST PASSED
1198 005404 012637 000004                MOV (SP)+, ERRVEC ; ; POP STACK INTO ERRVEC

```

/

MAIN MACY11 30(1046) 25-OCT-77 10 10 PAGE 30
DUPLAR P11 25-OCT-77 10 02 IOCM TEST

G 3

SEQ 0032

1199 005410 000137 003002 JMP MONIT

```

1200                                     , THIS TEST CHECKS IF EACH BIT OF CSR
1201                                     , IS CLEAR BY CBIT
1202 005414 TIOCM
1203 , , *****
1204 005414 000004 TST1 SCOPE
1205 005416 112737 000001 002022 MOVB #1, TNUMB
1206 005424 012737 005540 000004 MOV #10$, ERRVEC
1207 005432 112777 000074 174366 MOVB #74, @CSR , SET ALL BITS
1208 005440 152777 000002 174360 BISB #CBIT, @CSR , SET C BIT
1209 005446 005037 001766 CLR YLOOP , WAIT
1210 005452 005237 001766 64$ INC YLOOP
1211 005456 023727 001766 000007 CMP YLOOP, #7
1212 005464 001372 BNE 64$
1213 005466 152777 000002 174332 BISB #CBIT, @CSR , DO IT AGAIN
1214 005474 005037 001766 CLR YLOOP , WAIT
1215 005500 005237 001766 65$ INC YLOOP
1216 005504 023727 001766 000007 CMP YLOOP, #7
1217 005512 001372 BNE 65$
1218 005514 117737 174306 001126 MOVB @CSR, $BDDAT , STORE CONTENTS OF CSR IN BDDAT
1219 005522 005037 001124 CLR $GDDAT
1220 005526 005737 001126 TST $BDDAT , IS CSP CLEAR
1221 005532 001405 BEQ 2$ , YES
1222 005534 104001 ERROR 1 , CBIT NOT CLEARING IOCM
1223 005536 000207 RTS PC , FATAL ERROR RETURN TO MONITOR
1224 005540 104003 10$ ERROR 3 , IOCM NOT RESPONDING
1225 005542 022626 CMP (SP)+, (SP)+ , PESTORE STACK
1226 005544 000207 RTS PC
1227 005546 2$

```

```

1228                                     , THIS TEST CHECKS E BIT
1229                                     , *****
1230 005546 000004 TST2 SCOPE
1231 005550 112737 000002 002022      MOVB #2, TNUMB
1232 005556 012737 000100 001124      MOV #EBIT, $GDDAT , SET TESTED BIT
1233 005564 113737 017512 016326      MOVB M7, EM2X , SET EBIT IN ERROR MESSAGE
1234 005572 004737 012230              JSR PC, BITSET
1235 005576 104002                      ERROR 2 , BIT IS NOT SETTING
1236 005600 005037 001124              CLR $GDDAT
1237 005604 004737 012230              JSR PC, BITSET
1238 005610 104002                      ERROR 2
1239
1240                                     , THIS TEST CHECKS MBIT
1241                                     , *****
1242 005612 000004 TST3 SCOPE
1243 005614 112737 000003 002022      MOVB #3, TNUMB
1244 005622 012737 000240 001124      MOV #MBIT|#FBIT, $GDDAT, SET MAITENANCE BIT
1245 005630 113737 017507 016326      MOVB M6, EM2X , SET ERROR MESSAGE
1246 005636 004737 012230              JSR PC, BITSET
1247 005642 104002                      ERROR 2 , BIT NOT SETTING
1248 005644 005037 001124              CLR $GDDAT
1249 005650 004737 012230              JSR PC, BITSET,
1250 005654 104002                      ERROR 2 , BIT NOT CLEARING
1251
1252
1253                                     , THIS TEST CHECKS DBIT
1254                                     , *****
1255 005656 000004 TST4 SCOPE
1256 005660 112737 000004 002022      MOVB #4, TNUMB
1257 005666 012737 000020 001124      MOV #DBIT, $GDDAT
1258 005674 113737 017504 016326      MOVB M5, EM2X , FIX ERROR MESSAGE
1259 005702 004737 012230              JSR PC, BITSET , SET D BIT
1260 005706 104002                      ERROR 2 , BIT NOT SETTING
1261 005710 005037 001124              CLR $GDDAT
1262 005714 004737 012230              JSR PC, BITSET
1263 005720 104002                      ERROR 2 , BIT NOT CLEARING
1264 005722 112777 000074 174076      MOVB #74, @CSR , SET FEW BITS AT CSR
1265 005730 142777 000020 174070      BICB #DBIT, @CSR , CLEAR DBIT
1266 005736 005037 001766              CLR YLOOP , WAIT
1267 005742 005237 001766 64$        INC YLOOP
1268 005746 023727 001766 000007      CMP YLOOP, #7
1269 005754 001372                      BNE 64$
1270 005756 117737 174044 001124      MOVB @CSR, $GDDAT
1271 005764 132737 000177 001124      BITB #177, $GDDAT
1272 005772 001401                      BEQ 1$ , CHECK IF DBIT CLEAR'NG CLEARS CSR
1273 005774 104016                      ERROR 16 , CLEARING DBIT DOES NOT CLEAR CSR
1274 005776
1275 005776 152777 000002 174022 1$        BISB #CBIT, @CSR , SET C BIT
1276 006004 005037 001766              CLR YLOOP , WAIT
1277 006010 005237 001766 65$        INC YLOOP
1278 006014 023727 001766 000007      CMP YLOOP, #7
1279 006022 001372                      BNE 65$
1280 006024 152777 000002 173774      BISB #CBIT, @CSR , DO IT AGAIN
1281 006032 005037 001766              CLR YLOOP , WAIT
1282 006036 005237 001766 66$        INC YLOOP
1283 006042 023727 001766 000007      CMP YLOOP, #7
  
```

```
1284 006050 001372          BNE      66$
1285
1286
1287          , THIS TEST CHECKS TBIT
1288          , , *****
1289 006052 000004          TST5   SCOPE
1290 006054 112737 000005 002022      MOVB   #5, TNUMB
1291 006062 012737 000010 001124      MOV    #TBIT, $GDDAT
1292 006070 113737 017501 016326      MOVB  M4, EM2X      , SET ERROR MESSAGE
1293 006076 004737 012230          JSR    PC, BITSET   , SET AND CHECK T BIT
1294 006102 104002          ERROR 2          , BIT IS NOT SETTING
1295 006104 005037 001124          CLR    $GDDAT
1296 006110 004737 012230          JSR    PC, BITSET   , CLEAR T BIT
1297 006114 104002          ERROR 2          , BIT NOT CLEAR
1298 006116 004737 012360          JSR    PC, CLRINT   , CLEAR ALL INTERRUPT CAUSED BY T BIT
1299
1300          , THIS TEST CHECKS GBIT
1301          , , *****
1302 006122 000004          TST6   SCOPE
1303 006124 112737 000006 002022      MOVB   #6, TNUMB
1304 006132 012737 000004 001124      MOV    #GBIT, $GDDAT
1305 006140 113737 017476 016326      MOVB  M3, EM2X      , FIX ERPOR MESSAGE
1306 006146 004737 012230          JSR    PC, BITSET   , SET AND CHECK G BIT
1307 006152 104002          ERROR 2          , BIT NOT SETTING
1308 006154 005037 001124          CLR    $GDDAT
1309 006160 004737 012230          JSR    PC, BITSET   , CLEAR G BIT
1310 006164 104002          ERROR 2          , BIT'S NOT CLEARING
1311
1312
1313          ; THIS TEST CHECKS RBIT
1314          , , *****
1315 006166 000004          TST7   SCOPE
1316 006170 112737 000007 002022      MOVB   #7, TNUMB
1317 006176 012737 000001 001124      MOV    #RBIT, $GDDAT
1318 006204 113737 017474 016326      MOVB  M2, EM2X      , SET ERROR MESSAGE
1319 006212 004737 012230          JSR    PC, BITSET   , SET BIT
1320 006216 104002          ERROR 2          , BIT IS NOT SETTING
1321 006220 005037 001124          CLP    $GDDAT
1322 006224 004737 012230          JSR    PC, BITSET   , CLEAR BIT
1323 006230 104002          ERROR 2          , BIT IS NOT CLEAR
```

1324
1325
1326
1327
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1360
1361

006232 000004
006234 112737 000010 002022
006242 112777 000040 173556
006250 013700 002042
006254 005001
006256 005037 001124
006262 005037 001126
006266 111001
006270 042701 177400
006274 120001
006276 001405
006300 110037 001124
006304 110137 001126
006310 104004
006312 005200
006314 122700 000376
006320 001362
006322 152777 000002 173476
006330 005037 001766
006334 005237 001766
006340 023727 001766 000007
006346 001372
006350 152777 000002 173450
006356 005037 001766
006362 005237 001766
006366 023727 001766 000007
006374 001372

```
*****  
      , THIS TEST CHECKS ALL BITS OF DBUS IN A MAINTENANCE MODE  
      , IF MBIT IS SET AND CPU ADDRESSES ANY LOCATION BETWEEN  
      , 171000 AND 171375 IT SHOULD READ BACK A LOWER  
      , BYTE OF AN ADDRESS.  
*****  
*****  
TST10 SCOPE  
      MOVB #10, TNUMB  
      MOVB #MBIT, @CSR      , SET MAINTENANCE MODE  
      MOV  BASE, R0  
      CLR  R1  
      CLR  $GDDAT  
      CLR  $BDDAT  
15     MOVB (R0), R1      , READ FIRST ADDRESS  
      BIC  #177400, R1  
      CMPB R0, R1      , CHECK IF DBUS=ADDRESS  
      BEQ  2$  
      MOVB R0, $GDDAT  
      MOVB R1, $BDDAT  
      ERROR 4      , DBUS BIT STACK  
25     INC  R0  
      CMPB #376, R0      , IS IT THE LAST ADDRESS  
      BNE  1$  
      BLSB #CBIT, @CSR      , SET C BIT  
      CLR  YLOOP      , WAIT  
64$    INC  YLOOP  
      CMP  YLOOP, #7  
      BNE  64$  
      BLSB #CBIT, @CSR      , DO IT AGAIN  
      CLR  YLOOP      , WAIT  
65$    INC  YLOOP  
      CMP  YLOOP, #7  
      BNE  65$
```

```

1362
1363 ;,*****
1364
1365 ; THIS TEST WILL CHECK MAINTENANCE INTERRUPT
1366 ; IF MBIT & EBIT ARE SET IOCM WILL GENERATE AN INTERRUPT
1367 ; AT LOCATION 234 AND IAR WILL HAVE LOWER BYTE
1368 ; OF CSR ADDRESS
1369
1370 ;,*****
1371
1372 ;,*****
1373 006376 000004 TST11 SCOPE
1374 006400 112737 000011 002022 MOVB #11, TNUMB
1375 006406 012777 006502 173416 MOV #5$, @VECTO ; SET VECTOR ADDRESS
1376 006414 012777 000340 173412 MOV #PR7, @VECTORA ; SET VECTOR+2 ADDRESS
1377 006422 013746 000004 MOV ERRVEC, -(SP) ; PUSH ERRVEC ON STACK
1378 006426 012737 006664 000004 MOV #8$, ERRVEC
1379 006434 012746 000000 MOV #PRO, -(SP) ; SET PSW TO PRIORITY 0
1380 006440 012746 006446 MOV #64$, -(SP)
1381 006444 000002 RTI
1382 006446 000240 64$ NOP
1383 006450 112777 000140 173350 1$ MOVB #EBIT!#MBIT, @CSR, START INTERRUPT
1384 006456 005037 001766 CLR YLOOP ; WAIT
1385 006462 005237 001766 65$ INC YLOOP
1386 006466 023727 001766 177777 CMP YLOOP, #-1
1387 006474 001372 BNE 65$
1388 006476 104014 ERROR 14 ; NO INTERRUPT
1389 006500 000431 BR 6$
1390 006502 022626 5$ CMP (SP)+, (SP)+ ; INTERRUPT WORKED
1391 006504 117737 173320 002046 MOVB @IAR, TEMP ; READ ADDRESS OF INTERRUPTING MODULE
1392 006512 122737 000377 002046 CMPB #377, TEMP ; IAR SHOULD HAVE ALL ONES
1393 006520 001407 BEQ 7$
1394 006522 012737 000377 001124 MOV #377, $GDDAT
1395 006530 013737 002046 001126 MOV TEMP, $BDDAT
1396 006536 104006 ERROR 6 ; WRONG PATTERN IN IAR
1397 006540 152777 000001 173260 7$ BISB #RBIT, @CSR
1398 006546 105777 173254 TSTB @CSR ; CLEAR INTERRUPT
1399 006552 132777 000200 173246 BITB #FBIT, @CSR ; CHECK IF INTERRUPT CLEAR
1400 006560 001401 BEQ 6$
1401 006562 104015 ERROR 15 ; RIF BIT NOT CLEARING INTERRUPT
1402 006564 6$
1403 006564 152777 000002 173234 BISB #CBIT, @CSR ; SET C BIT
1404 006572 005037 001766 CLR YLOOP ; WAIT
1405 006576 005237 001766 66$ INC YLOOP
1406 006602 023727 001766 000007 CMP YLOOP, #7
1407 006610 001372 BNE 66$
1408 006612 152777 000002 173206 BISB #CBIT, @CSR ; DO IT AGAIN
1409 006620 005037 001766 CLR YLOOP ; WAIT
1410 006624 005237 001766 67$ INC YLOOP
1411 006630 023727 001766 000007 CMP YLOOP, #7
1412 006636 001372 BNE 67$
1413 006640 012637 000004 MOV (SP)+, ERRVEC ; POP STACK INTO ERRVEC
1414 006644 012746 000000 MOV #PRO, -(SP) ; SET PSW TO PRIORITY 0
1415 006650 012746 006656 MOV #68$, -(SP)
1416 006654 000002 RTI
1417 006656 000240 68$ NOP
  
```

1418 006660 000004
1419 006662 000207
1420
1421 006664 022626
1422 006666 104022
1423 006670 000735
1424

SCOPE
RTS PC
8\$ CMP (SP)+, (SP)+
ERROR 22
BR 6\$

. ADDRESS TEST WITH MBIT SET NOT WORKING

1425
1426
1427
1428
1429
1430
1431
1432
1433
1434
1435
1436
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1448
1449
1450
1451

006672

006672

006702

006710

006716

006724

006730

006736

006744

006752

006756

006764

006770

006772

000004

112737

152777

112737

012737

004737

152777

112737

012737

004737

142777

004737

000004

000207

000012

000020

000000

000004

011746

000010

000377

000004

011746

000010

012360

002022

173116

001124

002004

002004

173070

001124

002004

002004

173042

SBTTL DIGITAL MODUL TEST ,M5010
.,XX

, THIS TEST WILL CHECK M5010 MODULE
, 32 BIT NONISOLATED DC SENSE

.,XX

M5010
.,XX

```
TST12. SCOPE
        MOVB    #12,TNUMB
        BISB    #0BIT,@CSR      ;SET DISABLE BIT
        MOVB    #0,$GDDAT      ;SET WHAT DATA SHOULD BE
        MOV     #4,BYTNUM      ;SET NUMBER OF BYTES
        JSR     PC,TSTBYT      ;CHECK IF DATA IS CORRECT

        BISB    #TBIT,@CSR
        MOVB    #377,$GDDAT    ;SET WHAT DATA SHOULD BE
        MOV     #4,BYTNUM      ;SET NUMBER OF BYTES
        JSR     PC,TSTBYT      ;CHECK IF DATA IS CORRECT

        BICB    #TBIT,@CSR
        JSR     PC,CLRINT
        SCOPE
        RTS     PC
```



```

1452
1453          SBTTL  DIGITAL MODULE TEST ,M5013
1454
1455          ,, *****
1456
1457          , THIS TEST CHECKS M5013 MODULE
1458          , 8 BIT AC SENSE
1459
1460          ,, *****
1461
1462
1463 006774          M5013
1464          ,, *****
1465 006774 000004          TST13  SCOPE
1466 006776 112737 000013 002022          MOVB  #13, TNUMB
1467 007004 152777 000020 173014          BISB  #DBIT, @CSR          ; SET DISABLE BIT
1468 007012 112737 000000 001124          MOVB  #0, %GDDAT          ; SET WHAT DATA SHOULD BE
1469 007020 012737 000001 002004          MOV   #1, BYTNUM          ; SET NUMBER OF BYTES
1470 007026 004737 011746          JSR   PC, TSTBYT          ; CHECK IF DATA IS CORRECT
1471
1472 007032 152777 000010 172766          BISB  #TBIT, @CSR          ; SET COMPLIM BIT
1473 007040 004737 012360          JSR   PC, CLRINT
1474 007044 112737 000377 001124          MOVB  #377, %GDDAT          ; SET WHAT DATA SHOULD BE
1475 007052 012737 000001 002004          MOV   #1, BYTNUM          ; SET NUMBER OF BYTES
1476 007060 004737 011746          JSR   PC, TSTBYT          ; CHECK IF DATA IS CORRECT
1477 007064 142777 000010 172734          BICB  #TBIT, @CSR
1478 007072 004737 012360          JSR   PC, CLRINT
1479
1480
1481 007076 012746 000000          MOV   #PRO, -(SP)          ; SET PSW TO PRIORITY 0
1482 007102 012746 007110          MOV   #64$, -(SP)
1483 007106 000002          RTI
1484 007110 000240          64$  NOP
1485 007112 012777 007210 172712          MOV   #5$, @VECTO          ; SET INTERRUPTVECTOR ADDRESS
1486 007120 012777 000340 172706          MOV   #PR7, @VECTORA
1487 007126 152777 000010 172672          BISB  #TBIT, @CSR          ; START INTERRUPT
1488 007134 152777 000100 172664          BISB  #EBIT, @CSR          ; ENABLE INTERRUPT
1489 007142 005037 001766          CLR   YLOOP          ; WAIT
1490 007146 005237 001766          65$  INC   YLOOP
1491 007152 023727 001766 177777          CMP   YLOOP, #-1
1492 007160 001372          BNE   65$
1493 007162 104005          ERROR 5          ; NO INTERRUPT
1494 007164 142777 000100 172634          7$  BICB  #EBIT, @CSR          ; CLEAR CSR
1495 007172 142777 000010 172626          BICB  #TBIT, @CSR
1496 007200 004737 012360          JSR   PC, CLRINT          ; CLEAR ALL INTERRUPT
1497 007204 000004          SCOPE
1498 007206 000207          RTS   PC          ; RETURN TO MONITOR
1499
1500 007210 022626          5$  CMP   (SP)+, (SP)+          ; RESET STACK
1501 007212 005037 002050          CLR   TEMP1
1502 007216 013704 000004          MOV   ERRVEC, R4
1503 007222 012737 007430 000004          MOV   #4$, ERRVEC
1504 007230 117737 172574 002046          6$  MOVB  @IAR, TEMP          ; CHECK WHICH I/O INTERRUPTED
1505 007236 152777 000001 172562          BISB  #RBIT, @CSR          ; HERE IF INTERRUPT
1506 007244 123737 002046 002006          CMPB  TEMP, TADDR          ; IS IT MUT
1507 007252 001414          BEQ   2$
  
```



```
1541          SBTTL  DIGITAL MODULE TEST ,M6010
1542
1543          ,, *****
1544
1545          , THIS TEST CHECKS M6010 MODULE
1546          , 32 BIT NONISOLATED DC OUT
1547
1548          ,, *****
1549
1550 007440     M6010
1551          ,, *****
1552 007440     000004     TST14  SCOPE
1553 007442     112737     000014     002022     MOV      #14, TNUMB
1554 007450     152777     000020     172350     BISB    #DBIT, @CSR      , SET D BIT
1555 007456     013700     002006     MOV     TADDR, R0
1556 007462     105020     CLR     (R0)+          , CLEAR ALL FOUR BYTES OF I/O REGISTER
1557 007464     105020     CLR     (R0)+
1558 007466     105020     CLR     (R0)+
1559 007470     105020     CLR     (R0)+
1560 007472     005000     CLR     R0
1561 007474     005001     4$     CLR     R1
1562 007476     005037     001770     CLR     DBUFF          , CLEAR IMIGE OF I/O REGISRERS
1563 007502     005037     001772     CLR     DBUFF+2
1564 007506     116160     001774     001770     3$     MOV     PATT(R1), DBUFF(R0), SET DATA PATTERN IN IMIGE
1565 007514     013737     002006     002046     MOV     TADDR, TEMP
1566 007522     060037     002046     ADD     R0, TEMP
1567 007526     116177     001774     172312     MOV     PATT(R1), @TEMP  , SET DATA IN I/O REG
1568 007534     005002     CLR     R2
1569 007536     013737     002006     002050     2$     MOV     TADDR, TEMP1
1570 007544     060237     002050     ADD     R2, TEMP1
1571 007550     117737     172274     001126     MOV     @TEMP1, $BDDAT  , READ I/O REGISTER
1572 007556     123762     001126     001770     CMPB   $BDDAT, DBUFF(R2), IS DATA OK?
1573 007564     001404     BEQ     1$
1574 007566     116237     001770     001124     MOV     DBUFF(R2), $GDDAT
1575 007574     104017     ERROR  17          , DATA ERROR
1576 007576     005202     1$     INC     R2
1577 007600     022702     000004     CMP     #4, R2
1578 007604     001354     BNE     2$          , TEST IF OTHER BYTES ARE OK
1579 007606     005201     INC     R1
1580 007610     022701     000004     CMP     #4, R1          , LAST PATTERN?
1581 007614     001334     BNE     3$          , TEST NEXT DATA PATTERN
1582 007616     005200     INC     R0
1583 007620     022700     000004     CMP     #4, R0          , LAST BYTE?
1584 007624     001323     BNE     4$          , NO. TEST NEXT BYTE
1585 007626     000004     SCOPE
1586 007630     000207     RTS     PC
1587
1588
```

```

1589 . SBTTL DIGITAL MODULE TEST , M6012, M6013
1590
1591 ., *****
1592
1593 . THIS TEST CHECKS MODULES M6012 AND M6013
1594 . 8 BIT AC AND DC OUT
1595
1596 ., *****
1597
1598
1599 007632 M6012
1600 007632 M6013
1601 ., *****
1602 007632 000004 TST15 SCOPE
1603 007634 112737 000015 002022 MOVB #15, TNUMB
1604 007642 152777 000020 172156 BISB #DBIT, @CSR
1605 007650 012737 000125 001124 MOV #125, $GDDAT ; SET DATA PATTERN
1606 007656 012737 000001 002004 MOV #1, BYTNUM ; SET NUMBER OF BYTES
1607 007664 004737 012112 JSR PC, SETBYT ; SET DATA IN I/O MODULE
1608 007670 012737 000001 002004 MOV #1, BYTNUM
1609 007676 004737 011746 JSR PC, TSTBYT ; TEST IF DATA IS SET CORRECTLY
1610
1611
1612 007702 012737 000252 001124 MOV #252, $GDDAT ; SET DATA PATTERN
1613 007710 012737 000001 002004 MOV #1, BYTNUM ; SET NUMBER OF BYTES
1614 007716 004737 012112 JSR PC, SETBYT ; SET DATA IN I/O MODULE
1615 007722 012737 000001 002004 MOV #1, BYTNUM
1616 007730 004737 011746 JSR PC, TSTBYT ; TEST IF DATA IS SET CORRECTLY
1617
1618 007734 012737 000377 001124 MOV #377, $GDDAT ; SET DATA PATTERN
1619 007742 012737 000001 002004 MOV #1, BYTNUM ; SET NUMBER OF BYTES
1620 007750 004737 012112 JSR PC, SETBYT ; SET DATA IN I/O MODULE
1621 007754 012737 000001 002004 MOV #1, BYTNUM
1622 007762 004737 011746 JSR PC, TSTBYT ; TEST IF DATA IS SET CORRECTLY
1623
1624
1625 007766 012737 000000 001124 MOV #0, $GDDAT ; SET DATA PATTERN
1626 007774 012737 000001 002004 MOV #1, BYTNUM ; SET NUMBER OF BYTES
1627 010002 004737 012112 JSR PC, SETBYT ; SET DATA IN I/O MODULE
1628 010006 012737 000001 002004 MOV #1, BYTNUM
1629 010014 004737 011746 JSR PC, TSTBYT ; TEST IF DATA IS SET CORRECTLY
1630 010020 000004 SCOPE
1631 010022 000207 RTS PC

```

```
1632          SBTTL  DIGITAL MODULE TEST ,M6011
1633          ,, *****
1634
1635          , THIS TEST CHECKS MODULE M6011
1636          , ONE SHOT DC OUT
1637          ,, *****
1638
1639
1640 010024    M6011
1641          ,, *****
1642 010024    000004    TST16  SCOPE
1643 010026    112737    000016    002022    MOV      #16, TNUMB
1644 010034    013737    002006    002010    MOV      TADDR, TADDR1
1645 010042    005237    002010    INC      TADDR1
1646 010046    152777    000020    171752    BISB    #DBIT, @CSR
1647 010054    005037    002044    CLR     CLK          , CLEAR CLOCK COUNTER
1648 010060    012777    012352    171774    MOV     #COUNT, @CLKVC , SET LOC 100- CLOCK VECTOR
1649 010066    012777    000340    171770    MOV     #PR7, @CLKVCA , SET LOC 102
1650 010074    012746    000000    MOV     #PR0, -(SP) , SET PSW TO PRIORITY 0
1651 010100    012746    010106    MOV     #64$, -(SP)
1652 010104    000002    RTI
1653 010106    000240    64$    NOP
1654 010110    012737    000252    001124    MOV     #252, $GDDAT , SET DATA PATTERN
1655 010116    012737    000002    002004    MOV     #2, BYTNUM , SET NUMBER OF BYTES
1656 010124    004737    012112    JSR     PC, SETBYT , SET DATA IN I/O MODULE
1657 010130    012737    000002    002004    MOV     #2, BYTNUM
1658 010136    004737    011746    JSR     PC, TSTBYT , TEST IF DATA IS SET CORRECTLY
1659 010142    005001    CLR     R1
1660 010144    005201    11$    INC     R1          , CHECK IF LINE CLOCK IS INTERRUPTING
1661 010146    001376    BNE     11$
1662 010150    005737    002044    TST     CLK
1663 010154    001010    BNE     1$
1664 010156    104025    ERROR 25 , LINE CLOCK ISN'T INTERRUPTING
1665 010160    000453    BR      M6011B
1666 010162    105777    171620    2$    TSTB   @TADDR
1667 010166    001003    BNE     1$
1668 010170    105777    171614    TSTB   @TADDR1
1669 010174    001410    BEQ     M6011A
1670 010176    022737    001131    002044    1$    CMP     #601, CLK , WAIT 10SEC
1671 010204    001366    BNE     2$
1672 010206    005037    001124    CLR     $GDDAT
1673 010212    004737    012030    JSR     PC, TSTONE , PATTERN SHOULD BE CLEAR
1674 010216    M6011A
1675 010216    005037    002044    CLR     CLK          , CLEAR CLOCK COUNTER
1676 010222    012737    000125    001124    MOV     #125, $GDDAT , SET DATA PATTERN
1677 010230    012737    000002    002004    MOV     #2, BYTNUM , SET NUMBER OF BYTES
1678 010236    004737    012112    JSR     PC, SETBYT , SET DATA IN I/O MODULE
1679 010242    012737    000002    002004    MOV     #2, BYTNUM
1680 010250    004737    011746    JSR     PC, TSTBYT , TEST IF DATA IS SET CORRECTLY
1681 010254    105777    171526    2$    TSTB   @TADDR
1682 010260    001003    BNE     1$
1683 010262    105777    171522    TSTB   @TADDR1
1684 010266    001410    BEQ     M6011B
1685 010270    022737    001131    002044    1$    CMP     #601, CLK , WAIT 10 SEC
1686 010276    001366    BNE     2$
1687 010300    005037    001124    CLR     $GDDAT
```

MAIN MACY11 30(1046) 25-OCT-77 10 10 PAGE 43
DUPLAR P11 25-OCT-77 10 02 DIGITAL MODULE TEST ,M6011

SEQ 0045

| | | | | | | | |
|------|--------|--------|--------|--------|-------|---------------|--------------------------|
| 1688 | 010304 | 004737 | 012030 | | JSR | PC,TSTONE | ,PATTERN SHOULD BE CLEAR |
| 1689 | 010310 | 000004 | | M6011B | SCOPE | | |
| 1690 | 010312 | 012777 | 012350 | 171542 | MOV | #NOCLK,@CLKVC | ,RESTORE CLOCK VECTOR |
| 1691 | 010320 | 000207 | | | RTS | PC | |

SBTTL DIGITAL MODULE TEST ,M5011

,, *****

, THIS TEST CHECKS M5011 MODULE
, 16 BIT CHANGE OF STATE SENSE

,, *****

M5011

,, *****

| | | | | | | | | |
|------|--------|--------|--------|--------|---------|---|--|--|
| 1704 | 010322 | 000004 | | | TST17 | SCOPE | | |
| 1705 | 010324 | 112737 | 000017 | 002022 | MOVB | #17, TNUMB | | |
| 1706 | 010332 | 013737 | 002006 | 002014 | MOV | TADDR, COSADR | | |
| 1707 | 010340 | 062737 | 000002 | 002014 | ADD | #2, COSADR | | |
| 1708 | 010346 | 152777 | 000030 | 171452 | BISB | #DBIT!#TBIT, @CSR, SET DISABLE BIT AND TBIT | | |
| 1709 | 010354 | 112737 | 000377 | 001124 | MOVB | #377, \$GDDAT ; SET WHAT DATA SHOULD BE | | |
| 1710 | 010362 | 012737 | 000002 | 002004 | MOV | #2, BYTNUM ; SET NUMBER OF BYTES | | |
| 1711 | 010370 | 004737 | 011746 | | JSR | PC, TSTBYT ; CHECK IF DATA IS CORRECT | | |
| 1712 | | | | | | | | |
| 1713 | 010374 | 142777 | 000010 | 171424 | BICB | #TBIT, @CSR ; CLEAR COMPLIM BIT | | |
| 1714 | 010402 | 112737 | 000000 | 001124 | MOVB | #0, \$GDDAT ; SET WHAT DATA SHOULD BE | | |
| 1715 | 010410 | 012737 | 000002 | 002004 | MOV | #2, BYTNUM ; SET NUMBER OF BYTES | | |
| 1716 | 010416 | 004737 | 011746 | | JSR | PC, TSTBYT ; CHECK IF DATA IS CORRECT | | |
| 1717 | 010422 | 004737 | 012360 | | JSR | PC, CLRINT | | |
| 1718 | 010426 | 012746 | 000J00 | | MOV | #PRO, -(SP) ; SET PSW TO PRIORITY 0 | | |
| 1719 | 010432 | 012746 | 010440 | | MOV | #64\$, -(SP) | | |
| 1720 | 010436 | 000002 | | | RTI | | | |
| 1721 | 010440 | 000240 | | | NOP | | | |
| 1722 | 010442 | 012777 | 010540 | 171362 | MOV | #1\$, @VECTO ; SET INTERRUPT VECTOR | | |
| 1723 | 010450 | 012777 | 000340 | 171356 | MOV | #PR7, @VECTORA | | |
| 1724 | 010456 | 152777 | 000010 | 171342 | BISB | #TBIT, @CSR ; START INTERRUPT | | |
| 1725 | 010464 | 152777 | 000100 | 171334 | BISB | #EBIT, @CSR ; ENABLE INTERRUPT | | |
| 1726 | 010472 | 005037 | 001766 | | CLR | YLOOP ; WAIT | | |
| 1727 | 010476 | 005237 | 001766 | | INC | YLOOP | | |
| 1728 | 010502 | 023727 | 001766 | 177777 | 65\$ | CMP | YLOOP, #-1 | |
| 1729 | 010510 | 001372 | | | BNE | 65\$ | | |
| 1730 | 010512 | 104005 | | | ERROR 5 | | | |
| 1731 | 010514 | 142777 | 000100 | 171304 | 2\$ | BICB | #EBIT, @CSR ; NO INTERRUPT | |
| 1732 | 010522 | 142777 | 000010 | 171276 | | BICB | #TBIT, @CSR ; CLEAR CSR | |
| 1733 | 010530 | 004737 | 012360 | | | JSR | PC, CLRINT ; CLEAR ALL INTEPRUPTS | |
| 1734 | 010534 | 000004 | | | | SCOPE | | |
| 1735 | 010536 | 000207 | | | | RTS | PC ; RETURN TO MONITOR | |
| 1736 | | | | | | | | |
| 1737 | 010540 | 022626 | | | 1\$ | CMP | (SP)+, (SP)+ ; ADJUST STOCK POINTER | |
| 1738 | 010542 | 005037 | 002050 | | | CLR | TEMP1 | |
| 1739 | 010546 | 013704 | 000004 | | | MOV | ERRVEC, R4 | |
| 1740 | 010552 | 012737 | 011140 | 000004 | | MOV | #10\$, ERRVEC | |
| 1741 | 010560 | 117737 | 171244 | 002046 | 7\$ | MOVB | @IAR, TEMP ; FIND WHICH I/O INTERRUPTED | |
| 1742 | 010566 | 123737 | 002046 | 002006 | | CMPB | TEMP, TADDR ; IS IT MUT | |
| 1743 | 010574 | 001421 | | | | BEQ | 8\$; YES | |
| 1744 | 010576 | 053737 | 002042 | 002046 | | BIS | BASE, TEMP ; ASSEMBLE ADDRESS OF INTEPRUPTING MODULE | |
| 1745 | 010604 | 152777 | 000001 | 171214 | | BISB | #RBIT, @CSR ; CLEAR INTERRUPT | |
| 1746 | 010612 | 105777 | 171230 | | | TSTB | @TEMP | |
| 1747 | 010616 | 005237 | 002050 | | | INC | TEMP1 | |

| | | | | | | | | |
|------|--------|--------|--------|--------|------|----------|------------------|---|
| 1748 | 010622 | 022737 | 000400 | 002050 | | CMP | #400, TEMP1 | |
| 1749 | 010630 | 001353 | | | | BNE | 7\$ | |
| 1750 | 010632 | 104005 | | | | ERROR 5 | | , NO INTERRUPT |
| 1751 | 010634 | 000137 | 010514 | | | JMP | 2\$ | |
| 1752 | 010640 | 142777 | 000100 | 171160 | 8\$ | BICB | #EBIT, @CSR | |
| 1753 | 010646 | 112737 | 000377 | 001124 | | MOVB | #377, \$GDDAT | |
| 1754 | 010654 | 117737 | 171134 | 001126 | | MOVB | @COSADR, \$BDDAT | , TEST COS REGISTER |
| 1755 | 010662 | 123737 | 001124 | 001126 | | CMPB | \$GDDAT, \$BDDAT | , IS IT ALL ONES |
| 1756 | 010670 | 001401 | | | | BEQ | 3\$ | |
| 1757 | 010672 | 104020 | | | | ERROR 20 | | , COS REGISTER ERROR |
| 1758 | 010674 | 152777 | 000001 | 171124 | 3\$ | BISB | #RBIT, @CSR | |
| 1759 | 010702 | 105777 | 171106 | | | TSTB | @COSADR | , CLEAR INTERRUPT |
| 1760 | 010706 | 005037 | 001124 | | | CLR | \$GDDAT | |
| 1761 | 010712 | 117737 | 171076 | 001126 | | MOVB | @COSADR, \$BDDAT | |
| 1762 | 010720 | 001401 | | | | BEQ | 4\$ | |
| 1763 | 010722 | 104013 | | | | ERROR 13 | | , RIF BIT NOT CLEARING FF |
| 1764 | 010724 | 005237 | 002014 | | 4\$ | INC | COSADR | |
| 1765 | 010730 | 005237 | 002006 | | | INC | TADDR | , TEST NEXT BYTE |
| 1766 | 010734 | 117737 | 171070 | 002046 | | MOVB | @IAR, TEMP | |
| 1767 | 010742 | 123737 | 002046 | 002006 | | CMPB | TEMP, TADDR | , CHECK IF IT INTERRUPTED |
| 1768 | 010750 | 001403 | | | | BEQ | 11\$ | |
| 1769 | 010752 | 104005 | | | | ERROR 5 | | , NO INTERRUPT |
| 1770 | 010754 | 000137 | 010514 | | | JMP | 2\$ | |
| 1771 | 010760 | 112737 | 000377 | 001124 | 11\$ | MOVB | #377, \$GDDAT | |
| 1772 | 010766 | 117737 | 171022 | 001126 | | MOVB | @COSADR, \$BDDAT | , CHECK IF COS REGISTER IS ALL ONES |
| 1773 | 010774 | 123737 | 001124 | 001126 | | CMPB | \$GDDAT, \$BDDAT | |
| 1774 | 011002 | 001401 | | | | BEQ | 5\$ | |
| 1775 | 011004 | 104020 | | | | ERROR 20 | | , COS REGISTER ERROR |
| 1776 | 011006 | 152777 | 000001 | 171012 | 5\$ | BISB | #RBIT, @CSR | |
| 1777 | 011014 | 105777 | 170774 | | | TSTB | @COSADR | , CLEAR INTERRUPT |
| 1778 | 011020 | 005037 | 001124 | | | CLR | \$GDDAT | |
| 1779 | 011024 | 117737 | 170764 | 001126 | | MOVB | @COSADR, \$BDDAT | |
| 1780 | 011032 | 001401 | | | | BEQ | 6\$ | |
| 1781 | 011034 | 104013 | | | | ERROR 13 | | , RIF BIT NOT CLEARING FF |
| 1782 | 011036 | 005337 | 002014 | | 6\$ | DEC | COSADR | |
| 1783 | 011042 | 005337 | 002006 | | | DEC | TADDR | , RESET ADDRESS |
| 1784 | 011046 | 004737 | 012360 | | | JSR | PC, CLRINT | , CLEAR REMAINING INTERRUPTS |
| 1785 | 011052 | 132777 | 000010 | 170746 | | BITB | #TBIT, @CSR | , CHECK IF CLEARING T BIT CAUSE INTERPUPT |
| 1786 | 011060 | 001615 | | | | BEQ | 2\$ | |
| 1787 | 011062 | 012746 | 000000 | | | MOV | #PRO, -(SP) | , SET PSW TO PRIORITY 0 |
| 1788 | 011066 | 012746 | 011074 | | | MOV | #66\$, -(SP) | |
| 1789 | 011072 | 000002 | | | | RTI | | |
| 1790 | 011074 | 000240 | | | 66\$ | NOP | | |
| 1791 | 011076 | 142777 | 000010 | 170722 | | BICB | #TBIT, @CSR | , CLEAR INPUTS |
| 1792 | 011104 | 152777 | 000100 | 170714 | | BISB | #EBIT, @CSR | |
| 1793 | 011112 | 005037 | 001766 | | | CLR | YLOOP | , WAIT |
| 1794 | 011116 | 005237 | 001766 | | 67\$ | INC | YLOOP | |
| 1795 | 011122 | 023727 | 001766 | 177777 | | CMP | YLOOP, #-1 | |
| 1796 | 011130 | 001372 | | | | BNE | 67\$ | |
| 1797 | 011132 | 104005 | | | | ERROR 5 | | , NO INTERRUPT |
| 1798 | 011134 | 000137 | 010514 | | | JMP | 2\$ | |
| 1799 | | | | | | | | |
| 1800 | 011140 | 022626 | | | 10\$ | CMP | (SP)+, (SP)+ | , RESET STACK POINTER |
| 1801 | 011142 | 104005 | | | | ERROR 5 | | , NO INTERRUPT |
| 1802 | 011144 | 000137 | 010514 | | | JMP | 2\$ | |


```

1803 011150 000240 ANALOG NOP
1804 SBTTL DIGITAL MODULE TEST ,M5012
1805
1806 ,,*****
1807
1808 , THIS TEST CHECKS M5012 MODULE
1809 , ISOLATED 16 BIT DC INPUT
1810
1811 ,,*****
1812
1813 011152 M5012
1814 ,,*****
1815 011152 000004 TST20 SCOPE
1816 011154 012737 000020 002022 MOV #20, TNUMB
1817 011162 152777 000020 170636 BISB #DBIT, @CSR
1818 011170 112737 000000 001124 MOVB #0, $GDDAT ; SET WHAT DATA SHOULD BE
1819 011176 012737 000002 002004 MOV #2, BYTNUM ; SET NUMBER OF BYTES
1820 011204 004737 011746 JSR PC, TSTBYT ; CHECK IF DATA IS CORRECT
1821
1822 011210 152777 000010 170610 BISB #TBIT, @CSR
1823 011216 112737 000377 001124 MOVB #377, $GDDAT ; SET WHAT DATA SHOULD BE
1824 011224 012737 000002 002004 MOV #2, BYTNUM ; SET NUMBER OF BYTES
1825 011232 004737 011746 JSR PC, TSTBYT ; CHECK IF DATA IS CORRECT
1826 011236 142777 000010 170562 BICB #TBIT, @CSR
1827 011244 004737 012360 JSR PC, CLRINT
1828
1829 011250 012746 000J00 MOV #PR0, -(SP) ; SET PSW TO PRIORITY 0
1830 011254 012746 011262 MOV #64$, -(SP)
1831 011260 000002 RTI
1832 011262 000240 64$ NOP
1833 011264 012777 011362 170540 MOV #6$, @VECTO ; SET INTERRUPT VECTOR
1834 011272 012777 000340 170534 MOV #PR7, @VECTOA
1835 011300 152777 000100 170520 BISB #EBIT, @CSR ; ENABLE INTERRUPT
1836 011306 152777 000010 170512 BISB #TBIT, @CSR ; START INTERRUPT
1837 011314 005037 001766 CLR YLOOP ; WAIT
1838 011320 005237 001766 65$ INC YLOOP
1839 011324 023727 001766 177777 CMP YLOOP, #-1
1840 011332 001372 BNE 65$
1841 011334 104005 ERROR 5 ; NO INTERRUPT
1842 011336 142777 000100 170462 7$ BICB #EBIT, @CSR ; CLEAR CSR
1843 011344 142777 000010 170454 BICB #TBIT, @CSR
1844 011352 004737 012360 JSR PC, CLRINT ; CLEAR ALL INTERRUPT
1845 011356 000004 SCOPE
1846 011360 000207 RTS PC ; RETURN TU MONITOR
1847
1848 011362 022626 6$ CMP (SP)+, (SP)+ ; RESET STACK POINTER
1849 011364 005037 002050 CLR TEMP1
1850 011370 013704 000004 MOV ERRVEC, R4
1851 011374 012737 011626 000004 MOV #4$, ERRVEC
1852 011402 117737 170422 002046 3$ MOVB @IAR, TEMP ; FIND WHICH I/O INTERRUPTED
1853 011410 152777 000001 170410 BISB #RBIT, @CSR
1854 011416 123737 002046 002006 CMPB TEMP, TADDR ; IS IT MUT
1855 011424 001415 BEQ 2$ ; YES
1856 011426 053737 002042 002046 BIS BASE, TEMP ; ASSEMBLE ADDRESS OF INTEPRUPTING I/O
1857 011434 105777 170406 TSTB @TEMP ; CLEAR INTERRUPT
1858 011440 005237 002050 INC TEMP1
  
```

| | | | | | | | | |
|------|--------|--------|--------|--------|------|---------|--------------|---|
| 1859 | 011444 | 022737 | 000400 | 002050 | | CMP | #400, TEMP1 | |
| 1860 | 011452 | 001353 | | | | BNE | 3\$ | |
| 1861 | 011454 | 104005 | | | | ERROR 5 | | , NO INTERRUPT |
| 1862 | 011456 | 000727 | | | | BR | 7\$ | |
| 1863 | 011460 | 105777 | 170322 | | 2\$ | TSTB | @TADDR | , CLEAR INTERRUPT |
| 1864 | 011464 | 005237 | 002006 | | | INC | TADDR | |
| 1865 | 011470 | 152777 | 000001 | 170330 | | BISB | #RBIT, @CSR | |
| 1866 | 011476 | 105777 | 170304 | | | TSTB | @TADDR | , CLEAR INTERRUPT IN NEXT BYTE |
| 1867 | 011502 | 005337 | 002006 | | | DEC | TADDR | |
| 1868 | 011506 | 132777 | 000200 | 170312 | | BITB | #FBIT, @CSR | , CHECK IF INTERRUPT CLEAR |
| 1869 | 011514 | 001413 | | | | BEQ | 8\$ | |
| 1870 | 011516 | 117737 | 170306 | 002046 | | MOVB | @IAR, TEMP | |
| 1871 | 011524 | 123737 | 002046 | 002006 | | CPMB | TEMP, TADDR | |
| 1872 | 011532 | 001002 | | | | BNE | 9\$ | |
| 1873 | 011534 | 104015 | | | | ERROR | 15 | , RIF BIT IS NOT CLEARING INTERRUPT |
| 1874 | 011536 | 000430 | | | | BR | 5\$ | |
| 1875 | 011540 | 004737 | 012360 | | 9\$ | JSR | PC, CLRINT | , CLEAR ALL REMINDING INTERRUPTS |
| 1876 | 011544 | 132777 | 000010 | 170254 | 8\$ | BITB | #TBIT, @CSR | |
| 1877 | 011552 | 001422 | | | | BEQ | 5\$ | |
| 1878 | 011554 | 012746 | 000000 | | | MOV | #PRO, -(SP) | , SET PSW TO PRIORITY 0 |
| 1879 | 011560 | 012746 | 011566 | | | MOV | #66\$, -(SP) | |
| 1880 | 011564 | 000002 | | | | RTI | | |
| 1881 | 011566 | 000240 | | | 66\$ | NOP | | |
| 1882 | 011570 | 142777 | 000010 | 170230 | | BICB | #TBIT, @CSR | , CHECK IF CLEARING T BIT CAUSE INTERRUPT |
| 1883 | 011576 | 005037 | 001766 | | | CLR | YLOOP | , WAIT |
| 1884 | 011602 | 005237 | 001766 | | 67\$ | INC | YLOOP | |
| 1885 | 011606 | 023727 | 001766 | 177777 | | CMP | YLOOP, #-1 | |
| 1886 | 011614 | 001372 | | | | BNE | 67\$ | |
| 1887 | 011616 | 104005 | | | | ERROR 5 | | , NO INTERRUPT |
| 1888 | 011620 | 010437 | 000004 | | 5\$ | MOV | R4, ERRVEC | |
| 1889 | 011624 | 000644 | | | | BR | 7\$ | |
| 1890 | | | | | | | | |
| 1891 | 011626 | 022626 | | | 4\$ | CMP | (SP)+, (SP)+ | , RESET STACK |
| 1892 | 011630 | 104005 | | | | ERROR 5 | | , NO INTERRUPT |
| 1893 | 011632 | 000772 | | | | BR | 5\$ | |

MAIN MACY11 30(1046) 25-OCT-77 10 10 PAGE 48
DUPLAR P11 25-OCT-77 10 02 DIGITAL MODULE TEST ,M5012

L 4

SEQ 0050

1894 011634 000137 003002 EXERC JMP MONIT

| | | | | | | | | |
|------|--------|--------|--------|--------|-------|---------------------------------|--|---|
| 1895 | 011640 | | | | F5010 | | | |
| 1896 | 011640 | | | | F5011 | | | |
| 1897 | 011640 | 012737 | 000004 | 002004 | MOV | #4,BYTNUM | | ;# OF BYTES PER MODULE |
| 1898 | 011646 | 004737 | 012724 | | JSR | PC,MONDAT | | ;SUBROUTINE TO MONITOP DATA CONTINUOUSLY |
| 1899 | | | | | | | | |
| 1900 | 011652 | | | | F5012 | | | |
| 1901 | 011652 | 012737 | 000002 | 002004 | MOV | #2,BYTNUM | | ;# OF BYTES PER MODULE |
| 1902 | 011660 | 004737 | 012724 | | JSR | PC,MONDAT | | |
| 1903 | | | | | | | | |
| 1904 | 011664 | | | | F5013 | | | |
| 1905 | 011664 | 012737 | 000001 | 002004 | MOV | #1,BYTNUM | | ;# OF BYTES PER MODULE |
| 1906 | 011672 | 004737 | 012724 | | JSR | PC,MONDAT | | |
| 1907 | | | | | | | | |
| 1908 | 011676 | | | | F6010 | | | |
| 1909 | 011676 | 012737 | 000004 | 002004 | MOV | #4,BYTNUM;MODULE IS 4 BYTE LONG | | |
| 1910 | 011704 | 012700 | 000004 | | MOV | #4,RO ,SET A BYTE COUNTER | | |
| 1911 | 011710 | 004737 | 012654 | | JSR | PC,SETPTN | | |
| 1912 | 011714 | 000207 | | | RTS | PC | | |
| 1913 | | | | | | | | |
| 1914 | 011716 | | | | F6012 | | | |
| 1915 | 011716 | | | | F6013 | | | |
| 1916 | 011716 | 012737 | 000001 | 002004 | MOV | #1,BYTNUM | | |
| 1917 | 011724 | 004737 | 012654 | | JSR | PC,SETPTN | | ,ROUTINE TO OUTPUT ANY PATTERN TOOUT MODULE |
| 1918 | 011730 | 000207 | | | RTS | PC | | |
| 1919 | | | | | | | | |
| 1920 | 011732 | | | | F6011 | | | |
| 1921 | 011732 | 012737 | 000J02 | 002004 | MOV | #2,BYTNUM | | ,MODULE IS 2 BYTE LONG |
| 1922 | 011740 | 004737 | 012654 | | JSR | PC,SETPTN | | |
| 1923 | 011744 | 000207 | | | RTC | PC | | |

```

1924          .SBTTL  SUBROUTINES
1925          ,THIS SUBROUTINE IS USED TO CHECK IF
1926          ,A BITE OF DATA IN $GDDAT=$BDDAT
1927          ,IT CHECKS AS MANY BYTES AS SPECIFIED
1928          ,BY      BYTNUM
1929
1930          TSTBYT
1931 011746      MOV      BYTNUM, -(SP)      ;, PUSH BYTNUM ON STACK
1932 011752 013746 002004      MOV      TADDR, -(SP)      ;, PUSH TADDR ON STACK
1933 011756 117737 170024 001126 2$      MOVB     @TADDR, $BDDAT      ;, MOV DATA TO BDDAT
1934 011764 123737 001124 001126      CMPB     $GDDAT, $BDDAT      ;, IS DATA OK
1935 011772 001404      BEQ      1$
1936 011774 013737 002006 002012      MOV      TADDR, TBADDR
1937 012002 104007      ERROR 7
1938 012004 005237 002006 1$      INC      TADDR      ;, NEXT BYTE
1939 012010 005337 002004      DEC      BYTNUM
1940 012014 001360      BNE     2$
1941 012016 012637 002006      MOV     (SP)+, TADDR      ;, POP STACK INTO TADDR
1942 012022 012637 002004      MOV     (SP)+, BYTNUM    ;, POP STACK INTO BYTNUM
1943 012026 000207      RTS      PC
1944
1945          TSTONE
1946 012030      MOV      BYTNUM, -(SP)      ;, PUSH BYTNUM ON STACK
1947 012034 013746 002006      MOV      TADDR, -(SP)      ;, PUSH TADDR ON STACK
1948 012040 117737 167742 001126 2$      MOVB     @TADDR, $BDDAT      ;, MOV DATA TO BDDAT
1949 012046 123737 001124 001126      CMPB     $GDDAT, $BDDAT      ;, IS DATA OK
1950 012054 001404      BEQ      1$
1951 012056 013737 002006 002012      MOV      TADDR, TBADDR
1952 012064 104011      ERROR 11
1953 012066 005237 002006 1$      INC      TADDR      ;, NEXT BYTE
1954 012072 005337 002004      DEC      BYTNUM
1955 012076 001360      BNE     2$
1956 012100 012637 002006      MOV     (SP)+, TADDR      ;, POP STACK INTO TADDR
1957 012104 012637 002004      MOV     (SP)+, BYTNUM    ;, POP STACK INTO BYTNUM
1958 012110 000207      RTS      PC
  
```

```

1959                                     , THIS SUBROUTINE IS USED TO SET AS MANY
1960                                     , BYTES AS SPECIFIED BY BYTNUM IN MUT
1961
1962 012112                               SETBYT
1963 012112 013746 002004                 MOV   BYTNUM, -(SP)      ; , PUSH BYTNUM ON STACK
1964 012116 013746 002006                 MOV   TADDR, -(SP)     ; , PUSH TADDR ON STACK
1965 012122 113777 001124 167656 1$      MOVB  $GDDAT, @TADDR   ; SET DATA IN MUT
1966 012130 005237 002006                 INC   TADDR
1967 012134 005337 002004                 DEC   BYTNUM
1968 012140 001370                         BNE   1$
1969 012142 012637 002006                 MOV   (SP)+, TADDR    ; , POP STACK INTO TADDR
1970 012146 012637 002004                 MOV   (SP)+, BYTNUM   ; , POP STACK INTO BYTNUM
1971 012152 000207                         RTS   PC

```

1972

1973

1974

1975 012154 005004

1976 012156 012703 001700

1977 012162 022301

1978 012164 001404

1979 012166 062704 000006

1980 012172 005713

1981 012174 001372

1982 012176 000207

1983

1984

1985

1986

1987 012200 005004

1988 012202 012703 001700

1989 012206 022301

1990 012210 001404

1991 012212 062704 000002

1992 012216 005713

1993 012220 001372

1994 012222 010022

1995 012224 010125

1996 012226 000207

, THESE TWO SUBROUTINES ARE USED IN MAPPING THE SYSTEM

GENCOD. CLR R4
MOV #GENER, R3
3\$ CMP (R3)+, R1
BEQ 1\$
ADD #6, R4
TST (R3)
BNE 3\$
1\$ RTS PC

TABLE CLR R4
MOV #GENER, R3
3\$ CMP (R3)+, R1
BEQ 1\$
ADD #2, R4
TST (R3)
BNE 3\$
1\$ MOV R0, (R2)+
MOV R1, (R5)+
RTS PC

```

1997                                     ; THIS SUBROUTINE CHECKS IF A BIT GET SET AND CLEAR IN CSR
1998
1999 012230 113777 001124 167570 BITSET MOVB  $GDDAT, @CSR      , LOAD TESTED BIT
2000 012236 005037 001126          CLR  $BDDAT
2001 012242 005037 001766          CLR  YLOOP          , WAIT
2002 012246 005237 001766          INC  YLOOP
2003 012252 023727 001766 000007 64$   CMP  YLOOP, #7
2004 012260 001372          BNE  64$
2005 012262 117737 167540 001126 MOVB  @CSR, $BDDAT      , READ CSR
2006 012270 023727 002022 000005 2$   CMP  TNUMB, #5
2007 012276 001003          BNE  2$
2008 012300 142737 000200 001126 BICB  #FBIT, $BDDAT
2009 012306 123737 001124 001126 2$   CMPB  $GDDAT, $BDDAT    , COMPARE THEME
2010 012314 001002          BNE  1$
2011 012316 062716 000002          ADD  #2, (SP)          , NO ERROR
2012 012322 000337 001124          SWAB $GDDAT
2013 012326 000337 001126          SWAB $BDDAT
2014 012332 042737 177400 001126 BIC  #177400, $BDDAT
2015 012340 042737 177400 001124 BIC  #177400, $GDDAT
2016 012346 000207          RTS  PC
2017
2018                                     ; THIS IS INTERRUPT ROUTINE FOR LINE CLOCK
2019                                     ; WHEN DIAGNOSTIC DOES NOT USE IT
2020
2021 012350          NOCLK
2022 012350 000002          RTI
2023
2024                                     ; THIS IS INTERRUPT ROUTINE FOR CLOCK TO COUNT TICKS
2025
2026 012352 005237 002044          COUNT INC  CLK
2027 012356 000002          RTI
2028

```


MAIN MACY11 30(1046) 25-OCT-77 10 10 PAGE 54
 DVP LPA P11 25-OCT-77 10 02 SUBROUTINES

SEQ 0056

```

2029                                     , THIS SUBROUTINE CLEARS UNEXPECTED INTERRUPTS FROM DBUS
2030 012360 005037 002052                CLRINT CLR    TEMP3
2031 012364 005037 002054                CLR    TEMP4
2032 012370 132777 000200 167430 2$    BITB   #FBIT, @CSR    , IS INTERRUPT PENDING?
2033 012376 001422                        BEQ    1$            , NO
2034 012400 117737 167424 002052        MOVB   @IAR, TEMP3
2035 012406 053737 002042 002052        BIS    BASE, TEMP3
2036 012414 152777 000001 167404        BISB   #RBIT, @CSR
2037 012422 105777 167424                TSTB   @TEMP3
2038 012426 005237 002054                INC    TEMP4
2039 012432 032737 000400 002054        BIT    #BIT8, TEMP4
2040 012440 001753                        BEQ    2$
2041 012442 104023                        ERROR  23            , UNABLE TO CLEAR INTERRUPT
2042 012444 000207                        1$    RTS    PC
2043
2044
2045                                     , THIS SUBROUTINE CHECKS FOR CONTROL C
2046
2047
2048 012446 117746 166474                CNTRC  MOVB   @$TKB, -(SP)
2049 012452 042716 000200                BIC    #BIT7, (SP)    , CLEAR PARITY BIT
2050 012456 122716 000003                CMPB   #3, (SP)      , CONTROL C?
2051 012462 001030                        BNE    1$            , NO
2052 012464 152777 000002 167334        BISB   #CBIT, @CSR    , SET C BIT
2053 012472 005037 001766                CLR    YLOOP          , WAIT
2054 012476 005237 001766 64$          INC    YLOOP
2055 012502 023727 001766 000007        CMP    YLOOP, #7
2056 012510 001372                        BNE    64$
2057 012512 152777 000002 167306        BISB   #CBIT, @CSR    , DO IT AGAIN
2058 012520 005037 001766                CLR    YLOOP          , WAIT
2059 012524 005237 001766 65$          INC    YLOOP
2060 012530 023727 001766 000007        CMP    YLOOP, #7
2061 012536 001372                        BNE    65$
2062 012540 000137 003002                JMP    MONIT
2063 012544 005726                        1$    TST    (SP)+
2064 012546 000207                        RTS    PC
2065
2066
2067 012550 004737 012446                KBINT  JSR    PC, CNTRC
2068 012554 000002                        RTI

```

MAIN MACY11 30(1046) 25-OCT-77 10 10 PAGE 55
 DVP/AR P11 25-OCT-77 10 02 SUBROUTINES

SEQ 0057

```

2069                                     , THIS SUBROUTINE IS USED IN THE LOOP TEST
2070                                     , AND IT IS SIMILAR TO THE TSTBYT ROUTINE
2071
2072 012556                                CHKBYT.
2073 012556 013746 002050                 MOV     TEMP1, -(SP)           , , PUSH TEMP1 ON STACK
2074 012562 013746 002004                 MOV     BYTNUM, -(SP)        , , PUSH BYTNUM ON STACK
2075 012566 117737 167256 001126 2$      MOV     @TEMP1, $BDDAT       , READ DATA FROM INPUT MODULE
2076 012574 123737 001124 001126        CMP     $GDDAT, $BDDAT
2077 012602 001410                         BEQ     1$
2078 012604 013737 002006 002012        MOV     TADDR, TBADDR
2079 012612 012637 002004                 MOV     (SP)+, BYTNUM       , , POP STACK INTO BYTNUM
2080 012616 012637 002050                 MOV     (SP)+, TEMP1       , , POP STACK INTO TEMP1
2081 012622 000207                         RTS     PC                   , ERROR
2082 012624 005237 002050 1$           INC     TEMP1               , GO TO NEXT BYTE
2083 012630 005337 002004                 DEC     BYTNUM
2084 012634 001354                         BNE     2$
2085 012636 012637 002004                 MOV     (SP)+, BYTNUM       , , POP STACK INTO BYTNUM
2086 012642 012637 002050                 MOV     (SP)+, TEMP1       , , POP STACK INTO TEMP1
2087 012646 062716 000002                 ADD     #2, (R6)           , BYPASS ERROR IN MAINLINE CODE
2088 012652 000207                         RTS     PC
2089

```

MAIN MACY11 30(1046) 25-OCT-77 10 10 PAGE 56
 DVPLAR P11 25-OCT-77 10 02 SUBROUTINES

SEQ 0058

```

2090                                     , THIS SUBROUTINE ALLOWS FIELD ENGINEER TO SELECT ANY
2091                                     , DATA PATTERN FOR OUTPUT MODULES
2092
2093
2094 012654 142777 000020 167144 SETPTN BICB   #DBIT, @CSR
2095 012662 142777 000004 167136 BICB   #GBIT, @CSR
2096 012670 104401 021323          1$ TYPE   , MASS19          , SELECT A DATA PATTERN
2097 012674 104410 RDOCT
2098 012676 012637 001124          MOV    (SP)+, $GDDAT      , , POP STACK INTO $GDDAT
2099 012702 113777 001124 167076 MOVB   $GDDAT, @TADDR    , , OUTPUT PATTERN TO MODULE
2100 012710 005237 002006          INC    TADDR
2101 012714 005337 002004          DEC    BYTNUM
2102 012720 001363          BNE   1$
2103 012722 000207          RTS    PC
2104
2105                                     , THIS SUBROUTINE IS USED IN FIELD TEST TO CONTINUOUSLY
2106                                     , MONITOR DATA FROM INPUT MODULE DATA IS PRINTED ONLY
2107                                     , DURING FIRST PASS UNLESS THERE IS A CHANGE IN THE DATA
2108 012724 005001          MONDAT CLR    R1              , CLEAR PASS REGISTER
2109 012726 013700 002004          6$ MOV    BYTNUM, R0      , # OF BYTES PER MODULE
2110 012732 005300          5$ DEC    R0
2111 012734 117737 167046 001124 1$ MOVB   @TADDR, $GDDAT    , STORE DATA
2112 012742 005701          TST   R1              , IS IT FIRST PASS?
2113 012744 001016          BNE   4$              , NO
2114 012746 117760 167034 001770 2$ MOVB   @TADDR, DBUFF(R0)    , STORE DATA IN TABLE
2115 012754 013746 002006          MOV    TADDR, -(SP)     , , SAVE TADDR FOR TYPEOUT
2116 012760 104402          TYPC  , GO TYPE--OCTAL ASCII(ALL DIGITS)
2117 012762 104401 021645          TYPE , MASS27          , COLON & TAB
2118 012766 013746 001124          MOV    $GDDAT, -(SP)   , , SAVE $GDDAT FOR TYPEOUT
2119 012772 104402          TYPC  , GO TYPE--OCTAL ASCII(ALL DIGITS)
2120 012774 104401 020417          TYPE , MASS0           , CR & LF
2121 013000 000404          BR    3$
2122 013002 123760 001124 001770 4$ CMPB   $GDDAT, DBUFF(R0)    , ANY CHANGE IN DATA ?
2123 013010 001356          BNE   2$              , YES
2124 013012 005237 002006          3$ INC    TADDR          , GO TO NEXT BYTE IN MODULE
2125 013016 005700          TST   R0              , LAST BYTE IN MODULE ?
2126 013020 001344          BNE   5$              , NO
2127 013022 163737 002004 002006 SUB    BYTNUM, TADDR     , RESTORE ADDRESS OF MUT
2128 013030 004737 012446          JSR   PC, CNTRC        , TYPE CONTROL C TO RETURN TO MONITOR
2129 013034 012701 000001          MOV    #1, R1         , NEW PASS
2130 013040 000732          BR    6$

```

```

2131
2132                , THIS SUBROUTINE IS USED TO WRAP AROUND DATA "LOOP TEST"
2133
2134
2135 013042          LOPTST
2136 013042 152777 000002 166756  BISB  #CBIT, @CSR  , SET C BIT
2137 013050 005037 001766          CLR  YLOOP  , WAIT
2138 013054 005237 001766          INC  YLOOP
2139 013060 023727 001766 000007 64$  CMP  YLOOP, #7
2140 013066 001372          BNE  64$
2141 013070 152777 000002 166730  BISB  #CBIT, @CSR  ; DO IT AGAIN
2142 013076 005037 001766          CLR  YLOOP  ; WAIT
2143 013102 005237 001766          INC  YLOOP
2144 013106 023727 001766 000007 65$  CMP  YLOOP, #7
2145 013114 001372          BNE  65$
2146 013116 020127 000141          CMP  R1, #141 , IS IT M5010?
2147 013122 001131          BNE  1$      , YES
2148 013124 012737 000125 001124  MOV  #125, $GDDAT
2149 013132 012737 000004 002004  MOV  #4, BYTNUM
2150 013140 004737 012112          JSR  PC, SETBYT  , OUTPUT DATAT PATTERN
2151 013144 005037 001766          CLR  YLOOP  , WAIT
2152 013150 005237 001766          INC  YLOOP
2153 013154 023727 001766 001777 66$  CMP  YLOOP, #1777
2154 013162 001372          BNE  66$
2155 013164 012737 000004 002004  MOV  #4, BYTNUM
2156 013172 004737 012556          JSR  PC, CHKBYT  , COMPARE DATA
2157 013176 104021          ERROR  21
2158 013200 012737 000252 001124  MOV  #252, $GDDAT
2159 013206 012737 000004 002004  MOV  #4, BYTNUM
2160 013214 004737 012112          JSR  PC, SETBYT  , OUTPUT DATAT PATTERN
2161 013220 005037 001766          CLR  YLOOP  , WAIT
2162 013224 005237 001766          INC  YLOOP
2163 013230 023727 001766 001777 67$  CMP  YLOOP, #1777
2164 013236 001372          BNE  67$
2165 013240 012737 000004 002004  MOV  #4, BYTNUM
2166 013246 004737 012556          JSR  PC, CHKBYT  , COMPARE DATA
2167 013252 104021          ERROR  21
2168 013254 012737 000377 001124  MOV  #377, $GDDAT
2169 013262 012737 000004 002004  MOV  #4, BYTNUM
2170 013270 004737 012112          JSR  PC, SETBYT  , OUTPUT DATAT PATTERN
2171 013274 005037 001766          CLR  YLOOP  , WAIT
2172 013300 005237 001766          INC  YLOOP
2173 013304 023727 001766 001777 68$  CMP  YLOOP, #1777
2174 013312 001372          BNE  68$
2175 013314 012737 000004 002004  MOV  #4, BYTNUM
2176 013322 004737 012556          JSR  PC, CHKBYT  , COMPARE DATA
2177 013326 104021          ERROR  21
2178 013330 012737 000000 001124  MOV  #0, $GDDAT
2179 013336 012737 000004 002004  MOV  #4, BYTNUM
2180 013344 004737 012112          JSR  PC, SETBYT  , OUTPUT DATAT PATTERN
2181 013350 005037 001766          CLR  YLOOP  , WAIT
2182 013354 005237 001766          INC  YLOOP
2183 013360 023727 001766 001777 69$  CMP  YLOOP, #1777
2184 013366 001372          BNE  69$
2185 013370 012737 000004 002004  MOV  #4, BYTNUM
2186 013376 004737 012556          JSR  PC, CHKBYT  , COMPARE DATA

```

| | | | | | | | |
|------|--------|--------|--------|--------|-------|---------------|------------------------|
| 2187 | 013402 | 104021 | | | ERROR | 21 | |
| 2188 | 013404 | 000530 | | | BR | 25 | |
| 2189 | 013406 | | | | | | 15 |
| 2190 | 013406 | 012737 | 000125 | 001124 | MOV | #125, \$GDDAT | |
| 2191 | 013414 | 012737 | 000002 | 002004 | MOV | #2, BYTNUM | |
| 2192 | 013422 | 004737 | 012112 | | JSR | PC, SETBYT | , OUTPUT DATAT PATTERN |
| 2193 | 013426 | 005037 | 001766 | | CLR | YLOOP | ; WAIT |
| 2194 | 013432 | 005237 | 001766 | | INC | YLOOP | |
| 2195 | 013436 | 023727 | 001766 | 001777 | CMP | YLOOP, #1777 | |
| 2196 | 013444 | 001372 | | | BNE | 705 | |
| 2197 | 013446 | 012737 | 000002 | 002004 | MOV | #2, BYTNUM | |
| 2198 | 013454 | 004737 | 012556 | | JSR | PC, CHKBYT | ; COMPARE DATA |
| 2199 | 013460 | 104021 | | | ERROR | 21 | |
| 2200 | 013462 | 012737 | 000252 | 001124 | MOV | #252, \$GDDAT | |
| 2201 | 013470 | 012737 | 000002 | 002004 | MOV | #2, BYTNUM | |
| 2202 | 013476 | 004737 | 012112 | | JSR | PC, SETBYT | , OUTPUT DATAT PATTERN |
| 2203 | 013502 | 005037 | 001766 | | CLR | YLOOP | ; WAIT |
| 2204 | 013506 | 005237 | 001766 | | INC | YLOOP | |
| 2205 | 013512 | 023727 | 001766 | 001777 | CMP | YLOOP, #1777 | |
| 2206 | 013520 | 001372 | | | BNE | 715 | |
| 2207 | 013522 | 012737 | 000002 | 002004 | MOV | #2, BYTNUM | |
| 2208 | 013530 | 004737 | 012556 | | JSR | PC, CHKBYT | ; COMPARE DATA |
| 2209 | 013534 | 104021 | | | ERROR | 21 | |
| 2210 | 013536 | 012737 | 000377 | 001124 | MOV | #377, \$GDDAT | |
| 2211 | 013544 | 012737 | 000002 | 002004 | MOV | #2, BYTNUM | |
| 2212 | 013552 | 004737 | 012112 | | JSR | PC, SETBYT | , OUTPUT DATAT PATTERN |
| 2213 | 013556 | 005037 | 001766 | | CLR | YLOOP | ; WAIT |
| 2214 | 013562 | 005237 | 001766 | | INC | YLOOP | |
| 2215 | 013566 | 023727 | 001766 | 001777 | CMP | YLOOP, #1777 | |
| 2216 | 013574 | 001372 | | | BNE | 725 | |
| 2217 | 013576 | 012737 | 000002 | 002004 | MOV | #2, BYTNUM | |
| 2218 | 013604 | 004737 | 012556 | | JSR | PC, CHKBYT | ; COMPARE DATA |
| 2219 | 013610 | 104021 | | | ERROR | 21 | |
| 2220 | 013612 | 012737 | 000000 | 001124 | MOV | #0, \$GDDAT | |
| 2221 | 013620 | 012737 | 000002 | 002004 | MOV | #2, BYTNUM | |
| 2222 | 013626 | 004737 | 012112 | | JSR | PC, SETBYT | , OUTPUT DATAT PATTERN |
| 2223 | 013632 | 005037 | 001766 | | CLR | YLOOP | ; WAIT |
| 2224 | 013636 | 005237 | 001766 | | INC | YLOOP | |
| 2225 | 013642 | 023727 | 001766 | 001777 | CMP | YLOOP, #1777 | |
| 2226 | 013650 | 001372 | | | BNE | 735 | |
| 2227 | 013652 | 012737 | 000002 | 002004 | MOV | #2, BYTNUM | |
| 2228 | 013660 | 004737 | 012556 | | JSR | PC, CHKBYT | ; COMPARE DATA |
| 2229 | 013664 | 104021 | | | ERROR | 21 | |
| 2230 | 013666 | 000207 | | | RTS | PC | 25 |

SBTTL SCOPE HANDLER ROUTINE

```
2231          SBTTL SCOPE HANDLER ROUTINE
2232
2233          , , *****
2234          , *THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
2235          , *AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG (DISPLAY<7 0>)
2236          , *AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15 08>
2237          , *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE
2238          , *SW09=1      LOOP ON ERROR
2239          , *CALL
2240          , *      SCOPE          , , SCOPE=10T
2241
2242 013670      $$SCOPE
2243          , #####START OF CODE FOR THE XOR TESTER#####
2244 013670 000416  $XTSTR BR      6$          , , IF RUNNING ON THE "XOR" TESTER CHANGE
2245          , , THIS INSTRUCTION TO A "NOP" (NOP=240)
2246 013672 013746 000004      MOV      @#ERRVEC, -(SP) , , SAVE THE CONTENTS OF THE ERROR VECTOR
2247 013676 012737 013716 000004      MOV      #5$, @#ERRVEC , , SET FOR TIMEOUT
2248 013704 005737 177060      TST      @#177060 , , TIME OUT ON XOR?
2249 013710 012637 000004      MOV      (SP)+, @#ERRVEC , , RESTORE THE ERROR VECTOR
2250 013714 000421          BR      $$VLAD , , GO TO THE NEXT TEST
2251 013716 022626          5$      CMP      (SP)+, (SP)+ , , CLEAR THE STACK AFTER A TIME OUT
2252 013720 012637 000004      MOV      (SP)+, @#ERRVEC , , RESTORE THE ERROR VECTOR
2253 013724 000407          BR      7$ , , LOOP ON THE PRESENT TEST
2254 013726          6$ , #####END OF CODE FOR THE XOR TESTER#####
2255 013726 105737 001103 2$      TSTB   $ERFLG , , HAS AN ERROR OCCURRED?
2256 013732 001412          BEQ     $$VLAD , , BR IF NO
2257 013734 032777 001000 165176      BIT     #BIT09, @SWR , , LOOP ON ERROR?
2258 013742 001404          BEQ     4$ , , BR IF NO
2259 013744 013737 001110 001106 7$      MOV     $LPERR, $LPADR , , SET LOOP ADDRESS TO LAST SCOPE
2260 013752 000415          BR     $OVER
2261 013754 105037 001103 4$      CLRB   $ERFLG , , ZERO THE ERROR FLAG
2262 013760 105237 001102 5$      $SVLAD INCB $TSTNM , , COUNT TEST NUMBERS
2263 013764 011637 001106      MOV     (SP), $LPADR , , SAVE SCOPE LOOP ADDRESS
2264 013770 011637 001110      MOV     (SP), $LPERR , , SAVE ERROR LOOP ADDRESS
2265 013774 005037 001172      CLR     $ESCAPE , , CLEAR THE ESCAPE FROM ERROR ADDRESS
2266 014000 112737 000001 001115      MOVB   #1, $ERMAX , , ONLY ALLOW ONE(1) ERROR ON NEXT TEST
2267 014006 013777 001102 165126 $OVER  MOV     $TSTNM, @DISPLAY , , DISPLAY TEST NUMBER
2268 014014 013716 001106      MOV     $LPADR, (SP) , , FUDGE RETURN ADDRESS
2269 014020 000002          RTI    , , FIXES PS
2270 014022          $$W08TBL
2271 014022 005416          WORD   TST1+2 , , STARTING ADDRESS OF TEST 1
2272 014024 005550          WORD   TST2+2 , , STARTING ADDRESS OF TEST 2
2273 014026 005614          WORD   TST3+2 , , STARTING ADDRESS OF TEST 3
2274 014030 005660          WORD   TST4+2 , , STARTING ADDRESS OF TEST 4
2275 014032 006054          WORD   TST5+2 , , STARTING ADDRESS OF TEST 5
2276 014034 006124          WORD   TST6+2 , , STARTING ADDRESS OF TEST 6
2277 014036 006170          WORD   TST7+2 , , STARTING ADDRESS OF TEST 7
2278 014040 006234          WORD   TST10+2 , , STARTING ADDRESS OF TEST 10
2279 014042 006400          WORD   TST11+2 , , STARTING ADDRESS OF TEST 11
2280 014044 006674          WORD   TST12+2 , , STARTING ADDRESS OF TEST 12
2281 014046 006776          WORD   TST13+2 , , STARTING ADDRESS OF TEST 13
2282 014050 007442          WORD   TST14+2 , , STARTING ADDRESS OF TEST 14
2283 014052 007634          WORD   TST15+2 , , STARTING ADDRESS OF TEST 15
2284 014054 010026          WORD   TST16+2 , , STARTING ADDRESS OF TEST 16
2285 014056 010324          WORD   TST17+2 , , STARTING ADDRESS OF TEST 17
2286 014060 011154          WORD   TST20+2 , , STARTING ADDRESS OF TEST 20
```

```

2287          SBTTL  TTY INPUT ROUTINE
2288
2289          , , *****
2290          ENABL  LSB
2291
2292          DSABL  LSB
2293
2294
2295          , , *****
2296          *THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
2297          *CALL
2298          *      RDCHR          ;; INPUT A SINGLE CHARACTER FROM THE TTY
2299          *      RETURN HERE   ;; CHARACTER IS ON THE STACK
2300          *                      ;; WITH PARITY BIT STRIPPED OFF
2301          /
2302
2303 014062 011646          SRDCHR  MOV      (SP), -(SP)      ;; PUSH DOWN THE PC
2304 014064 016666 000004 000002  MOV      4(SP), 2(SP)      ;; SAVE THE PS
2305 014072 105777 165046          1$      TSTB      @STKS          ;; WAIT FOR
2306 014076 100375          BPL      1$          ;; A CHARACTER
2307 014100 117766 165042 000004  MOVB     @STKB, 4(SP)      ;; READ THE TTY
2308 014106 042766 177600 000004  BIC      # (177), 4(SP)    ;; GET RID OF JUNK IF ANY
2309 014114 026627 000004 000023  CMP      4(SP), #23       ;; IS IT A CONTROL-S?
2310 014122 001013          BNE      3$          ;; BRANCH IF NO
2311 014124 105777 165014          2$      TSTB      @STKS          ;; WAIT FOR A CHARACTER
2312 014130 100375          BPL      2$          ;; LOOP UNTIL ITS THERE
2313 014132 117746 165010  MOVB     @STKB, -(SP)      ;; GET CHARACTER
2314 014136 042716 177600          BIC      # (177), (SP)    ;; MAKE IT 7-BIT ASCII
2315 014142 022627 000021          CMP      (SP)+, #21       ;; IS IT A CONTROL-Q?
2316 014146 001366          BNE      2$          ;; IF NOT DISCARD IT
2317 014150 000750          BR       1$          ;; YES, RESUME
2318 014152 026627 000004 000140  3$      CMP      4(SP), #140    ;; IS IT UPPER CASE?
2319 014160 002407          BLT      4$          ;; BRANCH IF YES
2320 014162 026627 000004 000175  CMP      4(SP), #175     ;; IS IT A SPECIAL CHAR?
2321 014170 003003          BGT      4$          ;; BRANCH IF YES
2322 014172 042766 000040 000004  BIC      #40, 4(SP)       ;; MAKE IT UPPER CASE
2323 014200 000002          4$      RTI          ;; GO BACK TO USER
2324          , , *****
2325          *THIS ROUTINE WILL INPLT A STRING FROM THE TTY
2326          *CALL
2327          *      RDLIN          ;; INPUT A STRING FROM THE TTY
2328          *      RETURN HERE   ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2329          *                      ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
2330
2331 014202 010346          SRDLIN  MOV      R3, -(SP)      ;; SAVE R3
2332 014204 012703 014310          1$      MOV      #STTYIN, R3    ;; GET ADDRESS
2333 014210 022703 014320          2$      CMP      #STTYIN+8, R3  ;; BUFFER FULL?
2334 014214 101405          BLOS     4$          ;; BR IF YES
2335 014216 104406          RDCHR   ;; GO READ ONE CHARACTER FROM THE TTY
2336 014220 112613          MOVB     (SP)+, (R3)      ;; GET CHARACTER
2337 014222 122713 000177          10$     CMPB     #177, (R3)       ;; IS IT A RUBOUT
2338 014226 001003          BNE      3$          ;; SKIP IF NOT
2339 014230 104401 001174          4$      TYPE     , SQUES     ;; TYPE A '?'
2340 014234 000763          BR       1$          ;; CLEAR THE BUFFER AND LOOP
2341 014236 111337 014306          3$      MOVB     (R3), 9$     ;; ECHO THE CHARACTER
2342 014242 104401 014306          TYPE     , 9$

```

```

2343 014246 122723 000015      CMPB   #15,(R3)+      ;; CHECK FOR RETURN
2344 014252 001356              BNE    2$            ;; LOOP IF NOT RETURN
2345 014254 105063 177777      CLRB   -1(R3)        ;; CLEAR RETURN (THE 15)
2346 014260 104401 001176      TYPE   , $LF         ;; TYPE A LINE FEED
2347 014264 012603              MOV    (SP)+,R3      ;; RESTORE R3
2348 014266 011646              MOV    (SP),-(SP)    ;; ADJUST THE STACK AND PUT ADDRESS OF THE
2349 014270 016666 000004 000002  MOV    4(SP),2(SP)   ;; FIRST ASCII CHARACTER ON IT
2350 014276 012766 014310 000004  MOV    #$TTYIN,4(SP)
2351 014304 000002              RTI                    ;; RETURN
2352 014306      000          9$    BYTE   0                ;; STORAGE FOR ASCII CHAR TO TYPE
2353 014307      000              BYTE   0                ;; TERMINATOR
2354 014310 000010              $TTYIN BLKB   8                ;; RESERVE 8 BYTES FOR TTY INPUT
2355 014320 052536 005015      000  $CNTLU  ASCIZ / U/<15><12>  ;; CONTROL "U"
2356 014325      136 006507 000012  $CNTLG  ASCIZ / G/<15><12>  ;; CONTROL "G"
2357 014332 005015 053523 020122  $MSWR   ASCIZ <15><12>/SWR = /
2358 014340 020075      000
2359 014343      040 047040 053505  $MNEW   ASCIZ / NEW = /
2360 014350 036440 000040
2361          SBTTL  READ AN OCTAL NUMBER FROM THE TTY
2362
2363          ; *****
2364          ; *THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
2365          ; *CHANGE IT TO BINARY.
2366          ; *THE INPUT CHARACTERS WILL BE CHECKED TO INSURED THEY ARE LEGAL
2367          ; *OCTAL DIGITS IF AN ILLEGAL CHARACTER IS READ A "?" WILL BE TYPED
2368          ; *FOLLOWED BY A CARRIAGE RETURN-LINE FEED THE COMPLETE NUMBER MUST
2369          ; *THEN BE RETYPED THE INPUT IS TERMINATED BY TYPING A CARRIAGE RETURN
2370          ; *CALL
2371          ; *      RDOCT                ;; READ AN OCTAL NUMBER
2372          ; *      RETURN HERE          ;; LOW ORDER BITS ARE ON TOP OF THE STACK
2373          ; *                          ;; HIGH ORDER BITS ARE IN $HIOCT
2374
2375 014354 011646              $RDOCT MOV    (SP),-(SP)    ;; PROVIDE SPACE FOR THE
2376 014356 016666 000004 000002  MOV    4(SP),2(SP)   ;; INPUT NUMBER
2377 014364 010046              MOV    R0,-(SP)     ;; PUSH R0 ON STACK
2378 014366 010146              MOV    R1,-(SP)     ;; PUSH R1 ON STACK
2379 014370 010246              MOV    R2,-(SP)     ;; PUSH R2 ON STACK
2380 014372 104407      1$    RDLIN                ;; READ AN ASCII LINE
2381 014374 012600              MOV    (SP)+,R0     ;; GET ADDRESS OF 1ST CHARACTER
2382 014376 010037 014502      MOV    R0,$$        ;; AND SAVE IT
2383 014402 005001              CLR    R1            ;; CLEAR DATA WORD
2384 014404 005002              CLR    R2
2385 014406 112046      2$    MOVB   (R0)+,-(SP)    ;; PICKUP THIS CHARACTER
2386 014410 001420              BEQ   3$            ;; IF ZERO GET OUT
2387 014412 122716 000060      CMPB   #'0,(SP)     ;; MAKE SURE THIS CHARACTER
2388 014416 003026              BGT   4$            ;; IS AN OCTAL DIGIT
2389 014420 122716 000067      CMPB   #'7,(SP)
2390 014424 002423              BLT   4$
2391 014426 006301              ASL   R1              ;; *2
2392 014430 006102              ROL   R2
2393 014432 006301              ASL   R1              ;; *4
2394 014434 006102              ROL   R2
2395 014436 006301              ASL   R1              ;; *8
2396 014440 006102              ROL   R2
2397 014442 042716 177770      BIC   #(7,(SP)     ;; STRIP THE ASCII JUNK
2398 014446 062601              ADD   (SP)+,R1      ;; ADD IN THIS DIGIT

```



```

2399 014450 000756          BR      2$          ;; LOOP
2400 014452 005726          3$      TST      (SP)+        ;; CLEAN TERMINATOR FROM STACK
2401 014454 010166 000012  MOV      R1,12(SP)    ;; SAVE THE RESULT
2402 014460 010237 014512  MOV      R2,$HIOCT
2403 014464 012602          MOV      (SP)+,R2    ;; POP STACK INTO R2
2404 014466 012601          MOV      (SP)+,R1    ;; POP STACK INTO R1
2405 014470 012600          MOV      (SP)+,R0    ;; POP STACK INTO R0
2406 014472 000002          RTI                    ;; RETURN
2407 014474 005726          4$      TST      (SP)+        ;; CLEAN PARTIAL FROM STACK
2408 014476 105010          CLR      (R0)        ;; SET A TERMINATOR
2409 014500 104401          TYPE                    ;; TYPE UP THRU THE BAD CHAR.
2410 014502 000000          5$      WORD      0
2411 014504 104401 001174  TYPE      , $QUES    ;; "?" "CR" & "LF"
2412 014510 000730          BR      1$          ;; TRY AGAIN
2413 014512 000000  $HIOCT WORD      0    ;; HIGH ORDER BITS GO HERE
2414          SBTTL  ERROR HANDLER ROUTINE
2415
2416          ;; *****
2417          ;; *THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
2418          ;; *SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
2419          ;; *AND GO TO $ERRTYP ON ERROR
2420          ;; *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE
2421          ;; *SW15=1      HALT ON ERROR
2422          ;; *SW13=1      INHIBIT ERROR TYPEOUTS
2423          ;; *SW09=1     LOOP ON ERROR
2424          ;; *CALL
2425          ;; *      ERROR  N      ;; ERROR=EMT AND N=ERROR ITEM NUMBER
2426
2427          $ERROR
2428 014514 105237 001103  7$      INCB     $ERFLG    ;; SET THE ERROR FLAG
2429 014520 001775          BEQ      7$          ;; DON'T LET THE FLAG GO TO ZERO
2430 014522 013777 001102 164412  MOV      $TSTNM,@DISPLAY ;; DISPLAY TEST NUMBER AND ERROR FLAG
2431 014530 005237 001112          INC      $ERTTL    ;; INC THE ERROR COUNT
2432 014534 011637 001116  MOV      (SP), $ERRPC ;; GET ADDRESS OF ERROR INSTRUCTION
2433 014540 162737 000002 001116  SUB      #2, $ERRPC
2434 014546 117737 164344 001114  MOVB    @ $ERRPC, $ITEMB ;; STRIP AND SAVE THE ERROR ITEM CODE
2435 014554 032777 020000 164356  BIT     #BIT13, @SWR   ;; SKIP TYPEOUT IF SET
2436 014562 001004          BNE     20$          ;; SKIP TYPEOUTS
2437 014564 004737 014646  JSR     PC, $ERRTYP   ;; GO TO USER ERROR ROUTINE
2438 014570 104401 001175  TYPE    , $CRLF
2439 014574          20$
2440 014574 005777 164340  2$      TST      @SWR        ;; HALT ON ERROR
2441 014600 100001          BPL     3$          ;; SKIP IF CONTINUE
2442 014602 000000          HALT                    ;; HALT ON ERROR!
2443 014604 032777 001000 164326  3$      BIT     #BIT09, @SWR ;; LOOP ON ERROR SWITCH SET?
2444 014612 001402          BEQ     4$          ;; BR IF NO
2445 014614 013716 001110  MOV     $LPERR, (SP)   ;; FUDGE RETURN FOR LOOPING
2446 014620 005737 001172  4$      TST     $ESCAPE     ;; CHECK FOR AN ESCAPE ADDRESS
2447 014624 001402          BEQ     5$          ;; BR IF NONE
2448 014626 013716 001172  MOV     $ESCAPE, (SP) ;; FUDGE RETURN ADDRESS FOR ESCAPE
2449 014632          5$
2450 014632 022737 004300 000042  CMP     # $SENDAD, @#42 ;; ACT-11 AUTO-ACCEPT?
2451 014640 001001          BNE     6$          ;; BRANCH IF NO
2452 014642 000000          HALT                    ;; YES
2453 014644          6$
2454 014644 000002          RTI                    ;; RETURN
  
```

```
2455 .SBTTL ERROR MESSAGE TYPEOUT ROUTINE
2456
2457 ; ;*****
2458 ; *THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
2459 ; *ERROR IS TO BE REPORTED IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
2460 ; *AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR
2461
2462 $ERRTYP
2463 014646 104401 001175 TYPE , $CRLF ; "CARRIAGE RETURN" & "LINE FEED"
2464 014652 010046 MOV RO, -(SP) ; SAVE RO
2465 014654 005000 CLR RO ; PICKUP THE ITEM INDEX
2466 014656 153700 001114 BLSB @#$ITEMB, RO
2467 014662 001004 BNE 1$ ; IF ITEM NUMBER IS ZERO, JUST
2468 ; TYPE THE PC OF THE ERROR
2469 014664 013746 001116 MOV $ERRPC, -(SP) ; SAVE $ERRPC FOR TYPEOUT
2470 ; ERROR ADDRESS
2471 014670 104402 TYPOC ; GO TYPE--OCTAL ASCII(ALL DIGITS)
2472 014672 000426 BR 6$ ; GET OUT
2473 014674 005300 1$ DEC RO ; ADJUST THE INDEX SO THAT IT WILL
2474 014676 006300 ASL RO ; WORK FOR THE ERROR TABLE
2475 014700 006300 ASL RO
2476 014702 006300 ASL RO
2477 014704 062700 002112 ADD #$ERRTB, RO ; FORM TABLE POINTER
2478 014710 012037 014720 MOV (RO)+, 2$ ; PICKUP "ERROR MESSAGE" POINTER
2479 014714 001404 BEQ 3$ ; SKIP TYPEOUT IF NO POINTER
2480 014716 104401 TYPE ; TYPE THE "ERROR MESSAGE"
2481 014720 000000 2$ WORD 0 ; "ERROR MESSAGE" POINTER GOES HERE
2482 014722 104401 001175 TYPE , $CRLF ; "CARRIAGE RETURN" & "LINE FEED"
2483 014726 012037 014736 3$ MOV (RO)+, 4$ ; PICKUP "DATA HEADER" POINTER
2484 014732 001404 BEQ 5$ ; SKIP TYPEOUT IF 0
2485 014734 104401 TYPE ; TYPE THE "DATA HEADER"
2486 014736 000000 4$ WORD 0 ; "DATA HEADER" POINTER GOES HERE
2487 014740 104401 001175 TYPE , $CRLF ; "CARRIAGE RETURN" & "LINE FEED"
2488 014744 011000 5$ MOV (RO), RO ; PICKUP "DATA TABLE" POINTER
2489 014746 001004 BNE 7$ ; GO TYPE THE DATA
2490 014750 012600 6$ MOV (SP)+, RO ; RESTORE RO
2491 014752 104401 001175 TYPE , $CRLF ; "CARRIAGE RETURN" & "LINE FEED"
2492 014756 000207 7$ RTS PC ; RETURN
2493 014760
2494 014760 013046 MOV @ (RO)+, -(SP) ; SAVE @ (RO)+ FOR TYPEOUT
2495 014762 104402 TYPOC ; GO TYPE--OCTAL ASCII(ALL DIGITS)
2496 014764 005710 TST (RO) ; IS THERE ANOTHER NUMBER?
2497 014766 001770 BEQ 6$ ; BR IF NO
2498 014770 104401 014776 TYPE , 8$ ; TYPE TWO(2) SPACES
2499 014774 000771 BR 7$ ; LOOP
2500 014776 020040 000 8$ ASCIZ / / ; TWO(2) SPACES
2501 015002 .EVEN
2502 .SBTTL TYPE ROUTINE
2503
2504 ; ;*****
2505 ; *ROUTINE TO TYPE ASCIZ MESSAGE MESSAGE MUST TERMINATE WITH A 0 BYTE
2506 ; *THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED
2507 ; *NOTE1 $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER
2508 ; *NOTE2 $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED
2509 ; *NOTE3 $FILLC CONTAINS THE CHARACTER TO FILL AFTER
2510 ; *
```

```

2511 ;*CALL.
2512 ;*1) USING A TRAP INSTRUCTION
2513 ;* TYPE MESADR ; MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
2514 ;*OR
2515 ;* TY E
2516 ;* MESADR
2517 ;*
2518
2519 015002 105737 001157 $TYPE TSTB $TPFLG ; IS THERE A TERMINAL?
2520 015006 100002 BPL 1$ ; BR IF YES
2521 015010 000000 HALT ; HALT HERE IF NO TERMINAL
2522 015012 000407 BR 3$ ; LEAVE
2523 015014 010046 1$ MOV RO, -(SP) ; SAVE RO
2524 015016 017600 000002 MOV @2(SP), RO ; GET ADDRESS OF ASCIZ STRING
2525 015022 112046 2$ MOVB (RO)+, -(SP) ; PUSH CHARACTER TO BE TYPED ONTO STACK
2526 015024 001005 BNE 4$ ; BR IF IT ISN'T THE TERMINATOR
2527 015026 005726 TST (SP)+ ; IF TERMINATOR POP IT OFF THE STACK
2528 015030 012600 60$ MOV (SP)+, RO ; RESTORE RO
2529 015032 062716 3$ ADD #2, (SP) ; ADJUST RETURN PC
2530 015036 000002 RTI ; RETURN
2531 015040 122716 000011 4$ CMPB #HT, (SP) ; BRANCH IF <HT>
2532 015044 001430 BEQ 8$
2533 015046 122716 000200 CMPB #CRLF, (SP) ; BRANCH IF NOT <CRLF>
2534 015052 001006 BNE 5$
2535 015054 005726 TST (SP)+ ; POP <CR><LF> EQUIV
2536 015056 104401 TYPE ; TYPE A CR AND LF
2537 015060 001175 $CRLF
2538 015062 105037 015216 CLRB $CHARCNT ; CLEAR CHARACTER COUNT
2539 015066 000755 BR 2$ ; GET NEXT CHARACTER
2540 015070 004737 015152 5$ JSR PC, $TYPEC ; GO TYPE THIS CHARACTER
2541 015074 123726 001156 6$ CMPB $FILLC, (SP)+ ; IS IT TIME FOR FILLER CHARS ?
2542 015100 001350 BNE 2$ ; IF NO GO GET NEXT CHAR
2543 015102 013746 001154 MOV $NULL, -(SP) ; GET # OF FILLER CHARS NEEDED
2544 ; AND THE NULL CHAR
2545 015106 105366 000001 7$ DECB 1(SP) ; DOES A NULL NEED TO BE TYPED?
2546 015112 002770 BLT 6$ ; BR IF NO--GO POP THE NULL OFF OF STACK
2547 015114 004737 015152 JSR PC, $TYPEC ; GO TYPE A NULL
2548 015120 105337 015216 DECB $CHARCNT ; DO NOT COUNT AS A COUNT
2549 015124 000770 BR 7$ ; LOOP
2550
2551 .HORIZONTAL TAB PROCESSOR
2552
2553 015126 112716 000040 8$ MOVB #' , (SP) ; REPLACE TAB WITH SPACE
2554 015132 004737 015152 9$ JSR PC, $TYPEC ; TYPE A SPACE
2555 015136 132737 000007 015216 BITB #7, $CHARCNT ; BRANCH IF NOT AT
2556 015144 001372 BNE 9$ ; TAB STOP
2557 015146 005726 TST (SP)+ ; POP SPACE OFF STACK
2558 015150 000724 BR 2$ ; GET NEXT CHARACTER
2559 015152 105777 163772 $TYPEC. TSTB @STPS ; WAIT UNTIL PRINTER IS READY
2560 015156 100375 BPL $TYPEC
2561 015160 116677 000002 163764 MOVB 2(SP), @STPB ; LOAD CHAR TO BE TYPED INTO DATA REG
2562 015166 122766 000015 000002 CMPB #CR, 2(SP) ; IS CHARACTER A CARRIAGE RETURN?
2563 015174 001003 BNE 1$ ; BRANCH IF NO
2564 015176 105037 015216 CLRB $CHARCNT ; YES--CLEAR CHARACTER COUNT
2565 015202 000406 BR $TYPEX ; EXIT
2566 015204 122766 000012 000002 1$ CMPB #LF, 2(SP) ; IS CHARACTER A LINE FEED?

```

```

2567 015212 001402          BEQ     $TYPEX      ;; BRANCH IF YES
2568 015214 105227          INCB    (PC)+        ;; COUNT THE CHARACTER
2569 015216 000000          $CHARCNT WORD 0      ;; CHARACTER COUNT STORAGE
2570 015220 000207          $TYPEX. RTS     PC
2571
2572          SBTTL   CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
2573
2574          ;; *****
2575          ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
2576          ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
2577          ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
2578          ;*BEFORE THE FIRST DIGIT OF THE NUMBER LEADING ZEROS WILL ALWAYS BE
2579          ;*REPLACED WITH SPACES
2580          ;*CALL
2581          ;*      MOV     NUM, -(SP)      ;; PUT THE BINARY NUMBER ON THE STACK
2582          ;*      TYPDS      ;; GO TO THE ROUTINE
2583
2584          $TYPDS
2585          015222 010046      MOV     R0, -(SP)      ;; PUSH R0 ON STACK
2586          015224 010146      MOV     R1, -(SP)      ;; PUSH R1 ON STACK
2587          015226 010246      MOV     R2, -(SP)      ;; PUSH R2 ON STACK
2588          015230 010346      MOV     R3, -(SP)      ;; PUSH R3 ON STACK
2589          015232 010546      MOV     R5, -(SP)      ;; PUSH R5 ON STACK
2590          015234 012746 020200  MOV     #20200, -(SP)  ;; SET BLANK SWITCH AND SIGN
2591          015240 016605 000020  MOV     20(SP), R5     ;; GET THE INPUT NUMBER
2592          015244 100004      BPL     1$            ;; BR IF INPUT IS POS
2593          015246 005405      NEG     R5            ;; MAKE THE BINARY NUMBER POS
2594          015250 112766 000055 000001  MOVB    #'-, 1(SP)    ;; MAKE THE ASCII NUMBER NEG
2595          015256 005000          1$      CLR     R0            ;; ZERO THE CONSTANTS INDEX
2596          015260 012703 015436      MOV     #$DBLK, R3    ;; SETUP THE OUTPUT POINTER
2597          015264 112723 000040      MOVB    #' , (R3)+    ;; SET THE FIRST CHARACTER TO A BLANK
2598          015270 005002          2$      CLR     R2            ;; CLEAR THE BCD NUMBER
2599          015272 016001 015426      MOV     $DTBL(R0), R1 ;; GET THE CONSTANT
2600          015276 160105          3$      SUB     R1, R5        ;; FORM THIS BCD DIGIT
2601          015300 002402          BLT     4$            ;; BR IF DONE
2602          015302 005202          INC     R2            ;; INCREASE THE BCD DIGIT BY 1
2603          015304 000774          BR     3$
2604          015306 060105          4$      ADD     R1, R5        ;; ADD BACK THE CONSTANT
2605          015310 005702          TST     R2            ;; CHECK IF BCD DIGIT=0
2606          015312 001002          BNE     5$            ;; FALL THROUGH IF 0
2607          015314 105716          TSTB   (SP)          ;; STILL DOING LEADING 0'S?
2608          015316 100407          BMI     7$            ;; BR IF YES
2609          015320 106316          5$      ASLB   (SP)          ;; MSD?
2610          015322 103003          BCC     6$            ;; BR IF NO
2611          015324 116663 000001 177777  MOVB    1(SP), -1(R3)  ;; YES--SET THE SIGN
2612          015332 052702 000060          6$      BIS     #'0, R2      ;; MAKE THE BCD DIGIT ASCII
2613          015336 052702 000040          7$      BIS     #' , R2      ;; MAKE IT A SPACE IF NOT ALREADY A DIGIT
2614          015342 110223          MOVB   R2, (R3)+     ;; PUT THIS CHARACTER IN THE OUTPUT BUFFER
2615          015344 005720          TST    (R0)+        ;; JUST INCREMENTING
2616          015346 020027 000010          CMP     R0, #10      ;; CHECK THE TABLE INDEX
2617          015352 002746          BLT    2$            ;; GO DO THE NEXT DIGIT
2618          015354 003002          BGT    8$            ;; GO TO EXIT
2619          015356 010502          MOV    R5, R2        ;; GET THE LSD
2620          015360 000764          BR     6$            ;; GO CHANGE TO ASCII
2621          015362 105726          8$      TSTB   (SP)+        ;; WAS THE LSD THE FIRST NON-ZERO?
2622          015364 100003          BPL    9$            ;; BR IF NO

```

MAIN MACY11 30(1046) 25-OCT-77 10 10 PAGE 66
 DVPCLAR P11 25-OCT-77 10 02

CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

SEQ 0068

```

2623 015366 116663 177777 177776      MOVB   -1(SP),-2(R3)  ;;YES--SET THE SIGN FOR TYPING
2624 015374 105013                    9$    CLRB   (R3)        ;;SET THE TERMINATOR
2625 015376 012605                    MOV    (SP)+,R5      ;;POP STACK INTO R5
2626 015400 012603                    MOV    (SP)+,R3      ;;POP STACK INTO R3
2627 015402 012602                    MOV    (SP)+,R2      ;;POP STACK INTO R2
2628 015404 012601                    MOV    (SP)+,R1      ;;POP STACK INTO R1
2629 015406 012600                    MOV    (SP)+,R0      ;;POP STACK INTO R0
2630 015410 104401 015436              TYPE   ,SDBLK        ;;NOW TYPE THE NUMBER
2631 015414 016666 000002 000004      MOV    2(SP),4(SP)   ;;ADJUST THE STACK
2632 015422 012616                    MOV    (SP)+,(SP)
2633 015424 000002                    RTI
2634 015426 023420                    SOTBL 10000
2635 015430 001750                    1000
2636 015432 000144                    100.
2637 015434 000012                    10
2638 015436 000004                    SDBLK  5LKW  4
2639                                SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
2640
2641                                ;;*****
2642                                ;;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
2643                                ;;*OCTAL (ASCII) NUMBER AND TYPE IT
2644                                ;;*STYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
2645                                ;;*CALL
2646                                ;;*   MOV    NUM,-(SP)      ;;NUMBER TO BE TYPED
2647                                ;;*   TYPOS      ;;CALL FOR TYPEOUT
2648                                ;;*   BYTE   N          ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
2649                                ;;*   BYTE   M          ;;M=1 OR 0
2650                                ;;*                               ;;1=TYPE LEADING ZEROS
2651                                ;;*                               ;;0=SUPPRESS LEADING ZEROS
2652                                ;;*
2653                                ;;*STYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
2654                                ;;*STYPOS OR STYPOC
2655                                ;;*CALL
2656                                ;;*   MOV    NUM,-(SP)      ;;NUMBER TO BE TYPED
2657                                ;;*   TYPON      ;;CALL FOR TYPEOUT
2658                                ;;*
2659                                ;;*STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
2660                                ;;*CALL
2661                                ;;*   MOV    NUM,-(SP)      ;;NUMBER TO BE TYPED
2662                                ;;*   TYPOC      ;;CALL FOR TYPEOUT
2663
2664 015446 017646 000000                STYPOS MOV    2(SP),-(SP)  ;;PICKUP THE MODE
2665 015452 116637 000001 015671      MOVB   1(SP),%DFILL  ;;LOAD ZERO FILL SWITCH
2666 015460 112637 015673              MOVB   (SP)+,%MODE+1 ;;NUMBER OF DIGITS TO TYPE
2667 015464 062716 000002              ADD    #2,(SP)      ;;ADJUST RETURN ADDRESS
2668 015470 000406                    BR     STYPON
2669 015472 112737 000001 015671      STYPOC MOVB   #1,%DFILL  ;;SET THE ZERO FILL SWITCH
2670 015500 112737 000006 015673      MOVB   #6,%MODE+1   ;;SET FOR SIX(6) DIGITS
2671 015506 112737 000005 015670      STYPON MOVB   #5,%OCNT   ;;SET THE ITERATION COUNT
2672 015514 010346                    MOV    R3,-(SP)     ;;SAVE R3
2673 015516 010446                    MOV    R4,-(SP)     ;;SAVE R4
2674 015520 010546                    MOV    R5,-(SP)     ;;SAVE R5
2675 015522 113704 015673              MOVB   %MODE+1,R4   ;;GET THE NUMBER OF DIGITS TO TYPE
2676 015526 005404                    NEG    R4
2677 015530 062704 000006              ADD    #6,R4        ;;SUBTRACT IT FOR MAX ALLOWED
2678 015534 110437 015672              MOVB   R4,%MODE     ;;SAVE IT FOR USE

```

```

2679 015540 113704 015671      MOV      $OFILL,R4      ;; GET THE ZERO FILL SWITCH
2680 015544 016605 000012      MOV      12(SP),R5     ;; PICKUP THE INPUT NUMBER
2681 015550 005003              CLR      R3            ;; CLEAR THE OUTPUT WORD
2682 015552 006105      1$     ROL      R5            ;; ROTATE MSB INTO "C"
2683 015554 000404              BR      3$            ;; GO DO MSB
2684 015556 006105      2$     ROL      R5            ;; FORM THIS DIGIT
2685 015560 006105              ROL      R5
2686 015562 006105              ROL      R5
2687 015564 010503              MOV      R5,R3
2688 015566 006103      3$     ROL      R3            ;; GET LSB OF THIS DIGIT
2689 015570 105337 015672      DECB    $OMODE         ;; TYPE THIS DIGIT?
2690 015574 100016              BPL     7$            ;; BR IF NO
2691 015576 042703 177770      BIC     #177770,R3     ;; GET RID OF JUNK
2692 015602 001002              BNE     4$            ;; TEST FOR 0
2693 015604 005704              TST     R4            ;; SUPPRESS THIS 0?
2694 015606 001403              BEQ     5$            ;; BR IF YES
2695 015610 005204      4$     INC      R4            ;; DON'T SUPPRESS ANYMORE 0'S
2696 015612 052703 000060      BIS     #'0,R3         ;; MAKE THIS DIGIT ASCII
2697 015616 052703 000040      5$     BIS     #' ,R3     ;; MAKE ASCII IF NOT ALREADY
2698 015622 110337 015666      MOV     R3,8$         ;; SAVE FOR TYPING
2699 015626 104401 015666      TYPE   ,8$           ;; GO TYPE THIS DIGIT
2700 015632 105337 015670      7$     DECB    $OCNT     ;; COUNT BY 1
2701 015636 003347              BGT     2$            ;; BR IF MORE TO DO
2702 015640 002402              BLT     6$            ;; BR IF DONE
2703 015642 005204              INC     R4            ;; INSURE LAST DIGIT ISN'T A BLANK
2704 015644 000744              BR      2$            ;; GO DO THE LAST DIGIT
2705 015646 012605      6$     MOV     (SP)+,R5     ;; RESTORE R5
2706 015650 012604              MOV     (SP)+,R4     ;; RESTORE R4
2707 015652 012603              MOV     (SP)+,R3     ;; RESTORE R3
2708 015654 016666 000002 000004      MOV     2(SP),4(SP)   ;; SET THE STACK FOR RETURNING
2709 015662 012616              MOV     (SP)+,(SP)
2710 015664 000002              RTI
2711 015666      000      8$     BYTE   0            ;; STORAGE FOR ASCII DIGIT
2712 015667      000              BYTE   0            ;; TERMINATOR FOR TYPE ROUTINE
2713 015670      000      $OCNT  BYTE   0            ;; OCTAL DIGIT COUNTER
2714 015671      000      $OFILL  BYTE   0            ;; ZERO FILL SWITCH
2715 015672 000000      $OMODE  WORD    0            ;; NUMBER OF DIGITS TO TYPE
2716              SBTTL  SAVE AND RESTORE R0-R5 ROUTINES
2717
2718              ;, *****
2719              ;*SAVE R0-R5
2720              ;*CALL
2721              ;* SAVREG
2722              ;*UPON RETURN FROM $SAVREG THE STACK WILL LOOK LIKE
2723              ;*
2724              ;*TOP---(+16)
2725              ;* +2---(+18)
2726              ;* +4---R5
2727              ;* +6---R4
2728              ;* +8---R3
2729              ;*+10---R2
2730              ;*+12---R1
2731              ;*+14---R0
2732
2733 015674      $SAVREG
2734 015674 010046      MOV     R0,-(SP)     ;; PUSH R0 ON STACK

```

```

2735 015676 010146      MOV      R1,-(SP)      ;; PUSH R1 ON STACK
2736 015700 010246      MOV      R2,-(SP)      ;; PUSH R2 ON STACK
2737 015702 010346      MOV      R3,-(SP)      ;; PUSH R3 ON STACK
2738 015704 010446      MOV      R4,-(SP)      ;; PUSH R4 ON STACK
2739 015706 010546      MOV      R5,-(SP)      ;; PUSH R5 ON STACK
2740 015710 016646 000022  MOV      22(SP),-(SP)  ;; SAVE PS OF MAIN FLOW
2741 015714 016646 000022  MOV      22(SP),-(SP)  ;; SAVE PC OF MAIN FLOW
2742 015720 016646 000022  MOV      22(SP),-(SP)  ;; SAVE PS OF CALL
2743 015724 016646 000022  MOV      22(SP),-(SP)  ;; SAVE PC OF CALL
2744 015730 000002      RTI

```

```

2745
2746      ;*RESTORE RO-R5
2747      ;*CALL
2748      ;*
2749      $RESREG RESREG

```

```

2750 015732 012666 000022  MOV      (SP)+,22(SP)  ;; RESTORE PC OF CALL
2751 015736 012666 000022  MOV      (SP)+,22(SP)  ;; RESTORE PS OF CALL
2752 015742 012666 000022  MOV      (SP)+,22(SP)  ;; RESTORE PC OF MAIN FLOW
2753 015746 012666 000022  MOV      (SP)+,22(SP)  ;; RESTORE PS OF MAIN FLOW
2754 015752 012605      MOV      (SP)+,R5     ;; POP STACK INTO R5
2755 015754 012604      MOV      (SP)+,R4     ;; POP STACK INTO R4
2756 015756 012603      MOV      (SP)+,R3     ;; POP STACK INTO R3
2757 015760 012602      MOV      (SP)+,R2     ;; POP STACK INTO R2
2758 015762 012601      MOV      (SP)+,R1     ;; POP STACK INTO R1
2759 015764 012600      MOV      (SP)+,R0     ;; POP STACK INTO R0
2760 015766 000002      RTI

```

SBTTL TRAP DECODER

```

2761
2762
2763      ;*****
2764      ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
2765      ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
2766      ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
2767      ;*GO TO THAT ROUTINE

```

```

2768
2769 015770 010046      $TRAP  MOV      RO,-(SP)  ;; SAVE RO
2770 015772 016600 000002  MOV      2(SP),RO     ;; GET TRAP ADDRESS
2771 015776 005740      TST      -(RO)        ;; BACKUP BY 2
2772 016000 111000      MOV      (RO),RO     ;; GET RIGHT BYTE OF TRAP
2773 016002 006300      ASL      RO          ;; POSITION FOR INDEXING
2774 016004 016000 016024  MOV      $TRPAD(RO),RO ;; INDEX TO TABLE
2775 016010 000200      RTS      RO          ;; GO TO ROUTINE

```

```

2776
2777
2778      ;; THIS IS USE TO HANDLE THE "GETPRI" MACRO
2779

```

```

2780 016012 011646      $TRAP2 MOV      (SP),-(SP)  ;; MOVE THE PC DOWN
2781 016014 016666 000004 000002  MOV      4(SP),2(SP)  ;; MOVE THE PSW DOWN
2782 016022 000002      RTI                ;; RESTORE THE PSW

```

SBTTL TRAP TABLE

```

2783
2784
2785      ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
2786      ;*BY THE "TRAP" INSTRUCTION

```

```

2787
2788      ROUTINE
2789      -----
2790

```

```

2791 016024 016012 STRPAD .WORD STRAP2
2792 016026 015002 $TYPE ;CALL=TYPE TRAP+1(104401) TTY TYPEOUT ROUTINE
2793 016030 015472 $TYPOC ;CALL=TYPOC TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
2794 016032 015446 $TYPOS ;CALL=TYPOS TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
2795 016034 015506 $TYPON ;CALL=TYPON TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
2796 016036 015222 $TYPDS ;CALL=TYPDS TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
2797
2798
2799 016040 014062 $RDCHR ;CALL=RDCHR TRAP+6(104406) TTY TYPEIN CHARACTER ROUTINE
2800 016042 014202 $RDLIN ;CALL=RDLIN TRAP+7(104407) TTY TYPEIN STRING ROUTINE
2801 016044 014354 $RDOCT ;CALL=RDOCT TRAP+10(104410) READ AN OCTAL NUMBER FROM TTY
2802 016046 015674 $SAVREG ;CALL=SAVREG TRAP+11(104411) SAVE R0-R5 ROUTINE
2803 016050 015732 $RESREG ;CALL=RESREG TRAP+12(104412) RESTORE R0-R5 ROUTINE
2804 SBTTL POWER DOWN AND UP ROUTINES
2805
2806 ,,*****
2807 ,POWER DOWN ROUTINE
2808 016052 012737 016212 000024 SPWRDN MOV $SILLUP,@#PWRVEC ;SET FOR FAST UP
2809 016060 012737 000340 000026 MOV #340,@#PWRVEC+2 ;PRIO 7
2810 016066 010046 MOV R0,-(SP) ;PUSH R0 ON STACK
2811 016070 010146 MOV R1,-(SP) ;PUSH R1 ON STACK
2812 016072 010246 MOV R2,-(SP) ;PUSH R2 ON STACK
2813 016074 010346 MOV R3,-(SP) ;PUSH R3 ON STACK
2814 016076 010446 MOV R4,-(SP) ;PUSH R4 ON STACK
2815 016100 010546 MOV R5,-(SP) ;PUSH R5 ON STACK
2816 016102 017746 163032 MOV @SWR,-(SP) ;PUSH @SWR ON STACK
2817 016106 010637 016216 MOV SP,$SAVR6 ;SAVE SP
2818 016112 012737 016124 000024 MOV #SPWRUP,@#PWRVEC ;SET UP VECTOR
2819 016120 000000 HALT
2820 016122 000776 BR -2 ;HANG UP
2821
2822 ,,*****
2823 ,POWER UP ROUTINE
2824 016124 012737 016212 000024 SPWRUP MOV $SILLUP,@#PWRVEC ;SET FOR FAST DOWN
2825 016132 013706 016216 MOV $SAVR6,SP ;GET SP
2826 016136 005037 016216 CLR $SAVR6 ;WAIT LOOF FOR THE TTY
2827 016142 005237 016216 15 INC $SAVR6 ;WAIT FOR THE INC
2828 016146 001375 BNE 15 ;OF WORD
2829 016150 012677 162764 MOV (SP)+,@SWR ;POP STACK INTO @SWR
2830 016154 012605 MOV (SP)+,R5 ;POP STACK INTO R5
2831 016156 012604 MOV (SP)+,R4 ;POP STACK INTO R4
2832 016160 012603 MOV (SP)+,R3 ;POP STACK INTO R3
2833 016162 012602 MOV (SP)+,R2 ;POP STACK INTO R2
2834 016164 012601 MOV (SP)+,R1 ;POP STACK INTO R1
2835 016166 012600 MOV (SP)+,R0 ;POP STACK INTO R0
2836 016170 012737 016052 000024 MOV #SPWRDN,@#PWRVEC ;SET UP THE POWER DOWN VECTOR
2837 016176 012737 000340 000026 MOV #340,@#PWRVEC+2 ;PRIO 7
2838 016204 104401 TYPE ;REPORT THE POWER FAILURE
2839 016206 016220 SPWRMG .WORD $POWER ;POWER FAIL MESSAGE POINTER
2840 016210 000002 RTI
2841 016212 000000 SILLUP HALT ;THE POWER UP SEQUENCE WAS STARTED
2842 016214 000776 BR -2 ;BEFORE THE POWER DOWN WAS COMPLETE
2843 016216 000000 $SAVR6 0 ;PUT THE SP HERE
2844 016220 005015 047520 042527 $POWER ASCII <15><12>"POWER"
2845 016226 000122
2846 EVEN

```


MAIN MACY11 30(1046) 25-OCT-77 10 10 PAGE 70
 DUPLCA P11 25-OCT-77 10 02 POWER DOWN AND UP ROUTINES

SEQ 0072

| | | | | | | | |
|------|--------|--------|--------|--------|------|-------|--|
| 2847 | 016230 | 041103 | 052111 | 047040 | EM1 | ASCIZ | /CBIT NOT CLEARING IOCM/ |
| 2848 | 016236 | 052117 | 041440 | 042514 | | | |
| 2849 | 016244 | 051101 | 047111 | 020107 | | | |
| 2850 | 016252 | 047511 | 046503 | 000 | | | |
| 2851 | 016257 | 111 | 041517 | 020115 | EM2 | ASCIZ | /IOCM TEST/<15><12> |
| 2852 | 016264 | 042524 | 052123 | 005015 | | | |
| 2853 | 016272 | 051503 | 020122 | 040504 | EM2Y | ASCIZ | /CSR DATA ERROR WHEN TESTING / |
| 2854 | 016300 | 040524 | 042440 | 051122 | | | |
| 2855 | 016306 | 051117 | 053440 | 042510 | | | |
| 2856 | 016314 | 020116 | 042524 | 052123 | | | |
| 2857 | 016322 | 047111 | 020107 | | | | |
| 2858 | 016326 | 020040 | 044502 | 020124 | EM2X | ASCIZ | / BIT /<15><12> |
| 2859 | 016334 | 006440 | 000012 | | | | |
| 2860 | 016340 | 041520 | 052011 | 052123 | DH1 | ASCIZ | /PC TST# GODAT BDDAT PASS/ |
| 2861 | 016346 | 004443 | 042107 | 040504 | | | |
| 2862 | 016354 | 004524 | 042102 | 040504 | | | |
| 2863 | 016362 | 004524 | 040520 | 051523 | | | |
| 2864 | 016370 | 000 | | | | | |
| 2865 | 016371 | 111 | 041517 | 020115 | EM3 | ASCIZ | /IOCM IS NOT RESPONDING/<15><12> |
| 2866 | 016376 | 051511 | 047040 | 052117 | | | |
| 2867 | 016404 | 051040 | 051505 | 047520 | | | |
| 2868 | 016412 | 042116 | 047111 | 006507 | | | |
| 2869 | 016420 | 000012 | | | | | |
| 2870 | 016422 | 041104 | 051525 | 041040 | EM4 | ASCIZ | /DBUS BIT IS FAILING/<15><12> |
| 2871 | 016430 | 052111 | 044440 | 020123 | | | |
| 2872 | 016436 | 040506 | 046111 | 047111 | | | |
| 2873 | 016444 | 006507 | 000012 | | | | |
| 2874 | 016450 | 047516 | 044440 | 052116 | EM5 | ASCIZ | /NO INTERRUPT /<15><12> |
| 2875 | 016456 | 051105 | 052522 | 052120 | | | |
| 2876 | 016464 | 006440 | 000012 | | | | |
| 2877 | 016470 | 041520 | 052011 | 052123 | DH5 | ASCIZ | /PC TST# MODULE ADDR PASS/ |
| 2878 | 016476 | 004443 | 047515 | 052504 | | | |
| 2879 | 016504 | 042514 | 020040 | 040440 | | | |
| 2880 | 016512 | 042104 | 004522 | 040520 | | | |
| 2881 | 016520 | 051523 | 000 | | | | |
| 2882 | 016523 | 127 | 047522 | 043516 | EM6 | ASCIZ | /WRONG PATTERN IN IAR AFTER INTEPRUPT/<15><12> |
| 2883 | 016530 | 050040 | 052101 | 042524 | | | |
| 2884 | 016536 | 047122 | 044440 | 020116 | | | |
| 2885 | 016544 | 040511 | 020122 | 043101 | | | |
| 2886 | 016552 | 042524 | 020122 | 047111 | | | |
| 2887 | 016560 | 042524 | 051122 | 050125 | | | |
| 2888 | 016566 | 006524 | 000012 | | | | |
| 2889 | 016572 | | | | EM17 | | |
| 2890 | 016572 | 040504 | 040524 | 042440 | EM7 | ASCIZ | /DATA ERROR/<15><12> |
| 2891 | 016600 | 051122 | 051117 | 005015 | | | |
| 2892 | 016606 | 000 | | | | | |
| 2893 | 016607 | 120 | 004503 | 047515 | DH7 | ASCIZ | /PC MODULE ADDRESS TST# GODAT BDDAT PASS/ |
| 2894 | 016614 | 052504 | 042514 | 040411 | | | |
| 2895 | 016622 | 042104 | 042522 | 051523 | | | |
| 2896 | 016630 | 052011 | 052123 | 004443 | | | |
| 2897 | 016636 | 042107 | 040504 | 004524 | | | |
| 2898 | 016644 | 042102 | 040504 | 004524 | | | |
| 2899 | 016652 | 040520 | 051523 | 000 | | | |
| 2900 | 016657 | 124 | 051505 | 020124 | EM10 | ASCIZ | /TEST ABORTED, IOCM ERROR/<15><12> |
| 2901 | 016664 | 041101 | 051117 | 042524 | | | |
| 2902 | 016672 | 026104 | 044440 | 041517 | | | |

| | | | | | | |
|------|--------|--------|--------|--------|------|---|
| 2903 | 016700 | 020115 | 051105 | 047522 | | |
| 2904 | 016706 | 006522 | 000012 | | | |
| 2905 | 016712 | 040504 | 040524 | 047040 | EM11 | ASCIZ /DATA NOT CLEAR AFTER 10 SEC/<15><12> |
| 2906 | 016720 | 052117 | 041440 | 042514 | | |
| 2907 | 016726 | 051101 | 040440 | 052106 | | |
| 2908 | 016734 | 051105 | 030440 | 020060 | | |
| 2909 | 016742 | 042523 | 006503 | 000012 | | |
| 2910 | 016750 | 047111 | 042524 | 051122 | EM12 | ASCIZ /INTERRUPT TOO LATE /<15><12> |
| 2911 | 016756 | 050125 | 020124 | 047524 | | |
| 2912 | 016764 | 020117 | 040514 | 042524 | | |
| 2913 | 016772 | 020040 | 006440 | 000012 | | |
| 2914 | 017000 | 044522 | 020106 | 044502 | EM13 | ASCIZ /RIF BIT NOT CLEARING COS REGISTERS /<15><12> |
| 2915 | 017006 | 020124 | 047516 | 020124 | | |
| 2916 | 017014 | 046103 | 040505 | 044522 | | |
| 2917 | 017022 | 043516 | 041440 | 051517 | | |
| 2918 | 017030 | 051040 | 043505 | 051511 | | |
| 2919 | 017036 | 042524 | 051522 | 006440 | | |
| 2920 | 017044 | 000012 | | | | |
| 2921 | 017046 | 047516 | 044440 | 052116 | EM14 | ASCIZ /NO INTERRUPT IN MAINTENANCE MODE/<15><12> |
| 2922 | 017054 | 051105 | 052522 | 052120 | | |
| 2923 | 017062 | 044440 | 020116 | 040515 | | |
| 2924 | 017070 | 047111 | 042524 | 040516 | | |
| 2925 | 017076 | 041516 | 020105 | 047515 | | |
| 2926 | 017104 | 042504 | 005015 | 000 | | |
| 2927 | 017111 | 122 | 043111 | 041040 | EM15 | ASCIZ /RIF BIT ISN'T CLEARING INTERRUPT/<15><12> |
| 2928 | 017116 | 052111 | 044440 | 047123 | | |
| 2929 | 017124 | 052042 | 041440 | 042514 | | |
| 2930 | 017132 | 051101 | 047111 | 020107 | | |
| 2931 | 017140 | 047111 | 042524 | 051122 | | |
| 2932 | 017146 | 050125 | 006524 | 000012 | | |
| 2933 | 017154 | 046103 | 040505 | 044522 | EM16 | ASCIZ /CLEARING DBIT ISN'T CLEARING CSR/<15><12> |
| 2934 | 017162 | 043516 | 042040 | 044502 | | |
| 2935 | 017170 | 020124 | 051511 | 021116 | | |
| 2936 | 017176 | 020124 | 046103 | 040505 | | |
| 2937 | 017204 | 044522 | 043516 | 041440 | | |
| 2938 | 017212 | 051123 | 005015 | 000 | | |
| 2939 | 017217 | 103 | 051517 | 051040 | EM20 | ASCIZ /COS REGISTER DATA ERROR/<15><12> |
| 2940 | 017224 | 043505 | 051511 | 042524 | | |
| 2941 | 017232 | 020122 | 040504 | 040524 | | |
| 2942 | 017240 | 042440 | 051122 | 051117 | | |
| 2943 | 017246 | 005015 | 000 | | | |
| 2944 | 017251 | 114 | 047517 | 020120 | EM21 | ASCIZ /LOOP ERROR ON 6010-5010 OR 5011/<15><12> |
| 2945 | 017256 | 051105 | 047522 | 020122 | | |
| 2946 | 017264 | 047117 | 033040 | 030460 | | |
| 2947 | 017272 | 026460 | 030065 | 030061 | | |
| 2948 | 017300 | 047440 | 020122 | 030065 | | |
| 2949 | 017306 | 030461 | 005015 | 000 | | |
| 2950 | 017313 | 120 | 004503 | 042107 | DM21 | ASCIZ /PC GDDAT BDDAT PASS/ |
| 2951 | 017320 | 040504 | 004524 | 042102 | | |
| 2952 | 017326 | 040504 | 004524 | 040520 | | |
| 2953 | 017334 | 051523 | 000 | | | |
| 2954 | 017337 | 101 | 042104 | 042522 | EM22 | ASCIZ /ADDRESS TEST WITH MBIT SET ISN'T WORKING /<15><12> |
| 2955 | 017344 | 051523 | 052040 | 051505 | | |
| 2956 | 017352 | 020124 | 044527 | 044124 | | |
| 2957 | 017360 | 046440 | 044502 | 020124 | | |
| 2958 | 017366 | 042523 | 020124 | 051511 | | |

MAIN MACY11 30(1046) 25-OCT-77 10 10 PAGE 72
 DVPLPA P11 25-OCT-77 10 02 POWER DOWN AND UP ROUTINES

SEQ 0074

| | | | | | | | | | |
|------|--------|--------|--------|--------|------|-------|---|--|--|
| 2959 | 017374 | 023516 | 020124 | 047527 | | | | | |
| 2960 | 017402 | 045522 | 047111 | 020107 | | | | | |
| 2961 | 017410 | 005015 | 000 | | | | | | |
| 2962 | 017413 | 125 | 040516 | 046102 | EM23 | ASCIZ | /UNABLE TO CLEAR INTERRUPT /<15><12> | | |
| 2963 | 017420 | 020105 | 047524 | 041440 | | | | | |
| 2964 | 017426 | 042514 | 051101 | 044440 | | | | | |
| 2965 | 017434 | 052116 | 051105 | 052522 | | | | | |
| 2966 | 017442 | 052120 | 006440 | 000012 | | | | | |
| 2967 | 017450 | 041520 | 052011 | 052123 | DH23 | ASCIZ | /PC TST# IAR/ | | |
| 2968 | 017456 | 004443 | 040511 | 000122 | | | | | |
| 2969 | 017464 | 006516 | 000012 | | M1 | ASCIZ | /N/<15><12> | | |
| 2970 | 017470 | 006531 | 000012 | | M0 | ASCIZ | /Y/<15><12> | | |
| 2971 | 017474 | 000122 | | | M2 | ASCIZ | /R/ | | |
| 2972 | 017476 | 020107 | 000 | | M3 | ASCIZ | /G / | | |
| 2973 | 017501 | 124 | 000040 | | M4 | ASCIZ | /T / | | |
| 2974 | 017504 | 020104 | 000 | | M5 | ASCIZ | /D / | | |
| 2975 | 017507 | 115 | 000040 | | M6 | ASCIZ | /M / | | |
| 2976 | 017512 | 020105 | 000 | | M7 | ASCIZ | /E / | | |
| 2977 | 017515 | 040 | 040520 | 051523 | M11 | ASCIZ | / PASSES COMPLETED/<15><12> | | |
| 2978 | 017522 | 051505 | 041440 | 046517 | | | | | |
| 2979 | 017530 | 046120 | 052105 | 042105 | | | | | |
| 2980 | 017536 | 005015 | 000 | | | | | | |
| 2981 | 017541 | 115 | 042117 | 046125 | EM24 | ASCIZ | /MODULE UNDER TEST IS NOT RESPONDING/<15><12> | | |
| 2982 | 017546 | 020105 | 047125 | 042504 | | | | | |
| 2983 | 017554 | 020122 | 042524 | 052123 | | | | | |
| 2984 | 017562 | 044440 | 020123 | 047516 | | | | | |
| 2985 | 017570 | 020124 | 042522 | 050123 | | | | | |
| 2986 | 017576 | 047117 | 044504 | 043516 | | | | | |
| 2987 | 017604 | 005015 | 000 | | | | | | |
| 2988 | 017607 | 015 | 052012 | 051505 | M13 | ASCIZ | <15><12>/TEST OPTIONS /<15><12> | | |
| 2989 | 017614 | 020124 | 050117 | 044524 | | | | | |
| 2990 | 017622 | 047117 | 035123 | 005015 | | | | | |
| 2991 | 017630 | 000 | | | | | | | |
| 2992 | 017631 | 123 | 020040 | 054523 | M14 | ASCIZ | /S SYSTEM TEST/<15><12> | | |
| 2993 | 017636 | 052123 | 046505 | 052040 | | | | | |
| 2994 | 017644 | 051505 | 006524 | 000012 | | | | | |
| 2995 | 017652 | 020104 | 042040 | 043511 | M15 | ASCIZ | /D DIGITAL MODULE/<15><12> | | |
| 2996 | 017660 | 052111 | 046101 | 046440 | | | | | |
| 2997 | 017666 | 042117 | 046125 | 006505 | | | | | |
| 2998 | 017674 | 000012 | | | | | | | |
| 2999 | 017676 | 020101 | 040440 | 040516 | M16 | ASCIZ | /A ANALOG MODULE/<15><12> | | |
| 3000 | 017704 | 047514 | 020107 | 047515 | | | | | |
| 3001 | 017712 | 052504 | 042514 | 005015 | | | | | |
| 3002 | 017720 | 000 | | | | | | | |
| 3003 | 017721 | 115 | 020040 | 040515 | M17 | ASCIZ | /M MAP OF DBLS INTERFACES/<15><12> | | |
| 3004 | 017726 | 020120 | 043117 | 042040 | | | | | |
| 3005 | 017734 | 052502 | 020123 | 047111 | | | | | |
| 3006 | 017742 | 042524 | 043122 | 041501 | | | | | |
| 3007 | 017750 | 051505 | 005015 | 000 | | | | | |
| 3008 | 017755 | 130 | 020040 | 054105 | M18 | ASCIZ | /X EXERCIZER/<15><12> | | |
| 3009 | 017762 | 051105 | 044503 | 042532 | | | | | |
| 3010 | 017770 | 006522 | 000012 | | | | | | |
| 3011 | 017774 | 020111 | 044440 | 041517 | M19 | ASCIZ | /I IOCM TEST/<15><12> | | |
| 3012 | 020002 | 020115 | 042524 | 052123 | | | | | |
| 3013 | 020010 | 005015 | 000 | | | | | | |
| 3014 | 020013 | 040 | 020040 | 006477 | M20 | ASCIZ | / ?/<15><12> | | |

MAIN MACY11 30(1046) 25-OCT-77 10 10 PAGE 73
 DVPLAR P11 25-OCT-77 10 02 POWER DOWN AND UP ROUTINES

SEQ 0075

| | | | | | | | |
|------|--------|--------|--------|--------|-------|--|--|
| 3015 | 020020 | 000012 | | | | | |
| 3016 | 020022 | 020124 | 051440 | 052105 | M21 | ASCIZ /T SET SWREG/<15><12> | |
| 3017 | 020030 | 051440 | 051127 | 043505 | | | |
| 3018 | 020036 | 005015 | 000 | | | | |
| 3019 | 020041 | 061 | 030060 | 030060 | M22 | ASCII /10000 HALT ON ERROR/<15><12> | |
| 3020 | 020046 | 020060 | 020040 | 040510 | | | |
| 3021 | 020054 | 052114 | 047440 | 020116 | | | |
| 3022 | 020062 | 051105 | 047522 | 006522 | | | |
| 3023 | 020070 | 012 | | | | | |
| 3024 | 020071 | 064 | 030060 | 030060 | | ASCII /40000 LOOP ON TEST/<15><12> | |
| 3025 | 020076 | 020040 | 020040 | 047514 | | | |
| 3026 | 020104 | 050117 | 047440 | 020116 | | | |
| 3027 | 020112 | 042524 | 052123 | 005015 | | | |
| 3028 | 020120 | 030062 | 030060 | 020060 | | ASCIZ /20000 INHIBIT ERROR & EOP PRINT/<15><12> | |
| 3029 | 020126 | 020040 | 044440 | 044116 | | | |
| 3030 | 020134 | 041111 | 052111 | 042440 | | | |
| 3031 | 020142 | 051122 | 051117 | 023040 | | | |
| 3032 | 020150 | 042440 | 050117 | 050040 | | | |
| 3033 | 020156 | 044522 | 052116 | 005015 | | | |
| 3034 | 020164 | 000 | | | | | |
| 3035 | 020165 | 061 | 030060 | 030060 | | ASCIZ /10000 LOOP ON ERROR/<15><12> | |
| 3036 | 020172 | 020040 | 020040 | 047514 | | | |
| 3037 | 020200 | 050117 | 047440 | 020116 | | | |
| 3038 | 020206 | 051105 | 047522 | 006522 | | | |
| 3039 | 020214 | 000012 | | | | | |
| 3040 | 020216 | 005015 | 053523 | 042522 | M23 | ASCIZ <15><12>/SWREG =/ | |
| 3041 | 020224 | 020107 | 000075 | | | | |
| 3042 | 020230 | 020106 | 043040 | 042511 | M24 | ASCIZ /F FIELD TEST/<15><12> | |
| 3043 | 020236 | 042114 | 052040 | 051505 | | | |
| 3044 | 020244 | 006524 | 000012 | | | | |
| 3045 | 020250 | 020127 | 046040 | 047517 | M25 | ASCIZ /W LOOP TEST/<15><12> | |
| 3046 | 020256 | 020120 | 042524 | 052123 | | | |
| 3047 | 020264 | 005015 | 000 | | | | |
| 3048 | 020267 | 114 | 020040 | 047514 | M26 | ASCIZ /L LOAD ITERATION COUNT/<15><12> | |
| 3049 | 020274 | 042101 | 044440 | 042524 | | | |
| 3050 | 020302 | 040522 | 044524 | 047117 | | | |
| 3051 | 020310 | 041440 | 052517 | 052116 | | | |
| 3052 | 020316 | 005015 | 000 | | | | |
| 3053 | 020321 | 116 | 046525 | 042502 | M27 | ASCIZ /NUMBER OF ITERATIONS(OCT) = / | |
| 3054 | 020326 | 020122 | 043117 | 044440 | | | |
| 3055 | 020334 | 042524 | 040522 | 044524 | | | |
| 3056 | 020342 | 047117 | 024123 | 041517 | | | |
| 3057 | 020350 | 024524 | 036440 | 000040 | | | |
| 3058 | 020356 | 044514 | 042516 | 041440 | EM25 | ASCIZ /LINE CLOCK IS NOT INTERRUPTING/<15><12> | |
| 3059 | 020364 | 047514 | 045503 | 044440 | | | |
| 3060 | 020372 | 020123 | 047516 | 020124 | | | |
| 3061 | 020400 | 047111 | 042524 | 051122 | | | |
| 3062 | 020406 | 050125 | 044524 | 043516 | | | |
| 3063 | 020414 | 005015 | 000 | | | | |
| 3064 | 020417 | 015 | 000012 | | MASS0 | ASCIZ <15><12> | |
| 3065 | 020422 | 042101 | 051104 | 051505 | MASS2 | ASCIZ /ADDRESS / | |
| 3066 | 020430 | 020123 | 000040 | | | | |
| 3067 | 020434 | 020040 | 046440 | 030065 | MASS3 | ASCIZ / M5010 NONISOLATED 32 BIT DC INPUT/<15><12> | |
| 3068 | 020442 | 030061 | 047040 | 047117 | | | |
| 3069 | 020450 | 051511 | 046117 | 052101 | | | |
| 3070 | 020456 | 042105 | 031440 | 020062 | | | |

MAIN MACY11 30(1046) 25-OCT-77 10 10 PAGE 74
 DVPLAR P11 25-OCT-77 10 02 POWER DOWN AND UP ROUTINES

SEQ 0076

| | | | | | | | | |
|------|--------|--------|--------|--------|--------|--------------------------------------|---|--|
| 3071 | 020464 | 044502 | 020124 | 041504 | | | | |
| 3072 | 020472 | 044440 | 050116 | 052125 | | | | |
| 3073 | 020500 | 005015 | 000 | | | | | |
| 3074 | 020503 | 040 | 020040 | 032515 | MASS4 | ASCIZ / | M5011 NONISOLATED 16 BIT DC INPUT/<15><12> | |
| 3075 | 020510 | 030460 | 020061 | 047516 | | | | |
| 3076 | 020516 | 044516 | 047523 | 040514 | | | | |
| 3077 | 020524 | 042524 | 020104 | 033061 | | | | |
| 3078 | 020532 | 041040 | 052111 | 042040 | | | | |
| 3079 | 020540 | 020103 | 047111 | 052520 | | | | |
| 3080 | 020546 | 006524 | 000012 | | | | | |
| 3081 | 020552 | 020040 | 046440 | 030065 | MASS5 | ASCIZ / | M5012 ISOLATED 16 BIT DC INPUT/<15><12> | |
| 3082 | 020560 | 031061 | 044440 | 047523 | | | | |
| 3083 | 020566 | 040514 | 042524 | 020104 | | | | |
| 3084 | 020574 | 033061 | 041040 | 052111 | | | | |
| 3085 | 020602 | 042040 | 020103 | 047111 | | | | |
| 3086 | 020610 | 052520 | 006524 | 000012 | | | | |
| 3087 | 020616 | 020040 | 046440 | 030065 | MASS6 | ASCIZ / | M5013 8 BIT AC INPUT/<15><12> | |
| 3088 | 020624 | 031461 | 034040 | 041040 | | | | |
| 3089 | 020632 | 052111 | 040440 | 020103 | | | | |
| 3090 | 020640 | 047111 | 052520 | 006524 | | | | |
| 3091 | 020646 | 000012 | | | | | | |
| 3092 | 020650 | 020040 | 046440 | 030066 | MASS7 | ASCIZ / | M6010 NONISOLATED 32 BIT DC OUTPUT/<15><12> | |
| 3093 | 020656 | 030061 | 047040 | 047117 | | | | |
| 3094 | 020664 | 051511 | 046117 | 052101 | | | | |
| 3095 | 020672 | 042105 | 031440 | 020062 | | | | |
| 3096 | 020700 | 044502 | 020124 | 041504 | | | | |
| 3097 | 020706 | 047440 | 052125 | 052520 | | | | |
| 3098 | 020714 | 006524 | 000012 | | | | | |
| 3099 | 020720 | 020040 | 046440 | 030066 | MASS8 | ASCIZ / | M6011 16 BIT ONESHOT OUTPUT/<15><12> | |
| 3100 | 020726 | 030461 | 030440 | 020066 | | | | |
| 3101 | 020734 | 044502 | 020124 | 047117 | | | | |
| 3102 | 020742 | 051505 | 047510 | 020124 | | | | |
| 3103 | 020750 | 052517 | 050124 | 052125 | | | | |
| 3104 | 020756 | 005015 | 000 | | | | | |
| 3105 | 020761 | 040 | 020040 | 033115 | MASS9 | ASCIZ / | M6012 ISOLATED 8 BIT DC OUTPUT/<15><12> | |
| 3106 | 020766 | 030460 | 020062 | 051511 | | | | |
| 3107 | 020774 | 046117 | 052101 | 042105 | | | | |
| 3108 | 021002 | 034040 | 041040 | 052111 | | | | |
| 3109 | 021010 | 042040 | 020103 | 052517 | | | | |
| 3110 | 021016 | 050124 | 052125 | 005015 | | | | |
| 3111 | 021024 | 000 | | | | | | |
| 3112 | 021025 | 040 | 020040 | 033115 | MASS10 | ASCIZ / | M6013 8 BIT AC OUTPUT/<15><12> | |
| 3113 | 021032 | 030460 | 020063 | 020070 | | | | |
| 3114 | 021040 | 044502 | 020124 | 041501 | | | | |
| 3115 | 021046 | 047440 | 052125 | 052520 | | | | |
| 3116 | 021054 | 006524 | 000012 | | | | | |
| 3117 | 021060 | 005015 | 020040 | 052440 | MASS11 | ASCIZ <15><12>/ | UNKNOWN GENERIC CODE / | |
| 3118 | 021066 | 045516 | 047516 | 047127 | | | | |
| 3119 | 021074 | 043440 | 047105 | 051105 | | | | |
| 3120 | 021102 | 041511 | 041440 | 042117 | | | | |
| 3121 | 021110 | 020105 | 000040 | | | | | |
| 3122 | 021114 | 005015 | 044127 | 041511 | MASS12 | ASCIZ <15><12>/WHICH TEST OPTION ? / | | |
| 3123 | 021122 | 020110 | 042524 | 052123 | | | | |
| 3124 | 021130 | 047440 | 052120 | 047511 | | | | |
| 3125 | 021136 | 020116 | 020077 | 000040 | | | | |
| 3126 | 021144 | 054524 | 042520 | 040440 | MASS13 | ASCIZ /TYPE ADDRESS OF MUT / | | |

MAIN MACY11 30(1046) 25-OCT-77 10.10 PAGE 75
 DVP(LAR P11 25-OCT-77 10 02 POWER DOWN AND UP ROUTINES

SEQ 0077

| | | | | | | | |
|------|--------|--------|--------|--------|--------|-------|---|
| 3127 | 021152 | 042104 | 042522 | 051523 | | | |
| 3128 | 021160 | 047440 | 020106 | 052515 | | | |
| 3129 | 021166 | 020124 | 020040 | 000040 | | | |
| 3130 | 021174 | 047111 | 042524 | 051122 | MASS14 | ASCIZ | /INTERRUPT TOO LATE/<15><12> |
| 3131 | 021202 | 050125 | 020124 | 047524 | | | |
| 3132 | 021210 | 020117 | 040514 | 042524 | | | |
| 3133 | 021216 | 005015 | 000 | | | | |
| 3134 | 021221 | 105 | 042116 | 047440 | MASS15 | ASCIZ | /END OF MODULE TEST/<15><12> |
| 3135 | 021226 | 020106 | 047515 | 052504 | | | |
| 3136 | 021234 | 042514 | 052040 | 051505 | | | |
| 3137 | 021242 | 006524 | 000012 | | | | |
| 3138 | 021246 | 047105 | 020104 | 043117 | MASS16 | ASCIZ | /END OF IOCM TEST/<15><12> |
| 3139 | 021254 | 044440 | 041517 | 020115 | | | |
| 3140 | 021262 | 042524 | 052123 | 005015 | | | |
| 3141 | 021270 | 000 | | | | | |
| 3142 | 021271 | 105 | 042116 | 047440 | MASS17 | ASCIZ | /END OF AUTO TEST/<15><12> |
| 3143 | 021276 | 020106 | 040440 | 052125 | | | |
| 3144 | 021304 | 020117 | 042524 | 052123 | | | |
| 3145 | 021312 | 005015 | 000 | | | | |
| 3146 | 021315 | 040 | 020040 | 005015 | MASS18 | ASCIZ | / /<15><12> |
| 3147 | 021322 | 000 | | | | | |
| 3148 | 021323 | 124 | 050131 | 020105 | MASS19 | ASCIZ | /TYPE ONE BYTE DATA PATTERN / |
| 3149 | 021330 | 047117 | 020105 | 054502 | | | |
| 3150 | 021336 | 042524 | 042040 | 052101 | | | |
| 3151 | 021344 | 020101 | 040520 | 052124 | | | |
| 3152 | 021352 | 051105 | 020116 | 020040 | | | |
| 3153 | 021360 | 000040 | | | | | |
| 3154 | 021362 | 054524 | 042520 | 041440 | MASS20 | ASCIZ | /TYPE CONTROL-C TO RETURN TO MONITOR/<15><12> |
| 3155 | 021370 | 047117 | 051124 | 046117 | | | |
| 3156 | 021376 | 041455 | 052040 | 020117 | | | |
| 3157 | 021404 | 042522 | 052524 | 047122 | | | |
| 3158 | 021412 | 052040 | 020117 | 047515 | | | |
| 3159 | 021420 | 044516 | 047524 | 006522 | | | |
| 3160 | 021426 | 000012 | | | | | |
| 3161 | 021430 | 054524 | 042520 | 040440 | MASS21 | ASCIZ | /TYPE ADDRESS OF OUTPUT MUT/ |
| 3162 | 021436 | 042104 | 042522 | 051523 | | | |
| 3163 | 021444 | 047440 | 020106 | 052517 | | | |
| 3164 | 021452 | 050124 | 052125 | 046440 | | | |
| 3165 | 021460 | 052125 | 000 | | | | |
| 3166 | 021463 | 124 | 050131 | 020105 | MASS22 | ASCIZ | /TYPE ADDRESS OF INPUT MUT/ |
| 3167 | 021470 | 042101 | 051104 | 051505 | | | |
| 3168 | 021476 | 020123 | 043117 | 044440 | | | |
| 3169 | 021504 | 050116 | 052125 | 046440 | | | |
| 3170 | 021512 | 052125 | 000 | | | | |
| 3171 | 021515 | 103 | 047117 | 042516 | MASS23 | ASCIZ | /CONNECT OUTPUT TO INPUT MODULE/<15><12> |
| 3172 | 021522 | 052103 | 047440 | 052125 | | | |
| 3173 | 021530 | 052520 | 020124 | 047524 | | | |
| 3174 | 021536 | 044440 | 050116 | 052125 | | | |
| 3175 | 021544 | 046440 | 042117 | 046125 | | | |
| 3176 | 021552 | 006505 | 000012 | | | | |
| 3177 | 021556 | 047105 | 020104 | 043117 | MASS25 | ASCIZ | /END OF LOOP TEST/<15><12> |
| 3178 | 021564 | 046040 | 047517 | 020120 | | | |
| 3179 | 021572 | 042524 | 052123 | 005015 | | | |
| 3180 | 021600 | 000 | | | | | |
| 3181 | 021601 | 127 | 047522 | 043516 | MASS26 | ASCIZ | /WRONG MODULE-OUTPUT MUST BE M6010/<15><12> |
| 3182 | 021606 | 046440 | 042117 | 046125 | | | |

```

3183 021614 026505 052517 050124
3184 021622 052125 046440 051525
3185 021630 020124 042502 046440
3186 021636 030066 030061 005015
3187 021644      000
3188 021645      072 000011      MASS27  ASCIZ / /
3189 021650 051127 047117 020107  MASS28  ASCIZ /WRONG MODULE-INPUT MUST BE EITHER M5010 OR M5011/<15><12>
3190 021656 047515 052504 042514
3191 021664 044455 050116 052125
3192 021672 046440 051525 020.24
3193 021700 042502 042440 052111
3194 021706 042510 020122 032515
3195 021714 030460 020060 051117
3196 021722 046440 030065 030461
3197 021730 005015      000
3198      021734
3199 021734 001116 002002 002050  EVEN
3200 021742 002022 001124 001126  DT17  WORD  SERRPC, $MUT,  TEMP1,  TNUMB,  $GDDAT,  $BDDAT,  PASS,
3201 021750 002056 000000
3202 021754 001116 002002 002014  DT20  WORD  SERRPC, $MUT,  COSADR,  TNUMB,  $GDDAT,  $BDDAT,  PASS,
3203 021762 002022 001124 001126
3204 021770 002056 000000
3205 021774 001116 002022 002052  DT23  WORD  SERRPC,  TNUMB,  TEMP3,
3206 022002 000000
3207 022004 001116 002002 002012  DT7   WORD  SERRPC, $MUT,  TBADDR,  TNUMB,  $GDDAT,  $BDDAT,  PASS,
3208 022012 002022 001124 001126
3209 022020 002056 000000
3210 022024 001116 002022 001124  DT1   WORD  SERRPC,  TNUMB,  $GDDAT,  $BDDAT,  PASS,
3211 022032 001126 002056 000000
3212 022040 001116 002022 002002  DT5   WORD  SERRPC,  TNUMB,  $MUT,  TADDR,  PASS,
3213 022046 002006 002056 000000
3214 022054 001116 001124 001126  DT21  WORD  SERRPC,  $GDDAT,  $BDDAT,  PASS,
3215 022062 002056 000000
3216      000001      END

```

| | | | | | | | | | | | | | | |
|----------|--------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|-------|
| ANALOG | 011150 | 791 | 1803# | | | | | | | | | | | |
| ANSW | 001764 | 557# | | | | | | | | | | | | |
| ATABL | 001200 | 529# | 910 | 954 | | | | | | | | | | |
| AUTO | 003612 | 758 | 785 | 898# | | | | | | | | | | |
| BASE | 002042 | 582# | 908 | 1129 | 1338 | 1508 | 1744 | 1856 | 2035 | | | | | |
| BITSET | 012230 | 1234 | 1237 | 1246 | 1249 | 1259 | 1262 | 1293 | 1296 | 1306 | 1309 | 1319 | 1322 | 1999# |
| BIT0 = | 000001 | 455# | 478 | | | | | | | | | | | |
| BIT00 = | 000001 | 445# | 455 | | | | | | | | | | | |
| BIT01 = | 000002 | 444# | 454 | | | | | | | | | | | |
| BIT02 = | 000004 | 443# | 453 | | | | | | | | | | | |
| BIT03 = | 000010 | 442# | 452 | | | | | | | | | | | |
| BIT04 = | 000020 | 441# | 451 | | | | | | | | | | | |
| BIT05 = | 000040 | 440# | 450 | | | | | | | | | | | |
| BIT06 = | 000100 | 439# | 449 | | | | | | | | | | | |
| BIT07 = | 000200 | 438# | 448 | | | | | | | | | | | |
| BIT08 = | 000400 | 437# | 447 | | | | | | | | | | | |
| BIT09 = | 001000 | 436# | 446 | 2257 | 2443 | | | | | | | | | |
| BIT1 = | 000002 | 454# | 477 | | | | | | | | | | | |
| BIT10 = | 002000 | 435# | | | | | | | | | | | | |
| BIT11 = | 004000 | 434# | | | | | | | | | | | | |
| BIT12 = | 010000 | 433# | | | | | | | | | | | | |
| BIT13 = | 020000 | 432# | 857 | 969 | 1039 | 1190 | 2435 | | | | | | | |
| BIT14 = | 040000 | 431# | 862 | 974 | 1044 | 1195 | | | | | | | | |
| BIT15 = | 100000 | 430# | | | | | | | | | | | | |
| BIT2 = | 000004 | 453# | 476 | | | | | | | | | | | |
| BIT3 = | 000010 | 452# | 475 | | | | | | | | | | | |
| BIT4 = | 000020 | 451# | 474 | | | | | | | | | | | |
| BIT5 = | 000040 | 450# | 473 | | | | | | | | | | | |
| BIT6 = | 000100 | 449# | 472 | 750 | 761 | 777 | 821 | 900 | 1016 | 1071 | 1128 | 1185 | | |
| BIT7 = | 000200 | 448# | 471 | 2049 | | | | | | | | | | |
| BIT8 = | 000400 | 447# | 2039 | | | | | | | | | | | |
| BIT9 = | 001000 | 446# | | | | | | | | | | | | |
| BPTVEC = | 000014 | 462# | | | | | | | | | | | | |
| BTABL | 001440 | 530# | 911 | 943 | 949 | 962 | | | | | | | | |
| BYTNUM | 002004 | 567# | 1441* | 1446* | 1469* | 1475* | 1606* | 1608* | 1613* | 1615* | 1619* | 1621* | 1626* | 1628* |
| | | 1655* | 1657* | 1677* | 1679* | 1710* | 1715* | 1819* | 1824* | 1897* | 1901* | 1905* | 1909* | 1916* |
| | | 1921* | 1931 | 1939* | 1942* | 1946 | 1954* | 1957* | 1963 | 1967* | 1970* | 2074 | 2079* | 2083* |
| | | 2085* | 2101* | 2109 | 2127 | 2149* | 2155* | 2159* | 2165* | 2169* | 2175* | 2179* | 2185* | 2191* |
| | | 2197* | 2201* | 2207* | 2211* | 2217* | 2221* | 2227* | | | | | | |
| CBIT = | 000002 | 477# | 840 | 845 | 1052 | 1057 | 1083 | 1088 | 1208 | 1213 | 1275 | 1280 | 1352 | 1357 |
| | | 1403 | 1408 | 2052 | 2057 | 2136 | 2141 | | | | | | | |
| CHKBYT | 012556 | 2072# | 2156 | 2166 | 2176 | 2186 | 2198 | 2208 | 2218 | 2228 | | | | |
| CLK | 002044 | 583# | 1647* | 1662 | 1670 | 1675* | 1685 | 2026* | | | | | | |
| CLKADR | 002024 | 575# | 750* | | | | | | | | | | | |
| CLKVC | 002062 | 590# | 1648* | 1690* | | | | | | | | | | |
| CLKVCA | 002064 | 591# | 1649* | | | | | | | | | | | |
| CLRINT | 012360 | 1034 | 1298 | 1449 | 1473 | 1478 | 1496 | 1522 | 1717 | 1733 | 1784 | 1827 | 1844 | 1875 |
| | | 2030# | | | | | | | | | | | | |
| CNTRC | 012446 | 853 | 965 | 1035 | 1096 | 2048# | 2067 | 2128 | | | | | | |
| COSADR | 002014 | 571# | 1706* | 1707* | 1754 | 1759 | 1761 | 1764* | 1772 | 1777 | 1779 | 1782* | 3202 | |
| COUNT | 012352 | 1648 | 2026# | | | | | | | | | | | |
| CR = | 000015 | 370# | 2562 | 2572 | | | | | | | | | | |
| CRLF = | 000200 | 371# | 746 | 2533 | 2572 | | | | | | | | | |
| CSR | 002026 | 576# | 827 | 832* | 840* | 845* | 902 | 909* | 939* | 1023* | 1052* | 1057* | 1076* | 1083* |
| | | 1088* | 1130* | 1207* | 1208* | 1213* | 1218 | 1264* | 1265* | 1270 | 1275* | 1280* | 1337* | 1352* |
| | | 1357* | 1383* | 1397* | 1398 | 1399 | 1403* | 1408* | 1439* | 1444* | 1448* | 1467* | 1472* | 1477* |

MAIN MACY11 30(1046) 25-OCT-77 10 10 PAGE 86
DVPCAA P11 25-OCT-77 10 02

CROSS REFERENCE TABLE -- USER SYMBOLS

SEQ 0087

| | | | | | | | | | | | | | |
|----------|------|-------|------|------|-------|------|------|------|------|-------|------|-------|------|
| = 022066 | 342# | 346# | 351# | 353# | 355# | 357# | 359# | 487# | 528 | 529# | 530# | 707 | 719 |
| | 746# | 1008 | 2270 | 2290 | 2354# | 2355 | 2361 | 2414 | 2455 | 2501# | 2572 | 2638# | 2820 |
| | 2842 | 3198# | | | | | | | | | | | |

ABS 022066 000

ERRORS DETECTED 0

DVPCAA BIN. DVPCAA SEQ/CRF=DVPCAA P11
RUN-TIME 17 9 9 SECONDS
RUN-TIME RATIO 441/28=15 6
CORE USED 24K (47 PAGES)