

DRV11B

INTERPROCESSOR EXERCISER
MD-11-DVDRB-A

EP-DVDRB-A-DL-A

NOV 1976

COPYRIGHT © 1976

FICHE 1 OF 1

MADE IN U.S.A.

This microfiche card contains a grid of frames. The first column contains text-based data, likely instructions or test results. The second and third columns contain diagrams, possibly circuit schematics or data flow charts. The fourth column contains additional text or data. The frames are arranged in a regular grid pattern, typical of microfiche format.

.REM x

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DVDRB-A
PRODUCT NAME: DRV11B INTERPROCESSOR EXERCISER
DATE: OCTOBER 1976
MAINTAINER: DIAGNOSTIC ENGINEERING

COPYRIGHT (C) 1976
DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASS.

THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.

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

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

MAINDEC-11-DVDRB-A
DRV11B INTERPROCESSOR EXERCISER
MACY11 27(665)
12-OCT-76 13:23
PAGE 1

9887274-7292969599120-104720

TABLE OF CONTENTS

1.0	ABSTRACT
2.0	REQUIREMENTS
2.1	EQUIPMENT
2.2	STORAGE
3.0	LOADING PROCEDURE
4.0	STARTING PROCEDURE
5.0	SOFTWARE SWITCH REGISTER
5.1	OPTIONS
5.2	CONTROL
6.0	ERRORS
6.1	ERROR COMMENT
6.2	ERROR DATA
6.3	ERROR RECOVERY
7.0	MISCELLANEOUS
7.1	DRV11B BUS & VECTOR ADDRESS MODIFICATION
8.0	EXECUTION TIME
9.0	PROGRAM DESCRIPTION
9.1	GENERAL
9.2	PROGRAM SEGMENTS
10.0	LISTING

E01

MAINDEC-11-DVDRB-A DRV11B INTERPROCESSOR EXERCISER
DVDRBA.P11

MACY11 27(665) 12-OCT-76 13:23 PAGE 4

137
138
139
140
141

THE CHARACTER 'G' NEED BE TYPED (WITH THE DRV11B BINARY
TAPE IN THE APPROPRIATE READER).

6. REPEAT THE LOADING PROCEDURE IN THE OTHER COMPUTER SYSTEM.

153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195

4.0 STARTING PROCEDURE

1. MAKE SURE EACH DRV11B INTERFACE OF BOTH COMPUTER SYSTEMS ARE CABLED TOGETHER THRU THE I/O CONNECTIONS ON THE M7950 MODULE.
2. MAKE SURE THE DEVICE & VECTOR ADDRESSES AGREE WITH THE DEFAULT VALUES DEFINED IN SECTION 7.1. IF NOT, CHANGE LOCATION(S) AS DESIRED VIA THE 'ADDRESS/' ODT COMMAND.
3. INSURE THAT THE HALT SWITCH IS DISABLED (IF ANY).
4. DEFINE WHAT COMPUTER IS TO BE THE INITIAL SLAVE - THE OTHER WILL BE THE INITIAL MASTER (THE ONE STARTED AS MASTER WILL REPORT THE 'END OF PASS' MESSAGE).
5. START THE SLAVE **#FIRST#** AT ADDRESS 204 USING THE ODT 'ADDRESS G' SEQUENCE. AFTER THE SLAVE HAS BEEN STARTED THEN START THE OTHER COMPUTER (MASTER) AT ADDRESS 200.

5.0 SOFTWARE SWITCH REGISTER

5.1 OPTIONS

<u>SWITCH</u>	<u>OCTAL</u>	<u>FUNCTION</u>
BIT15=1	100000	HALT ON ERROR
BIT13=1	020000	INHIBIT ERROR TYPEOUTS
BIT10=1	002000	BELL ON ERROR

5.2 CONTROL

1. THE SOFTWARE SWITCH REGISTER 'SWREG' (LOC. 176) CAN BE CHANGED BY USING THE ODT FACILITIES.
2. THE SOFTWARE SWITCH REGISTER CAN BE CHANGED UNDER PROGRAM CONTROL BY TYPING THE 'CONTROL & G' KEYS. THIS KEYBOARD OPERATION WILL PRINT OUT THE CURRENT CONTENTS AND ACCEPT NEW OCTAL SWITCH REGISTER DATA TERMINATED WITH A CARRIAGE RETURN.

GO1

196
197
198
199
200

3. ONCE THE ODT MODE HAS BEEN ENTERED BECAUSE OF AN ERROR CONDITION WITH BIT15 SET (HALT ON ERROR), STEP #2 ABOVE IS OF NO VALUE, SO RESORT TO STEP #1 TO ALTER THE SOFT. RE SWITCH REGISTER IF DESIRED BEFORE TYPEING 'P' (CONTINUE).

101

MAINDEC-11-DVDRB-A DRV11B INTERPROCESSOR EXERCISER
DVDRBA.P11

MACY11 27(665) 12-OCT-76 13:23 PAGE 8

START
200

EXECUTION TIME IS ABOUT 1 MINUTE. NOTE THAT THE 'END OF
PASS' MESSAGE IS ONLY REPORTED AT THE COMPUTER WHICH WAS
STARTED AS THE INITIAL MASTER (START 200).

9.0 PROGRAM DESCRIPTION

9.1 GENERAL

THIS INTERPROCESSOR EXERCISER WAS DESIGNED TO TEST THE I/O ABILITY OF THE DRV11B GENERAL PURPOSE INTERFACE TO COMMUNICATE TO ANOTHER DRV11B LOCATED IN ANOTHER LSI-11 COMPUTER. THE TWO COMPUTERS ARE STARTED AT DIFFERENT ADDRESSES TO ESTABLISH INITIAL SYNCHRONIZATION. THE SLAVE COMPUTER IS STARTED 1ST (START 204) AND THE MASTER COMPUTER IS STARTED 2ND (START 200). THE TERMS 'MASTER' AND 'SLAVE' SHOULD BE USED LOOSELY AS THE MASTER WILL BECOME THE SLAVE AND THE SLAVE WILL BECOME THE MASTER AS THE PROGRAM ADVANCES. THE COMPUTER STARTED AT ADDRESS 200 WILL ALWAYS REPORT THE 'END OF PASS' MESSAGE.

9.2 PROGRAM SEGMENTS

1. NTST1 - MASTER SENDS OUT PROGRAM CONTROLLED SINGLE WORDS THRU THE DATA BUFFER REGISTER AND EXPECTS THE SLAVE TO ECHO EACH WORD BACK TO THE MASTER VIA THE DATA BUFFER REGISTER.
2. NTST2 - MASTER SENDS OUT A 'FNCT' BIT CODE IN THE COMMAND STATUS REGISTER AND EXPECTS THE SLAVE TO ECHO EACH CODE IN ITS 'FNCT' BITS. THE MASTER WILL READ THE 'STAT' BIT CODE FROM THE COMMAND/STATUS REGISTER AND COMPARE IT TO THE CODE WRITTEN.
3. NTST3 - MASTER SENDS OUT A 32 WORD DATA BLOCK TO THE SLAVE AND CHECKS FOR PROPER INTERRUPT STATUS, WORD COUNT, AND BUFFER ADDRESS AT THE COMPLETION OF THE TRANSFER.
4. STST1 - SLAVE ECHOS ALL CHANGES IN THE DATA BUFFER REGISTER
5. STST2 - SLAVE READS THE 'STAT' BIT CODE FROM IT'S COMMAND/STATUS REGISTER, CONVERTS THIS CODE AND WRITES IT INTO IT'S 'FNCT' BITS IN THE COMMAND/STATUS REGISTER.
6. STST3 - SLAVE RECEIVES A 32 WORD DATA BLOCK FROM THE MASTER AND CHECKS FOR PROPER INTERRUPT STATUS, WORD COUNT, BUFFER ADDRESS, AND DATA CONTENT.

10.0 LISTING

01-07-76 15:29
02-07-76 15:29
03-07-76 15:29
04-07-76 15:29
05-07-76 15:29
06-07-76 15:29
07-07-76 15:29
08-07-76 15:29
09-07-76 15:29
10-07-76 15:29
11-07-76 15:29
12-07-76 15:29
13-07-76 15:29
14-07-76 15:29
15-07-76 15:29
16-07-76 15:29
17-07-76 15:29
18-07-76 15:29
19-07-76 15:29
20-07-76 15:29
21-07-76 15:29
22-07-76 15:29
23-07-76 15:29
24-07-76 15:29
25-07-76 15:29
26-07-76 15:29
27-07-76 15:29
28-07-76 15:29
29-07-76 15:29
30-07-76 15:29
31-07-76 15:29
32-07-76 15:29
33-07-76 15:29
34-07-76 15:29
35-07-76 15:29
36-07-76 15:29
37-07-76 15:29
38-07-76 15:29
39-07-76 15:29
40-07-76 15:29
41-07-76 15:29
42-07-76 15:29
43-07-76 15:29
44-07-76 15:29
45-07-76 15:29
46-07-76 15:29
47-07-76 15:29
48-07-76 15:29
49-07-76 15:29
50-07-76 15:29
51-07-76 15:29
52-07-76 15:29
53-07-76 15:29
54-07-76 15:29
55-07-76 15:29
56-07-76 15:29
57-07-76 15:29
58-07-76 15:29
59-07-76 15:29
60-07-76 15:29
61-07-76 15:29
62-07-76 15:29
63-07-76 15:29
64-07-76 15:29
65-07-76 15:29
66-07-76 15:29
67-07-76 15:29
68-07-76 15:29
69-07-76 15:29
70-07-76 15:29
71-07-76 15:29
72-07-76 15:29
73-07-76 15:29
74-07-76 15:29
75-07-76 15:29
76-07-76 15:29
77-07-76 15:29
78-07-76 15:29
79-07-76 15:29
80-07-76 15:29
81-07-76 15:29
82-07-76 15:29
83-07-76 15:29
84-07-76 15:29
85-07-76 15:29
86-07-76 15:29
87-07-76 15:29
88-07-76 15:29
89-07-76 15:29
90-07-76 15:29
91-07-76 15:29
92-07-76 15:29
93-07-76 15:29
94-07-76 15:29
95-07-76 15:29
96-07-76 15:29
97-07-76 15:29
98-07-76 15:29
99-07-76 15:29
00-07-76 15:29

000001
160000
122000
000001

001100

000011
000012
000015
000200
177776

177774
177772
177570
177570

000000
000001
000002
000003
000004
000005
000006
000007

```
X
.TITLE MAINDEC-11-DVDRB-A DRV11B INTERPROCESSOR EXERCISER
;#COPYRIGHT (C) 1976
;#DIGITAL EQUIPMENT CORP.
;#MAYNARD, MASS. 01754
;#
;#PROGRAM BY R. MOORE
;#
;#THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
;#PACKAGE (MAINDEC-11-DZQAC-CD), MAR 21, 1976.
;#
$TN=1
$SMR=160000 ;;HALT ON ERROR, LOOP ON TEST, INHIBIT ERROR TYP0UT
$SMR=122000
$TN=1
.SBTTL OPERATIONAL SWITCH SETTINGS
;#
;# SWITCH USE
;# -----
;# 15 HALT ON ERROR
;# 13 INHIBIT ERROR TYPEOUTS
;# 10 BELL ON ERROR
.SBTTL BASIC DEFINITIONS
;#INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
STACK= 1100
.EQUIV ENT,ERROR ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE ;;BASIC DEFINITION OF SCOPE CALL
;#MISCELLANEOUS DEFINITIONS
HT= 11 ;;CODE FOR HORIZONTAL TAB
LF= 12 ;;CODE FOR LINE FEED
CR= 15 ;;CODE FOR CARRIAGE RETURN
CRLF= 200 ;;CODE FOR CARRIAGE RETURN-LINE FEED
PS= 177776 ;;PROCESSOR STATUS WORD
.EQUIV PS,PSM
STKLM= 177774 ;;STACK LIMIT REGISTER
PIRQ= 177772 ;;PROGRAM INTERRUPT REQUEST REGISTER
DSMR= 177570 ;;HARDWARE SWITCH REGISTER
DDISP= 177570 ;;HARDWARE DISPLAY REGISTER
;#GENERAL PURPOSE REGISTER DEFINITIONS
R0= X0 ;;GENERAL REGISTER
R1= X1 ;;GENERAL REGISTER
R2= X2 ;;GENERAL REGISTER
R3= X3 ;;GENERAL REGISTER
R4= X4 ;;GENERAL REGISTER
R5= X5 ;;GENERAL REGISTER
R6= X6 ;;GENERAL REGISTER
R7= X7 ;;GENERAL REGISTER
.EQUIV R6,SP ;;STACK POINTER
.EQUIV R7,PC ;;PROGRAM COUNTER
;#PRIORITY LEVEL DEFINITIONS
```


61
50
49
48
47
46
45
44
43
42
41
40
39
38
37
36
35
34
33
32
31
30
29
28
27
26
25
24
23
22
21
20
19
18
17
16
15
14
13
12
11
10
9
8
7
6
5
4
3
2
1

.EQUIV BIT09,BIT9
.EQUIV BIT08,BIT8
.EQUIV BIT07,BIT7
.EQUIV BIT06,BIT6
.EQUIV BIT05,BIT5
.EQUIV BIT04,BIT4
.EQUIV BIT03,BIT3
.EQUIV BIT02,BIT2
.EQUIV BIT01,BIT1
.EQUIV BIT00,BIT0

000004
000010
000014
000014
000014
000020
000024
000030
000034
000060
000064
000240
106427

;;BASIC "CPU" TRAP VECTOR ADDRESSES
ERRVEC= 4 ;: TIME OUT AND OTHER ERRORS
RESVEC= 10 ;: RESERVED AND ILLEGAL INSTRUCTIONS
TRITVEC= 14 ;: "T" BIT
TRTVEC= 14 ;: TRACE TRAP
BPTVEC= 14 ;: BREAKPOINT TRAP (BPT)
IOTVEC= 20 ;: INPUT/OUTPUT TRAP (IOT) **SCOPE**
PMRVEC= 24 ;: POWER FAIL
ENTVEC= 30 ;: EMULATOR TRAP (ENT) **ERROR**
TRAPVEC=34 ;: "TRAP" TRAP
TKVEC= 60 ;: TTY KEYBOARD VECTOR
TPVEC= 64 ;: TTY PRINTER VECTOR
PIRVEC=240 ;: PROGRAM INTERRUPT REQUEST VECTOR
MTPS=106427 ;: INSTR EQUATE THAT MOVES BYTE TO PSW

.SBTTL TRAP CATCHER

000000

.=0
;:ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
;:SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
;:LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS

000174 000174
000176 000000

.=174
DISPREG: .WORD 0 ;: SOFTWARE DISPLAY REGISTER
SWREG: .WORD 0 ;: SOFTWARE SWITCH REGISTER

000200 000137 001364
000204 000137 001372
000100 000100
000100 000104 000200 000002

.SBTTL STARTING ADDRESS(ES)
JMP @#START1 ;: JUMP TO STARTING ADDRESS OF PROGRAM
JMP @#START2 ;: GO START AS SLAVE COMPUTER
.=100
.WORD 104,200,2 ;: IF 'B EVENT' ON Q-BUS IS CONNECTED
;: JUST DO A RTI (IGNORE IT)

```

463
464
465
466
467
468 001100
469 001100 000000
470 001100 000000
471 001102 000
472 001103 000
473 001104 000000
474 001106 000000
475 001110 000000
476 001112 000000
477 001114 000
478 001115 001
479 001116 000000
480 001120 000000
481 001122 000000
482 001124 000000
483 001126 000000
484 001130 000000
485 001132 000000
486 001134 000
487 001136 000
488 001136 000000
489 001140 177570
490 001142 177570
491 001144 177560
492 001146 177562
493 001150 177564
494 001154 177566
495 001155 000
496 001155 002
497 001155 012
498 001157 000
499 001160 177607 000377
500 001164 077
501 001165 015
502 001166 000012
503

```

.SBTTL COMMON TAGS

```

*****
*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
*USED IN THE PROGRAM.

```

.=1100

```

SCMTAG: .WORD 0
SPASS: .WORD 000
STSTNM: .BYTE 000
SERFLG: .BYTE 000
SICNT: .WORD 000000
SLPADR: .WORD 000000
SLPERR: .WORD 000000
SERTTL: .WORD 000000
SITEMB: .BYTE 001
SERMAX: .BYTE 1
SERRPC: .WORD 000000
SGDADR: .WORD 000000
SEBADR: .WORD 000000
SGOODAT: .WORD 000000
SEBODAT: .WORD 000000
SAUTOB: .BYTE 000
SINTAG: .BYTE 000
SMR: .WORD DSHR
DISPLAY: .WORD DDISP
$TKS: 177560
$TKB: 177562
$TPS: 177564
$TPB: 177566
$NULL: .BYTE 0
$FILLS: .BYTE 2
$FILLC: .BYTE 12
$TPFLG: .BYTE 0
$BELL: .ASCIZ <207><377><377>
$QUES: .ASCIZ '?'
$CARLF: .ASCIZ <15>
$LF: .ASCIZ <12>

```

```

:: START OF COMMON TAGS
:: CONTAINS PASS COUNT
:: CONTAINS THE TEST NUMBER
:: CONTAINS ERROR FLAG
:: CONTAINS SUBTEST ITERATION COUNT
:: CONTAINS SCOPE LOOP ADDRESS
:: CONTAINS SCOPE RETURN FOR ERRORS
:: CONTAINS TOTAL ERRORS DETECTED
:: CONTAINS ITEM CONTROL BYTE
:: CONTAINS MAX. ERRORS PER TEST
:: CONTAINS PC OF LAST ERROR INSTRUCTION
:: CONTAINS ADDRESS OF 'GOOD' DATA
:: CONTAINS ADDRESS OF 'BAD' DATA
:: CONTAINS 'GOOD' DATA
:: CONTAINS 'BAD' DATA
:: RESERVED--NOT TO BE USED

:: AUTOMATIC MODE INDICATOR
:: INTERRUPT MODE INDICATOR

:: ADDRESS OF SWITCH REGISTER
:: ADDRESS OF DISPLAY REGISTER
:: TTY KBD STATUS
:: TTY KBD BUFFER
:: TTY PRINTER STATUS REG. ADDRESS
:: TTY PRINTER BUFFER REG. ADDRESS
:: CONTAINS NULL CHARACTER FOR FILLS
:: CONTAINS # OF FILLER CHARACTERS REQUIRED
:: INSERT FILL CHARS. AFTER A "LINE FEED"
:: "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
:: CODE FOR BELL
:: QUESTION MARK
:: CARRIAGE RETURN
:: LINE FEED

```

.SOTTL ERROR POINTER TABLE

:#THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
:#THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
:#LOCATION SITE#B. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
:#NOTE1: IF SITE#B IS 0 THE ONLY PERTINENT DATA IS (SERAPC).
:#NOTE2: EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:

::* EM ::POINTS TO THE ERROR MESSAGE
::* DH ::POINTS TO THE DATA HEADER
::* DT ::POINTS TO THE DATA
::* DF ::POINTS TO THE DATA FORMAT

001170

SERRTB:
;ERROR

1
EM1
DH1
DT1
0

;;SLAVE DRV11B FAILED TO ECHO DBR CONTENTS
;;ERRPC BUSADR EXPCT RCVD
;;SERAPC SBDADR SGO DAT SBDAT

001170 006056
001172 006503
001174 006576
001176 000000

;ERROR

2
EM2
DH1
DT1
0

;;SLAVE DRV11B FAILED TO ECHO 'STAT' BITS
;;ERRPC BUSADR EXPCT RCVD
;;SERAPC SBDADR SGO DAT SBDAT

001200 006120
001202 006503
001204 006576
001206 000000

;ERROR

3
EM3
DH1
DT1
0

;;FAILED TO INTR ON A 'DATI'
;;ERRPC BUSADR EXPCT RCVD
;;SERAPC SBDADR SGO DAT SBDAT

001210 006161
001212 006576
001214 006576
001216 000000

;ERROR

4
EM4
DH1
DT1
0

;;STATUS ER ON 'DATI'
;;ERRPC BUSADR EXPCT RCVD
;;SERAPC SBDADR SGO DAT SBDAT

001220 006214
001222 006503
001224 006576
001226 000000

000000
000001
000002
000003
000004
000005
000006
000007
000008
000009
000010
000011
000012
000013
000014
000015
000016
000017
000018
000019
000020
000021
000022
000023
000024
000025
000026
000027
000028
000029
000030
000031
000032
000033
000034
000035
000036
000037
000038
000039
000040
000041
000042
000043
000044
000045
000046
000047
000048
000049
000050
000051
000052
000053
000054
000055
000056
000057
000058
000059
000060
000061
000062
000063
000064
000065
000066
000067
000068
000069
000070
000071
000072
000073
000074
000075
000076
000077
000078
000079
000080
000081
000082
000083
000084
000085
000086
000087
000088
000089
000090
000091
000092
000093
000094
000095
000096
000097
000098
000099

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
57
58
59
60
61
62
63
64
65

001320 172410

001322 000124

001324 172410
001326 172412
001328 172414
001330 172416

001334 000124
001336 000126

001340 000000
001342 000000
001344 000000
001346 000001
001348 177740
001350 006610
001352 000111
001354 177740
001356 006610
001360 000113

;DRV11B BASE REGISTER ADDRESS ASSIGNMENT
DRVADR: 172410 ;MODIFY THIS LOC IF DIFFERENT
;DRV11B VECTOR ADDRESS ASSIGNMENT
DRVECT: 124 ;MODIFY THIS LOC IF DIFFERENT
;DRV11B BUS REGISTER ADDRESS POINTERS
DRVMCR: 172410 ;WORD COUNT
DRVBAR: 172412 ;BUFFER ADDRESS
DRVCSR: 172414 ;COMMAND/STATUS
DRVDBR: 172416 ;DATA BUFFER
;DRV11B VECTOR ADDRESS POINTERS
DRVCT0: 124 ;READY & NEX VECTOR
DRVCT2: 126 ;NEW PSM ON INTR
;COMMON PROGRAM LOCATION(S)
TIME: 0 ;GENERAL PURPOSE COUNTER
SAVE: 0 ;REG DATA SAVED HERE
MASTER: 0 ;0=MASTER START - NON-ZERO=SLAVE START
ICOUNT: 1 ;# OF TIMES TO REPEAT ALL TESTS BEFORE END PASS MSG
XPRAM: -32 ;XMIT WORD COUNT
DELUF ;XMIT BUFFER ADRS
111 ;XMIT STATUS/CONTROL
RPRAM: -32 ;RCV WORD COUNT
DELUF ;RCV BUFFER ADRS
113 ;RCV STATUS/CONTROL

```

616 .SBTTL PROGRAM START
617 START1: CLR MSTER ; THIS IT MASTER START
618 BR START ; SKIP NEXT
619 START2: NOV 8-1,MSTER ; SLAVE START
620 START:
621 .SBTTL INITIALIZE THE COMMON TAGS
622 ;; CLEAR THE COMMON TAGS (SCTAG) AREA
623 MOV #SCTAG,R6 ; FIRST LOCATION TO BE CLEARED
624 CLR (R6)+ ; CLEAR MEMORY LOCATION
625 CHP #SMR,R6 ;; DONE?
626 BNE -6 ; LOOP BACK IF NO
627 MOV #STACK,SP ; SETUP THE STACK POINTER
628 ;; INITIALIZE A FEW VECTORS
629 MOV #SCOPE, @IOTVEC ; IOT VECTOR FOR SCOPE ROUTINE
630 MOV #340, @IOTVEC+2 ; LEVEL 7
631 MOV #ERROR, @ENTVEC ; ENT VECTOR FOR ERROR ROUTINE
632 MOV #340, @ENTVEC+2 ; LEVEL 7
633 MOV #TRAP, @TRAPVEC ; TRAP VECTOR FOR TRAP CALLS
634 MOV #340, @TRAPVEC+2 ; LEVEL 7
635 ;; SIZE FOR A HARDWARE SWITCH REGISTER, IF NOT FOUND OR IT IS
636 ;; EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
637 MOV @ERRVEC, -(SP) ; SAVE ERROR VECTOR
638 MOV #648, @ERRVEC ; SET UP ERROR VECTOR
639 MOV #SMR, SMR ; SETUP FOR A HARDWARE SWICH REGISTER
640 MOV #DISP, DISPLAY ; AND A HARDWARE DISPLAY REGISTER
641 CHP 8-1, SMR ; TRY TO REFERENCE HARDWARE SMR
642 BNE 665 ; BRANCH IF NO TIMEOUT TRAP OCCURRED
643 ; AND THE HARDWARE SMR IS NOT = -1
644 BR 655 ; BRANCH IF NO TIMEOUT
645 648: MOV #655, (SP) ; SET UP FOR TRAP RETURN
646 RTI
647 655: MOV #SMREG, SMR ; POINT TO SOFTWARE SMR
648 MOV #DISPREG, DISPLAY
649 665: MOV (SP)+, @ERRVEC ; RESTORE ERROR VECTOR
650 MOV #DRVMCR, R0 ; SET UP REG ADRS POINTERS
651 MOV #DRVADR, R1 ; GET BASE ADRS
652 SETUP2: MOV R1, (R0)+ ; LOAD EN
653 ADD #2, R1
654 CHP #DRVADR+2, R0 ; ALL DONE?
655 BNE SETUP2 ; BR IF NOT
656 MOV #DRVCT0, R0 ; SET UP DRV11B VECTOR ADRS POINTER
657 MOV #DRVCT, R1 ; GET BASE VECTOR ADRS
658 SETUP3: MOV R1, (R0)+
659 ADD #2, R1
660 CHP #DRVCT2+2, R0 ; ALL DONE?
661 BNE SETUP3 ; BR IF NOT
662 .SBTTL TYPE PROGRAM NAME
663 ;; TYPE THE NAME OF THE PROGRAM IF FIRST PASS
664 INC 8-1 ; FIRST TIME?
665 BNE 648 ; BRANCH IF NO
666 TYPE #655 ; TYPE ASCIZ STRING
667 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
668 TST #42 ; ARE WE RUNNING UNDER XXDP/ACT?
669

```

670	001640	001006				BNE	665	:: BRANCH IF YES
671	001642	023727	001140	000176		CMP	SMR, #SWREG	:: SOFTWARE SWITCH REG SELECTED?
672	001650	001005				BNE	675	:: BRANCH IF NO
673	001652	104405				GTSMR		:: GET SOFT-SMR SETTINGS
674	001654	000403				BR	675	
675	001656	112737	000001	001134	665:	MOVB	#1, SAUTOB	:: SET AUTO-MODE INDICATOR
676	001664				675:			
677	001664	000432				BR	645	:: GET OVER THE ASCIZ
678					:: 655:	.ASCIZ	<CRLF>	#MD-11-DVDRB-A DRV11B INTERPROCESSOR EXERCISER #<CRLF>
679	001752				645:			
680	001752	106427	000200			MTPS,	200	:: SET PRIORITY TO HIGHEST LEVEL
681	001756	012706	001100			MOV	#STACK, SP	:: ALWAYS RESET STACK PTR
682	001762	000005				RESET		:: INITIALIZE DRV11B BEFORE TESTING
683	001764	012737	000001	001346		MOV	#1, ICOUNT	:: 1ST TIME DO ALL TEST ONCE - THEN 20000(8)
684	001772	005737	001344			TST	MASTER	:: START AS MASTER?
685	001776	001402				BEQ	15	:: YES - GO SEND TO SLAVE & CHECK ECHO
686	002000	000137	002672			JMP	STST1	:: NO - GO ECHO MASTER'S DATA
687	002004	012777	000400	177316	15:	MOV	#400, #DRVCSR	:: INSURE THAT CYCLE IS SET

688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741

002012
002013 005001
002014 012700 177776
002015 010077 177306
002016 042777 000400 177276
002017 010037 001124
002018 013737 001332 001122
002019 032777 000400 177256
002020 001774
002021 017737 177252 001126
002022 023737 001124 001126
002023 001402
002024 104001
002025 000751
002026 005100 45:
002027 005101 COM
002028 001746 COM
002029 005300 BNE
002030 005200 RSL
002031 103743 INC
002032 012777 177001 177212 BCS
002033 005077 177204 MOV
002034 032777 000400 177176 55:
002132 001774 BEQ

```
.SBTTL MTST1 TEST THAT SLAVE CAN ECHO THE DBR CORRECTLY
*****
*****
MASTER COMPUTER STARTS HERE FROM PROGRAM START
IT SENDS OUT SINGLE WORDS (FLOATING I/O PTRN) THRU THE
DBR TO THE SLAVE COMPUTER
IT LOOKS FOR THE SLAVE TO ECHO THE DBR WORD CORRECTLY
WITHIN A CERTIAN AMOUNT OF TIME
IF IT FAILS TO RETURN THE WORD AN ERROR IS REPORTED
THIS TEST IS NOT EXITED UNTIL ALL DATA PATTERNS HAVE
BEEN SENT AND RECEIVED CORRECTLY
*****
*****
MTST1:
1S: CLR R1 ;R1 SAYS FLOAT 0 LEFT WHEN ZERO
MOV R-2,RO ;START WITH #177776 IN RO
2S: MOV RO,DRVDBR ;SEND PATTERN OUT
BIC #400,DRVCSR ;CLRING CYCLE HERE SETS CYCLE IN SLAVE
MOV RO,$GDOAT ;SAVE IN EXPECTED
MOV DRVDBR,$BDADR ;SET UP DBR ADRS
3S: BIT #400,DRVCSR ;CYCLE SETS WHEN SLAVE ECHOS DATA
BEQ 3S ;WAIT FOR SLAVE TO RESPOND
MOV DRVDBR,$GDOAT ;READ IT BACK
CMP $GDOAT,$BDADR ;CORRECT?
BEQ 4S ;BR IF SO
ERROR 1 ;SLAVE FAILED TO ECHO THE DBR WORD
BR 4S ;LOOP ON ERROR ALWAYS
4S: COM RO,R1 ;CONVERT PTRN TO FLOATING I
COM RO,R1 ;TIME TO FLOAT LEFT?
BNE 5S ;BR IF NOT
RSL RO,R0 ;YES - FLOAT LEFT
INC RO ;KEEP LSB SET
BCS 5S ;BR IF ZERO NOT TO CARRY YET
MOV #177001,DRVDBR ;TELL SLAVE TO GO TO NEXT TEST
CLR DRVCSR ;TELL SLAVE TO READ TEST TERMINATOR
5S: BIT #400,DRVCSR ;IS IT THERE YET?
BEQ 5S ;WAIT FOR IT
```

```
.SBTTL MTST2 TEST THAT SLAVE CAN ECHO THE 'STAT' BITS CORRECTLY
*****
*****
MASTER SENDS OUT A 'FNCT' CODE (1-7) TO THE SLAVE COMPUTER
THE MASTER THEN LOOKS FOR THE SLAVE TO ECHO THE CODE VIA THE
'STAT' BITS WITHIN A CERTIAN AMOUNT OF TIME
IF IT FAILS TO RETURN THE CORRECT CODE AN ERROR IS REPORTED
THIS TEST IS NOT EXITED UNTIL ALL 'FNCT' CODES HAVE BEEN
SENT AND RECEIVED CORRECTLY
*****
*****
```

002134
002134 013737 001330 001122
002142 012700 000002
002146 012737 001602 001124
002154 010077 177150

```
MTST2:
1S: MOV DRVCSR,$BDADR ;SET UP CSR ADRS
MOV #2,RO ;SET UP INITIAL FNCT BIT COUNT
MOV #1602,$GDOAT ;LD EXPECTD
MOV RO,DRVCSR ;LOAD FNCT BITS
```

```

742 002160 032777 000400 177142 25: BIT #400,2DRVCSR ;LOOK FOR SLAVE TO ECHO VIA 'STAT' BITS
743 002166 001774 BEQ 25 ;WAIT ON SLAVE
744 002170 017737 177134 001126 MOV 2DRVCSR,SDDAT ;READ THE CSR
745 002176 023737 001124 001126 CMP SDDAT,SDDAT ;CORRECT?
746 002204 001402 BEQ 35 ;BR IF SO
747 002206 104002 ERROR 2 ;SLAVE FAILED TO ECHO 'STAT' BITS
748 002210 000761 BR 15 ;LOOP ON ERROR ALWAYS
749 002212 062737 001002 001124 35: ADD #1002,SDDAT ;ADVANCE EXPECTED
750 002220 062700 000002 ADD #2,R0 ;ADVANCE PTRN
751 002230 032700 000020 BIT #20,R0 ;ALL BEEN DONE?
752 002232 001751 BEQ 15 ;BR IF NOT
753 002232 012777 177002 177072 MOV #177002,2DRVDBR ;TELL SLAVE TO GO TO NEXT TEST
754 002240 005077 177064 CLR 2DRVCSR ;SET THERE CYCLE
755 002244 032777 000400 177056 45: BIT #400,2DRVCSR ;IS IT THERE YET?
756 002252 001774 BEQ 45 ;WAIT ON SLAVE

.SBTL NTST3 TEST THAT MASTER CAN XMIT A 32 WORD DATA BLOCK TO SLAVE
*****
MASTER XMITTS A 32 WORD BLOCK OF DATA TO SLAVE
THEN CHECKS FOR PROPER INTERRUPT STATUS, MC & BA
THE SLAVE CHECKS THE SAME PLUS THE DATA RECEIVED
THE TEST DOES NOT ADVANCE UNTIL A SUCCESSFUL XFER
*****
NTST3:
15: MTPS, 200 ;NO INTR ALLOWED YET
JSR RS,SETVEC ;SET UP INTR RETURN ADRS
35 ;RETURN TO 35 ON INTR
JSR RS,LDBUF ;GO LOAD 'DBUF' WITH DATA PTRN
-32 ;DO 32 LOCATIONS
MOV #100000,TIME ;SET UP A TIMER VALUE
MOV XPRAM,2DRVCSR ;SET UP WORD COUNT
MOV XPRAM+2,2DRVBAR ;SET UP BUFFER ADRS
MOV XPRAM+4,2DRVCSR ;SET UP CONTROL - IE, FNCT 3 & GO(CLRS CYCLE)
MTPS, 0 ;ALLOW THE RDY INTR
25: DEC TIME ;WAIT FOR INTR
BNE 25 ;UNTIL 0
MOV 2DRVCSR,SDDAT ;READ CSR - SHOULD NEVER GET HERE
BIC #100,2DRVCSR ;DISABLE IE
MOV #5710,SDDAT ;LD EXPECTED - STAT A & C, CYCLE, RDY, IE, FNCT 3
MOV 2DRVCSR,SDDADR ;SET UP CSR ADRS
ERROR 3 ;DATA FAILED TO CAUSE A MC INTR
BR 65 ;GO RESYNC ON ERROR
35: CMP (SP)+,(SP)+ ;FIX STACK SINCE NO RTI
MOV #40,TIME ;SET UP WAIT DELAY
115: BIT #400,2DRVCSR ;IS CYCLE SET?
BEQ 125 ;YES
DEC TIME ;DECREMENT DELAY COUNT
BNE 115
125: MOV 2DRVCSR,SDDAT ;READ STATUS
BIC #100,2DRVCSR ;DISABLE IE
MOV #5710,SDDAT ;LD EXPECTED - STAT A & C, CYCLE, RDY, IE, FNCT 3
CMP SDDAT,SDDAT ;CORRECT?

```

```

798 002450 001405          BEQ      45          ;BR IF SO
799 002452 013737 001330 001122  MOV     DRVCSR, SBDADR ;SET UP CSR ADRS
800 002454 104004          ERROR   4           ;DATI STATUS ERROR
801 002456 000442          BR      65          ;GO RESYNC ON ER
802 002458 005037 001124          CLR     $GDDAT        ;LD EXPECTED MC
803 002470 017737 176630 001126 45:  MOV     2DRVMCR, SBDAT ;READ WORD COUNT
804 002476 001405          BEQ     55          ;BR IF ZERO
805 002480 013737 001324 001122  MOV     DRVPCR, SBDADR ;SET UP PCR ADRS
806 002482 104005          ERROR   5           ;DATI WORD COUNT ERROR
807 002484 000427          BR      65          ;GO RESYNC ON ER
808 002486 013737 001352 001124 55:  MOV     XPRAM+2, $GDDAT ;GET STARTING ADRS OF XFER
809 002488 013700 001350          MOV     XPRAM, R0     ;GET MC
810 002490 005408          NEG     R0            ;GET ACTUAL #
811 002492 006300          ASL    R0            ;CONVERT TO WORD
812 002494 060037 001124          ADD     R0, $GDDAT   ;ADD TO BASE ADRS
813 002496 017737 176566 001126  MOV     2DRVBAR, SBDAT ;READ BAR ADRS
814 002498 042737 000001 001126  BIC     8BIT00, $SDAT ;ELIMINATE LSB
815 002500 023737 001124 001126  CMP     $GDDAT, SBDAT ;CORRECT?
816 002502 001416          BEQ     85          ;BR IF SO
817 002504 013737 001326 001122  MOV     DRVBAR, SBDADR ;SET UP BAR ADRS
818 002506 104006          ERROR   6           ;DATI BUFFER ADRS ERROR
819 002508 012737 000200 001340 65:  MOV     8200, TIME   ;WAIT FOR SLAVE TO COMPLETE DATA CK
820 002510 005337 001340 75:  DEC     TIME         ;COUNT AWAY
821 002512 001375          BNE    75          ;UNTIL DONE
822 002514 112777 000002 176516  MOVB   82, 2DRVCSR   ;TELL SLAVE WE HAVE AN ERROR
823 002516 000620          BR     15          ;REPEAT TEST ON ER
824 002518 012737 000100 001340 85:  MOV     8100, TIME  ;WAIT FOR SLAVE TO COMPLETE CKS
825 002520 005337 001340 205:  DEC     TIME        ;COUNT AWAY
826 002522 001375          BNE    205        ;UNTIL DONE
827 002524 107077 176474          CLRB   2DRVCSR     ;TELL SLAVE ALL OK
828 002526 017737 176470 001342 95:  MOV     2DRVCSR, SAVE ;READ CSR
829 002528 042737 170777 001342  BIC     8170777, SAVE ;SAVE 'STAT' BITS
830 002530 001406          BEQ    105        ;BR IF SLAVE DONE WITH NO ER'S
831 002532 022737 001000 001342  CMP     81000, SAVE  ;WAS THERE AN ER?
832 002534 001365          BNE    95        ;BR IF NOT
833 002536 000137 002254          JMP    15         ;YES - REPEAT TEST ON ER
834 002538 004737 003672 105:  JSR    PC, RSTVEC  ;GO RESTORE VECTOR

```

```

.SBTTL STST1 RECEIVE MASTER'S DBR DATA AND SEND IT BACK VIA DBR
*****
*****
NOW THIS COMPUTER BECOMES THE SLAVE AND ECHOS MASTER'S DBR DATA
SLAVE COMPUTER STARTS HERE FROM PROGRAM START 204
*****
*****

```

```

835 002672          STST1:
836 002672 005077 176432          15:  CLR     2DRVCSR     ;TELL MASTER WE ARE READY
837 002674 032777 000400 176424 25:  BIT     8400, 2DRVCSR ;DATA AVAILABLE?
838 002676 001774          BEQ    25          ;WAIT ON IT
839 002704 017737 176420 001342  MOV     2DRVDBR, SAVE ;GET DATA
840 002706 022737 177001 001342  CMP     8177001, SAVE ;TEST TERMINATOR?
841 002714 001404          BEQ    35          ;BR IF SO
842 002722 013777 001342 176400  MOV     SAVE, 2DRVDBR ;SEND IT BACK
843 002724 000757          BR     15         ;GO LOOK FOR MORE DATA

```

```

850 002734 000240 3S: NOP ;
851
852 .SBTTL STST2 RECEIVE MASTER'S 'STAT' BITS AND SEND IT BACK VIA 'FNCT' BITS
853 *****
854 *****
855 RECEIVE 'STAT' BITS AND CONVERT TO 'FNCT' BITS AND WRITE TO CSR
856 *****
857 *****
858 STST2:
859 002736 042777 000400 176364 1S: BIC 8400,2DRVCSR ; TELL MASTER WE ARE READY
860 002736 032777 000400 176356 2S: BIT 8400,2DRVCSR ; LOOK FOR CYCLE
861 002752 001774 ; BEQ 2S ; WAIT ON IT
862 002754 022777 177002 176350 ; CMP 8177002,2DRVDBR ; TEST TERMINATOR?
863 002762 001412 ; BEQ 3S ; BR IF SO
864 002764 017737 176340 001342 ; MOV 2DRVCSR,SAVE ; READ THE CSR
865 002772 042737 170777 001342 ; BIC 8170777,SAVE ; SAVE ONLY THE STAT BITS
866 003000 113777 001343 176322 ; MOVB SAVE+1,2DRVCSR ; ECHO WITH FNCT BITS
867 003006 000753 ; BR 1S ; LOOK FOR NEXT 'STAT' CODE
868 003010 005077 176314 3S: CLR 2DRVCSR ; TELL MASTER TO CONTINUE
869
870 .SBTTL STST3 RECEIVE A 32 WORD BLOCK OF DATA AND MAKE STATUS & DATA CHECKS
871 *****
872 *****
873 THIS TEST SETS UP TO RECEIVE A 32 WORD BLOCK OF DATA FROM THE MASTER
874 THEN CHECKS FOR PROPER INTERRUPT STATUS, MC, BA & DATA
875 IF AN ERROR IS DETECTED THE SLAVE TELLS THE MASTER, REPORTS THE
876 ERROR, AND RESYNCS ON TEST
877 IF ALL OK THEN THIS COMPUTER BECOMES MASTER AND GOES TO NTST1
878 *****
879 *****
880 STST3:
881 003014 106427 000200 1S: MTPS, 200 ; NO INTRs ALLOWED YET
882 003020 004537 003652 ; JSR RS,SETVEC ; SET UP THE INTR RETURN ADRS
883 003024 003142 ; 3S ; RETURN TO 3S ON DATO INTR
884 003026 004537 003712 ; JSR RS,CLBUF ; GO CLR 'DBUF'
885 003032 177740 ; -32 ; DO 32 LOCATIONS
886 003034 012737 100000 001340 ; MOV 810000,TIME ; SET UP A TIMER VALUE
887 003042 013777 001356 176254 ; MOV RPRM,2DRVCSR ; SET UP WORD COUNT
888 003050 013777 001360 176250 ; MOV RPRM+2,2DRVBAR ; SET UP BUFFER ADRS
889 003056 032777 000400 176244 20S: BIT 8400,2DRVCSR ; LOOK UP CYCLE
890 003064 001774 ; BEQ 20S ; WAIT FOR IT
891 003066 013777 001362 176234 ; MOV RPRM+4,2DRVCSR ; SET UP CONTROL - IE, FNCT 3 & 1 & GO
892 003074 106427 000000 ; MTPS, 0 ; ALLOW THE INTR
893 003100 005337 001340 2S: DEC TIME ; WAIT FOR INTR
894 003104 001375 ; BNE 2S ; UNTIL ZERO
895 003106 017737 176216 001126 ; MOV 2DRVCSR,SBDAT ; READ CSR - SHOULD NEVER GET HERE
896 003114 042777 000100 176206 ; BIC 8100,2DRVCSR ; DISABLE IE
897 003122 012737 004312 001124 ; MOV 84312,SBDAT ; LD EXPECTED - STAT A, RDY, IE & FNCT 3 & 1
898 003130 013737 001330 001122 ; MOV DRVCSR,SBDADR ; SET UP CSR ADRS
899 003136 104007 ; ERROR 7 ; DATO FAILED TO CAUSE A MC INTR
900 003140 000517 ; BR 10S ; GO RESYNC ON ER
901 003142 022626 3S: CMP (SP)+,(SP)+ ; FIX STACK SINCE NO RETURN
902 003144 017737 176160 001126 ; MOV 2DRVCSR,SBDAT ; READ STATUS
903 003152 042777 000100 176150 ; BIC 8100,2DRVCSR ; DISABLE IE

```

K02

904	003160	012737	004312	001124		MOV	#4312,SGDDAT	:LD EXPECTED - STAT A, RDY, IE & FNCT 3 & 1
905	003166	023737	001124	001126		CMF	SGDDAT,SBDDAT	:CORRECT?
906	003174	001405				BEQ	45	:BR IF SO
907	003176	013737	001330	001122		MOV	DRVCSR,SBADR	:SET UP CSR ADRS
908	003204	104010				ERROR	10	:DATO STATUS ERROR
909	003206	000474				BR	105	:GO RESYNC ON ERROR
910	003210	005037	001124		45:	CLR	SGDDAT	:LD EXPECTED MC
911	003214	017737	176104	001126		MOV	DRVCSR,SBDDAT	:READ MCR
912	003222	001405				BEQ	55	:BR IF SO
913	003224	013737	001324	001122		MOV	DRVCSR,SBADR	:SET UP MCR ADRS
914	003232	104011				ERROR	11	:DATO WORD COUNT ERROR
915	003234	000461				BR	105	:GO RESYNC ON ERROR
916	003236	013737	001360	001124	55:	MOV	RPRAM+2,SGDDAT	:GET STARTING ADRS OF XFER
917	003244	013700	001356			MOV	RPRAM,R0	:GET MC
918	003250	005400				NEG	R0	:GET ACTUAL #
919	003252	005300				ASL	R0	:CONVERT TO WORD
920	003254	060037	001124			ADD	R0,SGDDAT	:ADD TO BASE ADRS
921	003260	017737	176042	001126		MOV	DRVBAR,SBDDAT	:READ BAR ADRS
922	003262	042737	000001	001126		BIC	#BIT0,SBDDAT	:ELIMINATE LSB
923	003274	023737	001124	001126		CMF	SGDDAT,SBDDAT	:CORRECT?
924	003302	001405				BEQ	65	:BR IF SO
925	003304	013737	001326	001122		MOV	DRVBAR,SBADR	:SET UP BAR ADRS
926	003312	104012				ERROR	12	:DATO BUFFER ADRS ERROR
927	003314	000431				BR	105	:GO RESYNC ON ERROR
928	003316	013700	001360		65:	MOV	RPRAM+2,R0	:GET BUFFER ADRS
929	003322	013701	001356			MOV	RPRAM,R1	:GET WORD COUNT
930	003326	012702	177776		75:	MOV	#177776,R2	:GET 1ST DATA PTRN (FLOATING 0)
931	003330	005003				CLR	R3	:R3 SAYS WHEN TO SHIFT PTRN
932	003334	020220			85:	CMF	R2,(R0)+	:COMPARE DATA IN DBUF TO EXPECTED
933	003336	001011				BNE	95	:BR IF DATA ERROR
934	003340	005201				INC	R1	:COUNT THE WORD COUNT
935	003342	001431				BEQ	135	:BR IF DATA CHECKS DONE
936	003344	005102				COM	R2	:CONVERT PTRN TO FLOATING 1
937	003346	005103				COM	R3	:TIME TO SHIFT?
938	003350	001371				BNE	85	:BR IF NOT - GO CK NEXT
939	003352	005302				ASL	R2	:FLOAT PTRN LEFT
940	003354	005202				INC	R2	:KEEP LSB SET
941	003356	103363				BCC	75	:GO RESET FLOATING PTRN
942	003360	000765				BR	85	:GO CHECK NEXT
943	003362	014037	001126		95:	MOV	-(R0),SBDDAT	:GET BAD DATA
944	003366	010037	001122			MOV	R0,SBADR	:GET MEM ADRS OF DATA ER
945	003372	010237	001124			MOV	R2,SGDDAT	:LD EXPECTED DATA
946	003376	104013				ERROR	13	:DATO DATA ERROR
947	003400	012737	000100	001340	105:	MOV	#100,TIME	:LET MASTER FINISH ITS CHECKS
948	003406	005337	001340		115:	DEC	TIME	:COUNT AWAY
949	003412	001375			125:	BNE	115	:UNTIL DONE
950	003414	112777	000002	175706		MOVSB	#2,DRVCSR	:TELL MASTER WE HAVE AN ERROR
951	003422	000137	003014			JMP	15	:DO TEST AGAIN ON ERROR
952	003426	105077	175676		135:	CLRB	DRVCSR	:TELL MASTER ALL OK
953	003432	017737	175676	001342	145:	MOV	DRVCSR,SAVE	:READ CSR
954	003440	042737	170777	001342		BIC	#170777,SAVE	:SAVE 'STAT' BITS ONLY
955	003446	001406				BEQ	EOPT	:BR IF MASTER DONE WITH NO ER'S
956	003450	022737	001000	001342		CMF	#1000,SAVE	:DOES MASTER HAVE AN ERROR?
957	003456	001365				BNE	145	:BR IF NOT

```

958 003460 000137 003014      JMP      IS           ;YES - DO TEST AGAIN ON ERROR
959 003464 004737 003672      EOPT:   JSR      PC,RSTVEC ;GO RESTORE VECTOR
960 003470 104406                CKSMR                ;GO CHECK SMR
961 003472 005737 001344                TST      MSTER       ;WERE WE STARTED AS MASTER?
962 003476 001055                BNE     EOPTA        ;BR IF NOT
963 003500 005337 001346                DEC     ICOUNT       ;COUNT TEST PASS
964 003504 001052                BNE     EOPTA        ;BR IF NOT DUE FOR 'END PASS' MSG
965 003506 012737 003000 001346      MOV     #3000,ICOUNT ;RESET PASS COUNT
966
967 .SBTTL  END OF PASS ROUTINE
968
969 ;*****
970 ;INCREMENT THE PASS NUMBER (SPASS)
971 ;TYPE "END PASS #XXXXX" (WHERE XXXX IS A DECIMAL NUMBER)
972 ;IF THERES A MONITOR GO TO IT
973 ;IF THERE ISN'T JUMP TO EOPTA
974
975 SEOP:
976 003514 000240                NOP
977 003516 005037 001102      CLR     $TSTNM       ;ZERO THE TEST NUMBER
978 003522 005237 001100      INC     SPASS        ;INCREMENT THE PASS NUMBER
979 003526 042737 100000 001100    BIC     #100000,SPASS ;DON'T ALLOW A NEG. NUMBER
980 003534 005327                DEC     (PC)+        ;LOOP?
981 SEOPCT: .WORD 1
982 003540 003022                BGT     $DOAGN       ;YES
983 003542 012737                MOV     (PC)+,$2(PC)+ ;RESTORE COUNTER
984 SENDCT: .WORD 1
985 003546 003536                SEOPCT
986 003550 104400 003615      TYPE   SENDNG        ;TYPE "END PASS #"
987 003554 013746 001100      MOV     $SPASS,-(SP) ;SAVE SPASS FOR TYPEOUT
988 003560 104404                TYPDS                ;GO TYPE--DECIMAL ASCII WITH SIGN
989 003562 104400 003612      TYPE   $NULL         ;TYPE A NULL CHARACTER
990 003566 013700 000042    SGET42: MOV     $42,R0  ;GET MONITOR ADDRESS
991 003572 001405                BEQ     $DOAGN       ;BRANCH IF NO MONITOR
992 003574 000005                RESET                ;CLEAR THE WORLD
993 003576 004710                SENDAD: JSR     PC,(R0) ;GO TO MONITOR
994 003580 000240                NOP
995 003584 000240                NOP
996 003588 000240                NOP
997 003596 000240                SDOAGN: NOP
998 003606 000137                JMP     $2(PC)+      ;RETURN
999 003610 003632      SRTNAD: .WORD  EOPTA
1000 003612 377 377 000  SENDNG: .BYTE  -1,-1,0 ;NULL CHARACTER STRING
1001 003615 015 042412 042116  SENDNG: .ASCIZ  <15><12>/END PASS #/
1002 003622 050040 051501 020123
1003 003630 000043
1004 003632 012737 001000 001340  EOPTA:  MOV     #1000,TIME ;SET UP A COUNT
1005 003640 005337 001340  EOPTB:  DEC     TIME       ;STALL FOR OTHER CPU TO BECOME SLAVE
1006 003644 001375                BNE     EOPTB        ;UNTIL TIME=0
1007 003646 000137 002012                JMP     MTST1        ;NOW BECOME MASTER

```

1007
1008
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1060

003652 106427 000200
003656 012577 175452
003662 012777 000200 175446
003670 000205

003672 013777 001336 175434
003700 005077 175432
003704 106427 000200
003710 000207

003712 012500
003714 012701 006610
003720 005021
003722 005200
003724 001375
003726 000205

003730 012500
003732 012701 006610
003736 012702 177776
003742 005003
003744 010221
003746 005200
003750 001001
003752 000205
003754 005102
003756 005103
003760 001371
003762 006302
003764 005202
003766 103363
003770 000765

.SBTTL PROGRAM SUBROUTINES

: THIS ROUTINE SETS THE PRIORITY LEVEL FOR NO INTERRUPT -
: SETS UP THE DRV11B INTERRUPT TO RETURN ON INTERRUPT
: TO THE ADDRESS INDICATED ((RS)) BY THE CALL +2

SETVEC: MTPS, 200 ;SET UP FOR NO INTERRUPT
MOV (RS)+, DRVCT0 ;SET UP INTR RETURN ADRS
MOV #200, DRVCT2 ;KEEP PRIORITY LEVEL AT TOP ON INTR
RTS RS ;EXIT

: THIS ROUTINE RESTORES THE DRV11B INTERRUPT VECTOR TO A HALT
: AND RAISES THE PRIORITY LEVEL

RSTVEC: MOV DRVCT2, DRVCT0 ;POINT VECTOR TO HALT
CLR DRVCT2 ;SET UP HALT
MTPS, 200 ;RAISE PRIORITY LEVEL
RTS PC ;EXIT

: THIS ROUTINE CLEARS THE 'DBUF' BEFORE A 'DATI' XFER
: THE # OF LOCATIONS IN 'DBUF' TO BE CLEARED IS SPECIFIED BY
: THE VALUE IN THE CALL +2 - WHEN ALL CLEARED THE RETURN IS TO
: THE CALL +4

CLDBUF: MOV (RS)+, R0 ;GET THE LOC COUNT
MOV #DBUF, R1 ;GET 1ST ADRS
1S: CLR (R1)+ ;CLR MEM LOC
INC R0 ;COUNT LOC
BNE 1S ;UNTIL ALL DONE
RTS RS ;EXIT

: THIS ROUTINE LOADS 'DBUF' WITH A FLOATING ZERO/ONE PATTERN
: THE # OF LOCATIONS IN 'DBUF' TO BE LOADED IS SPECIFIED BY
: THE VALUE IN THE CALL +2 - WHEN ALL LOADED THE RETURN IS TO CALL +4

LOADBUF: MOV (RS)+, R0 ;GET LOC COUNT
MOV #DBUF, R1 ;GET 1ST ADRS
1S: MOV #177776, R2 ;SET UP FLOATING ZERO PTRN
CLR R3 ;R3 SAYS WHEN TO SHIFT PTRN
2S: MOV R2, (R1)+ ;LOAD MEM WITH PTRN
INC R0 ;COUNT LOC
BNE 1S ;BR IF MORE
RTS R5 ;EXIT
3S: COM R2 ;CONVERT PTRN TO FLOATING 1
COM R3 ;SHOULD WE SHIFT?
BNE 2S ;BR IF NOT - THIS WILL BE A FLOATING 1
ASL R2 ;FLOAT ZERO LEFT
INC R2 ;KEEP LSB SET
BCC 1S ;GO RESET FLOATING PTRN
BR 2S ;GO LOAD NEXT PTRN

.SBTTL SYSNAC ROUTINES

.SBTTL TYPE ROUTINE

1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1110
1111
1112
1113
1114

003772 105737 001157
 003776 100002
 004000 000000
 004002 000407
 004004 010046
 004006 017600 000002
 004012 112046
 004014 001005
 004016 005726
 004020 012600
 004022 062716 000002
 004026 000002
 004030 122716 000011
 004034 001430
 004036 122716 000200
 004042 001006
 004044 005726
 004046 104400
 004050 001165
 004052 105037 004206
 004056 000755
 004060 004737 004142
 004064 123726 001156
 004070 001350
 004072 013746 001154
 004076 105366 000001
 004102 002770
 004104 004737 004142
 004110 105337 004206
 004114 000770
 004116 112716 000040

```

*****
#ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
#THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
#NOTE1:          SNULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
#NOTE2:          SFILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
#NOTE3:          SFILLC CONTAINS THE CHARACTER TO FILL AFTER.
#
#CALL:
#1) USING A TRAP INSTRUCTION
#      TYPE      ,MESADR      ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
#OR
#      TYPE
#      MESADR
#
STYPE:  TSTB      STPFLG      ;; IS THERE A TERMINAL?
        BPL       1S          ;; BR IF YES
        HALT      ;; HALT HERE IF NO TERMINAL
        BR       3S          ;; LEAVE
1S:     MOV       RD, -(SP)    ;; SAVE RD
        MOV       22(SP), RD  ;; GET ADDRESS OF ASCIZ STRING
2S:     MOVVB    (RD)+, -(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
        BNE      4S          ;; BR IF IT ISN'T THE TERMINATOR
        TST      (SP)+       ;; IF TERMINATOR POP IT OFF THE STACK
60S:    MOV      (SP)+, RD    ;; RESTORE RD
3S:     ADD      #2, (SP)     ;; ADJUST RETURN PC
        RTI          ;; RETURN
4S:     CMPB     #HT, (SP)    ;; BRANCH IF <HT>
        BEQ      8S          ;; BRANCH IF NOT <CRLF>
        CMPB     #CRLF, (SP)
        BNE      5S          ;; POP <CR><LF> EQUIV
        TST      (SP)+       ;; TYPE A CR AND LF
        TYPE     SCRLF
        CLRB     SCHARCNT    ;; CLEAR CHARACTER COUNT
        BR       2S          ;; GET NEXT CHARACTER
5S:     JSR      PC, STYPEC   ;; GO TYPE THIS CHARACTER
6S:     CMPB     SFILLC, (SP)+ ;; IS IT TIME FOR FILLER CHARS.?
        BNE      2S          ;; IF NO GO GET NEXT CHAR.
        MOV      SNULL, -(SP) ;; GET # OF FILLER CHARS. NEEDED
        AND      THE NULL CHAR.
7S:     DECB     1(SP)       ;; DOES A NULL NEED TO BE TYPED?
        BLT      6S          ;; BR IF NO--GO POP THE NULL OFF OF STACK
        JSR      PC, STYPEC   ;; GO TYPE A NULL
        DECB     SCHARCNT    ;; DO NOT COUNT AS A COUNT
        BR       7S          ;; LOOP
;HORIZONTAL TAB PROCESSOR
8S:     MOVVB    #' , (SP)   ;; REPLACE TAB WITH SPACE
  
```

```

1115 004122 004737 004142 9S: JSR PC,STYPEC ;; TYPE A SPACE
1116 004126 132737 000007 004206 BITB 87,SCHARCNT ;; BRANCH IF NOT AT
1117 004134 001372 BNE 9S ;; TAB STOP
1118 004136 005726 TST (SP)+ ;; POP SPACE OFF STACK
1119 004140 000724 BR 25 ;; GET NEXT CHARACTER
1120 004142 105777 175002 STYPEC: TSTB 2STPS ;; WAIT UNTIL PRINTER IS READY
1121 004146 100375 BPL STYPEC
1122 004150 116677 000002 174774 NOVB 2(SP),2STPB ;; LOAD CHAR TO BE TYPED INTO DATA REG.
1123 004156 122766 000015 000002 CMPB BCR,2(SP) ;; IS CHARACTER A CARRIAGE RETURN?
1124 004164 001003 BNE 18 ;; BRANCH IF NO
1125 004166 105037 004206 CLAB SCHARCNT ;; YES—CLEAR CHARACTER COUNT
1126 004172 000406 JR STYPEX ;; EXIT
1127 004174 122766 000012 000002 18: CMPB BLF,2(SP) ;; IS CHARACTER A LINE FEED?
1128 004202 001402 BEQ STYPEX ;; BRANCH IF YES
1129 004204 105227 INCB (PC)+ ;; COUNT THE CHARACTER
1130 004206 000000 SCHARCNT: WORD 0 ;; CHARACTER COUNT STORAGE
1131 004210 000207 STYPEX: RTS PC

```

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

```

*****
#THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
#OCTAL (ASCII) NUMBER AND TYPE IT.
#STYPOS—ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
#CALL:

```

```

#   NOV   NUM,-(SP)   ;; NUMBER TO BE TYPED
#   TYPOS   ;; CALL FOR TYPEOUT
#   .BYTE  N   ;; N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
#   .BYTE  M   ;; M=1 OR 0
#                               ;; 1=TYPE LEADING ZEROS
#                               ;; 0=SUPPRESS LEADING ZEROS

```

```

#STYPON—ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
#STYPOS OR STYPOC

```

```

#CALL:
#   NOV   NUM,-(SP)   ;; NUMBER TO BE TYPED
#   TYPON   ;; CALL FOR TYPEOUT

```

```

#STYPOC—ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER

```

```

#CALL:
#   NOV   NUM,-(SP)   ;; NUMBER TO BE TYPED
#   TYPOC   ;; CALL FOR TYPEOUT

```

```

1155 004212 017646 000000 STYPOS: NOV 2(SP),-(SP) ;; PICKUP THE MODE
1156 004216 116637 000001 004435 NOVB 1(SP),SOFILL ;; LOAD ZERO FILL SWITCH
1157 004224 112637 004437 NOVB (SP)+,SOMODE+1 ;; NUMBER OF DIGITS TO TYPE
1158 004230 052716 000002 ADD 82,(SP) ;; ADJUST RETURN ADDRESS
1159 004234 000406 BR STYPON
1160 004236 112737 000001 004435 STYPOC: NOVB 81,SOFILL ;; SET THE ZERO FILL SWITCH
1161 004238 112737 000006 004437 NOVB 86,SOMODE+1 ;; SET FOR SIX(6) DIGITS
1162 004242 112737 000005 004434 STYPON: NOVB 85,SOCNT ;; SET THE ITERATION COUNT
1163 004244 010346 NOV R3,-(SP) ;; SAVE R3
1164 004246 010446 NOV R4,-(SP) ;; SAVE R4
1165 004248 010546 NOV R5,-(SP) ;; SAVE R5

```

```

1169 004266 113704 004437      MOVB      SOCODE+1,R4      ;;GET THE NUMBER OF DIGITS TO TYPE
1170 004272 005404      NEG       R4
1171 004274 002704 000006      ADD      R6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
1172 004300 110437 004436      MOVB      R4,SOCODE     ;;SAVE IT FOR USE
1173 004304 113704 004435      MOVB      SOFILL,R4     ;;GET THE ZERO FILL SWITCH
1174 004310 016606 000012      MOV      12(SP),R5     ;;PICKUP THE INPUT NUMBER
1175 004314 005003      CLR      R3           ;;CLEAR THE OUTPUT WORD
1176 004316 006106      ROL      R3           ;;ROTATE MSB INTO "C"
1177 004320 000406      BR      R3           ;;GO DO MSB
1178 004322 006106      ROL      R3           ;;FORM THIS DIGIT
1179 004324 006106      ROL      R3
1180 004326 006106      ROL      R3
1181 004328 006106      ROL      R3
1182 004330 010503      MOV      R3,R3         ;;GET LSB OF THIS DIGIT
1183 004332 006106      ROL      R3           ;;TYPE THIS DIGIT?
1184 004334 000006 004436      DECB     SOCODE        ;;TYPE THIS DIGIT?
1185 004336 000006 177770      BPL     R3           ;;BR IF NO
1186 004338 000006      BIC     R3,R3         ;;GET RID OF JUNK
1187 004340 000006      BNE     R3,R3         ;;TEST FOR 0
1188 004342 000006      TST     R3           ;;SUPPRESS THIS 0?
1189 004344 000006      BEQ     R3,R3         ;;BR IF YES
1190 004346 000006      INC     R3           ;;DON'T SUPPRESS ANYMORE 0'S
1191 004348 000006 000060      BIS     R3,R3         ;;MAKE THIS DIGIT ASCII
1192 004350 000006 000040      BIS     R3,R3         ;;MAKE ASCII IF NOT ALREADY
1193 004352 110337 004432      MOVB      R3,R5       ;;SAVE FOR TYPING
1194 004354 104400 004432      TYPE     R5           ;;GO TYPE THIS DIGIT
1195 004356 106337 004434      DECB     SOCNT        ;;COUNT BY 1
1196 004358 000347      BGT     R3,R3         ;;BR IF MORE TO DO
1197 004360 000240      BLT     R3,R3         ;;BR IF DONE
1198 004362 000520      INC     R3           ;;INSURE LAST DIGIT ISN'T A BLANK
1199 004364 000744      BR      R3           ;;GO DO THE LAST DIGIT
1200 004366 012606      MOV      (SP)+,R5     ;;RESTORE R5
1201 004368 012606      MOV      (SP)+,R4     ;;RESTORE R4
1202 004370 012606      MOV      (SP)+,R3     ;;RESTORE R3
1203 004372 016606 000002 000004      MOV      2(SP),4(SP)  ;;SET THE STACK FOR RETURNING
1204 004374 000006      MOV      (SP)+,(SP)
1205 004376 000006      RTI
1206 004378 000006      .BYTE   0           ;;RETURN
1207 004380 000006      .BYTE   0           ;;STORAGE FOR ASCII DIGIT
1208 004382 000006      .BYTE   0           ;;TERMINATOR FOR TYPE ROUTINE
1209 004384 000006      .BYTE   0           ;;OCTAL DIGIT COUNTER
1210 004386 000006      .WORD   0           ;;ZERO FILL SWITCH
1211 004388 000006      .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
1212
1213 *****
1214 #THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
1215 #SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
1216 #NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
1217 #BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
1218 #REPLACED WITH SPACES.
1219 #CALL:
1220 #     MOV     NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
1221 #     TYPDS
1222 #
STYPS:
004440

```

```

1293 004440 010046 MOV R0, -(SP)
1294 004441 010146 MOV R1, -(SP)
1295 004442 010246 MOV R2, -(SP)
1296 004443 010346 MOV R3, -(SP)
1297 004444 010446 MOV R4, -(SP)
1298 004445 010546 MOV R5, -(SP)
1299 004446 012746 020200 MOV R0, 20200, -(SP)
1300 004447 015646 000020 MOV R0, 20(SP), R5
1301 004448 100046 BPL R5
1302 004449 005446 NEG R5
1303 004450 112746 000055 000001 MOV R0, 55, 1(SP)
1304 004451 005046 15: CLR R0
1305 004452 012746 004654 MOV R0, SDBLK, R3
1306 004453 112746 000040 MOV R0, (R3)+
1307 004454 005046 25: CLR R2
1308 004455 016001 004644 MOV R0, SOTBL(R0), R1
1309 004456 160101 35: SUB R1, R5
1310 004457 005446 BLT R5
1311 004458 000774 INC R5
1312 004459 005010 45: BR R5
1313 004460 005702 ADD R1, R5
1314 004461 001002 TST R5
1315 004462 105716 BNE (SP)
1316 004463 100407 TSTB (SP)
1317 004464 105316 BMI RSLB (SP)
1318 004465 103003 BCC 65
1319 004466 116663 000001 177777 MOV R0, 1(SP), -1(R3)
1320 004467 000060 65: BIS R0, R2
1321 004468 000040 75: BIS R0, R2
1322 004469 110223 MOV R2, (R3)+
1323 004470 005720 TST (R0)+
1324 004471 000027 000010 CIP R5, #10
1325 004472 005724 BLT R5
1326 004473 010501 BGT R5, R2
1327 004474 005724 BR R5
1328 004475 105726 85: TSTB (SP)+
1329 004476 100003 BPL R5
1330 004477 116663 177777 177776 MOV R0, -1(SP), -2(R3)
1331 004478 105013 95: CLRB (R3)
1332 004479 012608 MOV (SP)+, R5
1333 004480 012603 MOV (SP)+, R3
1334 004481 012602 MOV (SP)+, R2
1335 004482 012601 MOV (SP)+, R1
1336 004483 104400 004654 MOV (SP)+, R0
1337 004484 016666 000002 000004 TYPE SDBLK
1338 004485 012616 MOV R0, 2(SP), 4(SP)
1339 004486 000020 MOV (SP)+, (SP)
1340 004487 023420 SOTBL: 10000.
1341 004488 0001750 1000.
1342 004489 000144 100.
1343 004490 000012 10.
1344 004491 000004 SDBLK: .BLKN 4

```

```

:: PUSH R0 ON STACK
:: PUSH R1 ON STACK
:: PUSH R2 ON STACK
:: PUSH R3 ON STACK
:: PUSH R4 ON STACK
:: PUSH R5 ON STACK
:: SET BLANK SWITCH AND SIGN
:: GET THE INPUT NUMBER
:: BR IF INPUT IS POS.
:: MAKE THE BINARY NUMBER POS.
:: MAKE THE ASCII NUMBER NEG.
:: ZERO THE CONSTANTS INDEX
:: SETUP THE OUTPUT POINTER
:: SET THE FIRST CHARACTER TO A BLANK
:: CLEAR THE BCD NUMBER
:: GET THE CONSTANT
:: FORM THIS BCD DIGIT
:: BR IF DONE
:: INCREASE THE BCD DIGIT BY 1

:: ADD BACK THE CONSTANT
:: CHECK IF BCD DIGIT=0
:: FALL THROUGH IF 0
:: STILL DOING LEADING 0'S?
:: BR IF YES
:: MSD?
:: BR IF NO
:: YES--SET THE SIGN
:: MAKE THE BCD DIGIT ASCII
:: MAKE IT A SPACE IF NOT ALREADY A DIGIT
:: PUT THIS CHARACTER IN THE OUTPUT BUFFER
:: JUST INCREMENTING
:: CHECK THE TABLE INDEX
:: GO DO THE NEXT DIGIT
:: GO TO EXIT
:: GET THE LSD
:: GO CHANGE TO ASCII
:: WAS THE LSD THE FIRST NON-ZERO?
:: BR IF NO
:: YES--SET THE SIGN FOR TYPING
:: SET THE TERMINATOR
:: POP STACK INTO R5
:: POP STACK INTO R3
:: POP STACK INTO R2
:: POP STACK INTO R1
:: POP STACK INTO R0
:: NOW TYPE THE NUMBER
:: ADJUST THE STACK

:: RETURN TO USER

```

1277
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1288
1289
1290
1291
1292
1293
1294
1295
1296
1297
1298
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1329
1330
1331
1332
1333

.SBTTL ERROR HANDLER ROUTINE

: THIS ROUTINE WILL INCREMENT THE ERROR FLAG, AND THE ERROR COUNT,
: SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
: AND GO TO SWACK ON ERROR
: THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
: SM15=1 HALT ON ERROR
: SM13=1 INHIBIT ERROR TYPEOUTS
: SM10=1 BELL ON ERROR
: CALL
: ERROR N ;;ERROR=ENT AND N=ERROR ITEM NUMBER

SERROR:

CKSMR ;: TEST FOR CHANGE IN SOFT-SMR
CKSMR ;GO LOOK FOR SMR CHANGE
7S: INCB SERFLG ;: SET THE ERROR FLAG
BEQ 7S ;: DON'T LET THE FLAG GO TO ZERO
MOV STSTNM,2DISPLAY ;: DISPLAY TEST NUMBER AND ERROR FLAG
BIT @BIT10,2SMR ;: BELL ON ERROR?
BEQ IS ;: NO - SKIP
TYPE ,SBELL ;: RING BELL
1S: INC @ENTTL ;: COUNT THE NUMBER OF ERRORS
MOV (SP),SERAPC ;: GET ADDRESS OF ERROR INSTRUCTION
SUB @SERAPC ;: STRIP AND SAVE THE ERROR ITEM CODE
MOV@ @SERAPC,SITEMB ;: SKIP TYPEOUT IF SET
BIT @BIT13,2SMR ;: SKIP TYPEOUTS
BNE 20S ;: GO TO USER ERROR ROUTINE
JSR PC,SWACK
TYPE ,SCLF
20S:
2S: TST 2SMR ;: HALT ON ERROR
BPL 3S ;: SKIP IF CONTINUE
HALT ;: HALT ON ERROR!
CKSMR ;: TEST FOR CHANGE IN SOFT-SMR
3S:
RTI ;: RETURN

: GO TYPE ERROR
: GO UPDATE SOFTWARE SMR IF 'CNTRL/G'

SWACK: JSR PC,SERRTYP ;: GO TYPE ERROR
CKSMR ;: GO LOOK FOR SMR CHANGE
RTS PC ;: RETURN TO ERROR HANDLER

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

: THIS ROUTINE USES THE "ITEM CONTROL BYTE" (SITEMB) TO DETERMINE WHICH
: ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" (SERRTB),
: AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.

SERRTYP:
TYPE SCLF ;: "CARRIAGE RETURN" & "LINE FEED"
MOV RD,-(SP) ;: SAVE RD

004664 104406
004664 104406
004666 104406
004670 105237 001103
004674 001775
004676 013777 001102 174236
004704 032777 002000 174226
004712 001402
004714 104400 001160
004720 005237 001112
004724 011637 001116
004730 162737 000002 001116
004736 117737 174154 001114
004744 032777 020000 174166
004752 001004
004754 004737 005000
004760 104400 001165
004764
004764 005777 174150
004770 100002
004772 000000
004774 104406
004776
004776 000002
005000 004737 005010
005004 104406
005006 000207
005010
005010 104400 001165
005014 010046


```

1385 005164 005737 177060      TST      @177060      :: TIME OUT ON XOR?
1386 005170 012637 000004      MOV      (SP)+,@ERRVEC  :: RESTORE THE ERROR VECTOR
1387 005174 000404      BR       $SVLAD      :: GO TO THE NEXT TEST
1388 005176 022626      5S:     CMP      (SP)+,(SP)+  :: CLEAR THE STACK AFTER A TIME OUT
1389 005200 012637 000004      MOV      (SP)+,@ERRVEC  :: RESTORE THE ERROR VECTOR
1390 005204 000406      BR       $OVER      :: LOOP ON THE PRESENT TEST
1391 005206      6S:     ::#####END OF CODE FOR THE XOR TESTER#####
1392 005206 105237 001102      $SVLAD: INCB     $STNM      :: COUNT TEST NUMBERS
1393 005212 011637 001106      MOV      (SP), $LADR    :: SAVE SCOPE LOOP ADDRESS
1394 005216 105037 001103      CLRB    $ERFLG      :: ZERO THE ERROR FLAG
1395 005222 013777 001102 173712 $OVER:  MOV      $STNM,$DISPLAY :: DISPLAY TEST NUMBER
1396 005230 013716 001106      MOV      $LADR,(SP)    :: FUDGE RETURN ADDRESS
1397 005234 000002      RTI                          :: FIXES PS
1398      .SBTTL  TTY INPUT ROUTINE
1399
1400      ::#####
1401      .ENABL  LSB
1402
1403      ::#####
1404      #SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
1405      #ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
1406      #SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
1407      #WHEN OPERATING IN TTY FLAG MODE.
1408 005236 022737 000176 001140 $CKSMR:  CMP      @SMREG,$SMR    :: IS THE SOFT-SMR SELECTED?
1409 005244 001074      BNE     $SS      :: BRANCH IF NO
1410 005246 105777 173672      TSTB   @STKS      :: CHAR THERE?
1411 005252 100071      BPL     $SS      :: IF NO, DON'T WAIT AROUND
1412 005254 117746 173666      MOVB   @STKB,-(SP)  :: SAVE THE CHAR
1413 005260 042716 177600      BIC    @1C177,(SP)  :: STRIP-OFF THE ASCII
1414 005264 022726 000007      CMP    @7,(SP)+    :: IS IT A CONTROL G?
1415 005270 001053      BNE     $SS      :: NO, RETURN TO USER
1416 005272 123727 001134 000001      CMPEB @AUTOB,@1    :: ARE WE RUNNING IN AUTO-MODE?
1417 005300 001456      BEQ    $SS      :: BRANCH IF YES
1418
1419 005302 104400 005763      TYPE   , $CNTLG     :: ECHO THE CONTROL-G (!G)
1420 005306 104400 005770      SGTSMR: TYPE   , $SMR      :: TYPE CURRENT CONTENTS
1421 005312 013746 000176      MOV    $SMREG,-(SP)  :: SAVE SMREG FOR TYPEOUT
1422 005316 104401      TYPOC      :: GO TYPE--OCTAL ASCII(ALL DIGITS)
1423 005320 104400 006001      TYPE   , $SMEN     :: PROMPT FOR NEW SMR
1424 005324 005046      19S:   CLR    -(SP)    :: CLEAR COUNTER
1425 005326 005046      CLR    -(SP)    :: THE NEW SMR
1426 005330 105777 173610      7S:   TSTB   @STKS      :: CHAR THERE?
1427 005334 100375      BPL     7S      :: IF NOT TRY AGAIN
1428
1429 005336 117746 173604      MOVB   @STKB,-(SP)  :: PICK UP CHAR
1430 005342 042716 177600      BIC    @1C177,(SP)  :: MAKE IT 7-BIT ASCII
1431
1432
1433
1434 005346 021627 000025      9S:   CMP    (SP),@25    :: IS IT A CONTROL-U?
1435 005352 001005      BNE     10S      :: BRANCH IF NOT
1436 005354 104400 005756      TYPE   , $CNTLU     :: YES, ECHO CONTROL-U (!U)
1437 005360 062706 000006      20S:  ADD    @6,SP      :: IGNORE PREVIOUS INPUT
1438 005364 000757      BR     19S      :: LET'S TRY IT AGAIN

```

```

1439
1440
1441 005366 021627 000015      10S:  CMP      (SP),#15      :: IS IT A <CR>?
1442 005372 001022                BNE      16S           :: BRANCH IF NO
1443 005374 005766 000004      TST      4(SP)        :: YES, IS IT THE FIRST CHAR?
1444 005400 001403                BEQ      11S           :: BRANCH IF YES
1445 005402 016677 000002 173530  NOV      2(SP),25HR   :: SAVE NEW SMR
1446 005410 062706 000006      11S:  ADD      #6,SP        :: CLEAR UP STACK
1447 005414 104400 001165      14S:  TYPE    $CALF      :: ECHO <CR> AND <LF>
1448 005420 123727 001135 000001  CMPB    $INTAG,#1    :: RE-ENABLE TTY KBD INTERRUPTS?
1449 005430 001003                BNE      15S           :: BRANCH IF NOT
1450 005438 012777 000100 173506  NOV      #100,25TKS  :: RE-ENABLE TTY KBD INTERRUPTS
1451 005438 000002      15S:  RTI                    :: RETURN
1452 005440 004737 004142      16S:  JSR      PC,STYEC   :: ECHO CHAR
1453 005444 021627 000060  CMP      (SP),#60    :: CHAR < 0?
1454 005450 002420                BLT      18S           :: BRANCH IF YES
1455 005458 021627 000067  CMP      (SP),#67    :: CHAR > 7?
1456 005466 003015                BGT      18S           :: BRANCH IF YES
1457 005466 042726 000060  BIC      #60,(SP)+   :: STRIP-OFF ASCII
1458 005464 005766 000002  TST      2(SP)       :: IS THIS THE FIRST CHAR
1459 005470 001403                BEQ      17S           :: BRANCH IF YES
1460 005472 006316                ASL      (SP)         :: NO, SHIFT PRESENT
1461 005474 006316                ASL      (SP)         :: CHAR OVER TO MAKE
1462 005476 006316                ASL      (SP)         :: ROOM FOR NEW ONE.
1463 005500 005266 000002      17S:  INC      2(SP)       :: KEEP COUNT OF CHAR
1464 005504 056616 177776  BIS      -2(SP),(SP) :: SET IN NEW CHAR
1465 005510 000707                BR                    :: GET THE NEXT ONE
1466 005512 104400 001164      18S:  TYPE    $QUES      :: TYPE ?<CR><LF>
1467 005516 000720                BR      20S          :: SIMULATE CONTROL-U
1468
1469
1470
1471
1472
1473
1474
1475
1476
1477
1478
1479 005520 011646 000004 000002  SRDCHR: MOV      (SP),-(SP)  :: PUSH DOWN THE PC
1480 005522 016666 000004 000002  MOV      4(SP),2(SP)  :: SAVE THE PS
1481 005530 105777 173410      1S:  TSTB   25TKS        :: WAIT FOR
1482 005534 100375                BPL      1S           :: A CHARACTER
1483 005536 117766 173404 000004  MOVB    25TKB,4(SP)   :: READ THE TTY
1484 005544 042766 177600 000004  BIC     #1C(177),4(SP) :: GET RID OF JUNK IF ANY
1485 005552 026627 000004 000023  CMP      4(SP),#23    :: IS IT A CONTROL-S?
1486 005560 001013                BNE      2S           :: BRANCH IF NO
1487 005562 105777 173356      2S:  TSTB   25TKS        :: WAIT FOR A CHARACTER
1488 005566 100375                BPL      2S           :: LOOP UNTIL ITS THERE
1489 005570 117746 173352  MOVB    25TKB,-(SP)   :: GET CHARACTER
1490 005574 042716 177600  BIC     #1C177,(SP)  :: MAKE IT 7-BIT ASCII
1491 005600 022627 000021  CMP      (SP)+,#21    :: IS IT A CONTROL-Q?
1492 005604 001366                BNE      2S           :: IF NOT DISCARD IT

```

 #THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY

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

```

1493 005606 000750          BR      15          ;; YES, RESUME
1494 005610 026627 000004 000140 3S:  CMP      4(SP),R140  ;; IS IT UPPER CASE?
1495 005616 002407          BLT      4S          ;; BRANCH IF YES
1496 005620 026627 000004 000175          CMP      4(SP),R175  ;; IS IT A SPECIAL CHAR?
1497 005626 003003          BGT      4S          ;; BRANCH IF YES
1498 005630 042766 000040 000004          BIC      R40,R4(SP)  ;; MAKE IT UPPER CASE
1499 005636 000002          RTI          ;; GO BACK TO USER
1500
1501 *****
1502 *THIS ROUTINE WILL INPUT A STRING FROM THE TTY
1503 *CALL:
1504 *
1505 *   RDLIN          ;; INPUT A STRING FROM THE TTY
1506 *   RETURN HERE  ;; ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
1507 *               ;; TERMINATOR WILL BE A BYTE OF ALL 0'S
1508
1507 005640 010346          SRDLIN: MOV      R3,-(SP)  ;; SAVE R3
1508 005642 012703 005746 1S:  MOV      @STTYIN,R3  ;; GET ADDRESS
1509 005646 022703 005756 2S:  CMP      @STTYIN+@B.,R3  ;; BUFFER FULL?
1510 005652 101405          BLOS     4S          ;; BR IF YES
1511 005654 104407          RDCHR          ;; GO READ ONE CHARACTER FROM THE TTY
1512 005656 112613          MOVW    (SP)+,(R3)  ;; GET CHARACTER
1513 005660 122713 000177 10S: CMPB    @177,(R3)  ;; IS IT A RUBOUT
1514 005664 001003          BNE     3S          ;; SKIP IF NOT
1515 005666 104400 001164 4S:  TYPE    @QUES      ;; TYPE A '?'
1516 005672 000763          BR      1S          ;; CLEAR THE BUFFER AND LOOP
1517 005674 111337 005744 3S:  MOVW    (R3),@9S    ;; ECHO THE CHARACTER
1518 005700 104400 005744          TYPE    @9S
1519 005704 122723 000015          CMPB    @15,(R3)+  ;; CHECK FOR RETURN
1520 005710 001356          BNE     2S          ;; LOOP IF NOT RETURN
1521 005712 105063 177777          CLRB    -1(R3)     ;; CLEAR RETURN (THE 15)
1522 005716 104400 001166          TYPE    @LF        ;; TYPE A LINE FEED
1523 005722 012603          MOV     (SP)+,R3   ;; RESTORE R3
1524 005724 011646          MOV     (SP)-,(SP) ;; ADJUST THE STACK AND PUT ADDRESS OF THE
1525 005726 016666 000004 000002          MOV     4(SP),2(SP) ;; FIRST ASCII CHARACTER ON IT
1526 005734 012766 005746 000004          MOV     @STTYIN,4(SP)
1527 005742 000002          RTI
1528 005744          000          ;; RETURN
1529 005745          000          ;; STORAGE FOR ASCII CHAR. TO TYPE
1530 005746 000010          STTYIN: .BLKB     8  ;; TERMINATOR
1531 005756 052536 005015 000  SCNTLU: .ASCIZ    /?U/<15><12>  ;; RESERVE 8 BYTES FOR TTY INPUT
1532 005763 136 006507 000012  SCNTLG: .ASCIZ    /?G/<15><12>  ;; CONTROL "U"
1533 005770 005015 053523 020122  SPSWR: .ASCIZ    <15><12>/SWR = /  ;; CONTROL "G"
1534 005776 020075          000
1535 006001 040 047040 053E05  SNEW:  .ASCIZ    / NEW = /
1536 006006 036440 000040
1537
1538 .EVEN
1539 .SBTTL TRAP DECODER
1540
1541 *****
1542 *THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
1543 *AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
1544 *OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
1545 *GO TO THAT ROUTINE.
1546
1546 006012 010046          STRAP: MOV     R0,-(SP)  ;;SAVE R0

```

```

1547 006014 016600 000002      MOV      2(SP),RO      ;; GET TRAP ADDRESS
1548 006020 005740          TST      -(RO)        ;; BACKUP BY 2
1549 006022 111000          MOVB    (RO),RO      ;; GET RIGHT BYTE OF TRAP
1550 006024 006300          ASL     RO           ;; POSITION FOR INDEXING
1551 006026 016000 006034      MOV     $TRPAD(RO),RO ;; INDEX TO TABLE
1552 006032 000200          RTS     RO           ;; GO TO ROUTINE

```

.SBTTL TRAP TABLE

;; THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
;; BY THE "TRAP" INSTRUCTION.

ROUTINE

STRPAD:

```

STYPER ;; CALL=TYPER TRAP+0(104400) TTY TYPEOUT ROUTINE
STYPOC ;; CALL=TYPOC TRAP+1(104401) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
STYPOS ;; CALL=TYPOS TRAP+2(104402) TYPE OCTAL NUMBER (NO LEADING ZEROS)
STYPON ;; CALL=TYPON TRAP+3(104403) TYPE OCTAL NUMBER (AS PER LAST CALL)
STYPDS ;; CALL=TYPDS TRAP+4(104404) TYPE DECIMAL NUMBER (WITH SIGN)

SGTSMR ;; CALL=GTSMR TRAP+5(104405) GET SOFT-SMR SETTING

SCKSMR ;; CALL=CKSMR TRAP+6(104406) TEST FOR CHANGE IN SOFT-SMR
SRDCHR ;; CALL=RDCHR TRAP+7(104407) TTY TYPEIN CHARACTER ROUTINE
SRDLIN ;; CALL=RDLIN TRAP+10(104410) TTY TYPEIN STRING ROUTINE

```

.SBTTL ASCII MESSAGES

EM1: .ASCIZ /SLAVE FAILED TO ECHO DBR CONTENTS/

EM2: .ASCIZ /SLAVE FAILED TO ECHO 'STAT' BITS/

EM3: .ASCIZ /FAILED TO INTR ON A 'DATI'/

EM4: .ASCIZ /STATUS ER ON 'DATI'/

EM5: .ASCIZ /WORD COUNT ER ON 'DATI'/

EM6: .ASCIZ /BUFFER ADRS ER ON 'DATI'/

```

1573 006056 046123 053101 020106
1574 006058 040502 046111 042105
1575 006060 052040 020117 041506
1576 006062 047510 042040 051102
1577 006064 041440 047117 042524
1578 006066 046111 043101 020106
1579 006068 040502 020117 042105
1580 006070 047510 020117 041506
1581 006072 051524 020047 044502
1582 006074 000000 044501 042514
1583 006076 020104 047524 044440
1584 006078 040502 020122 047117
1585 006080 044524 020101 040504
1586 006082 052123 052101 051525
1587 006084 042440 020122 047117
1588 006086 023440 040504 044524
1589 006088 000047 042122 041440
1590 006090 047527 052116 042440
1591 006092 020122 047117 023440
1592 006094 040504 044524 000047
1593 006096 052502 043106 051105
1600

```

```

1601 006276 040440 051104 020123
1602 006304 051105 047440 020116
1603 006312 042047 052101 023511
1604 000000 000000 000000 000000
1605 000000 000000 000000 000000
1606 000000 000000 000000 000000
1607 000000 000000 000000 000000
1608 000000 000000 000000 000000
1609 000000 000000 000000 000000
1610 000000 000000 000000 000000
1611 000000 000000 000000 000000
1612 000000 000000 000000 000000
1613 000000 000000 000000 000000
1614 000000 000000 000000 000000
1615 000000 000000 000000 000000
1616 000000 000000 000000 000000
1617 000000 000000 000000 000000
1618 000000 000000 000000 000000
1619 000000 000000 000000 000000
1620 000000 000000 000000 000000
1621 000000 000000 000000 000000
1622 000000 000000 000000 000000
1623 000000 000000 000000 000000
1624 000000 000000 000000 000000
1625 000000 000000 000000 000000
1626 000000 000000 000000 000000
1627 000000 000000 000000 000000
1628 000000 000000 000000 000000
1629 000000 000000 000000 000000
1630 000000 000000 000000 000000
1631 000000 000000 000000 000000
1632 000000 000000 000000 000000
1633 000000 000000 000000 000000
1634 000000 000000 000000 000000
1635 000000 000000 000000 000000
1636 000000 000000 000000 000000
1637 000000 000000 000000 000000
1638 000000 000000 000000 000000
1639 000000 000000 000000 000000
1640 000000 000000 000000 000000
1641 000000 000000 000000 000000
1642 000000 000000 000000 000000
1643 000000 000000 000000 000000
1644 006610 000000 000000 000000
1645 000000 000000 000000 000000

```

EM7: .ASCIZ /FAILED TO INTR ON A 'DATO'/'

EM10: .ASCIZ /STATUS ER ON 'DATO'/'

EM11: .ASCIZ /WORD COUNT ER ON 'DATO'/'

EM12: .ASCIZ /BUFFER ADRS ER ON 'DATO'/'

EM13: .ASCIZ /DATA ER ON 'DATO'/'

DH1: .ASCIZ /ERRPC BUSADR EXPCT RCVD/

DH2: .ASCIZ /ERRPC MEMADR EXPCT RCVD/

DT1: .EVEN
SERRPC, SBDADR, SGDDAT, SBDAT, 0

```

*****
; DBUF' IS THE WORKING AREA IN MEM FOR ALL NPR OPERATIONS
*****
DBUF: 0 ;1ST ADRS OF DATA BUFFER
.END

```


.SEOP	18	3158	966
.SEPO	18	3158	1277
.SEPT	18	3158	1321
.SMLT	18		
.SPOE	18		
.SPRD	18		
.SPRD	18		
.SPRD	18		
.SPRD	18		
.SPRD	18	3158	
.SPRD	18	3158	1398
.SPRD	18		
.SPRD	18		
.SPRD	18		
.SSCP	18	3158	1368
.SSIZ	18		
.SSUP	18		
.STRAP	18	3158	1538
.STYPB	18		
.STYPD	18	3158	1210
.STYPE	18	3158	1063
.STYPO	18	3158	1133
.SNOCA	18		
.1170	18		

ABD	654	660	749	750	810	920	1090	1161	1171	1242	1343	1437	1446		
ABD	718	809	919	939	1057	1340	1341	1342	1460	1461	1462	1550			
ABD	1247		1059	1248											
ABD	720														
ABD	685	709	712	724	743	746	752	756	789	796	802	814	828	844	847
ABD	861	863	890	906	912	924	935	955	990	1093	1128	1188	1294	1297	1345
ABD	1350	1363	1417	1444	1456										
BCT	881	1195	1256	1456	1497										
BTC	705	781	793	812	827	859	865	896	903	922	954	978	1185	1413	1430
BTC	1457	1484	1490	1498											
BIS	1190	1191	1250	1251	1464										
BISB	1332														
BIT	708	723	742	751	755	788	843	860	889	1296	1303				
BITB	1116														
BLOS	1510														
BLT	1107	1196	1239	1255	1454	1495									
BMT	1246														
BNE	626	642	656	662	666	670	672	717	779	791	819	824	830	894	933
BNE	938	949	957	962	964	1005	1038	1052	1056	1087	1095	1103	1117	1124	1186
BNE	1244	1304	1333	1355	1409	1415	1435	1442	1449	1486	1492	1514	1520		
BPL	1081	1121	1184	1230	1260	1309	1411	1427	1482	1488					
BR	618	644	674	677	714	748	785	799	805	821	849	867	900	909	915
BR	927	942	1060	1083	1100	1110	1119	1126	1162	1177	1198	1241	1258	1338	1365
BR	1381	1387	1390	1438	1465	1467	1493	1516							
CLR	617	624	702	722	754	800	842	868	910	931	976	1024	1036	1049	1175
CLR	1233	1236	1331	1424	1425	1394	1521								
CLRB	823	823	1099	1125	1262	1394	1521								
CLRB	641	655	661	671	671	711	745	786	795	813	829	846	862	901	905
CLRB	922	932	956	1254	1388	1408	1414	1434	1441	1453	1455	1485	1491	1494	1496
CLRB	1509														
CLRB	1092	1094	1102	1123	1127	1416	1448	1513	1519						
COM	715	716	936	937	1054	1055									
DEC	778	790	818	823	893	948	963	979	1004	1339					
DEC	1106	1109	1183	1194											
EXT	340														
HALT	453	1082	1310												
INC	665	719	934	940	977	1037	1051	1058	1189	1197	1240	1299	1463		
INCB	1129	1293	1392												
IOT	341														
JMP	457	458	686	831	951	958	997	1006							
JSR	769	771	832	882	884	959	992	1101	1108	1115	1305	1318	1452		
MOV	619	623	627	629	630	631	632	633	634	637	638	639	640	645	647
MOV	648	649	651	652	653	657	658	659	681	683	687	703	704	706	707
MOV	710	721	738	739	740	741	744	753	773	774	775	776	780	782	783
MOV	787	792	794	797	801	803	806	807	811	815	817	822	826	845	848
MOV	864	886	887	888	891	895	897	898	902	904	907	911	913	916	917
MOV	921	925	928	929	930	943	944	945	947	953	955	982	986	989	1003
MOV	1015	1016	1023	1034	1035	1046	1047	1048	1050	1084	1085	1089	1104	1158	1166
MOV	1167	1168	1174	1181	1199	1200	1201	1202	1203	1223	1224	1225	1226	1227	1228
MOV	1229	1234	1237	1257	1263	1264	1267	1268	1269	1269	1270	1295	1300	1330	1335
MOV	1344	1349	1354	1366	1360	1383	1384	1386	1389	1393	1395	1396	1421	1445	1450
MOV	1479	1480	1507	1508	1523	1524	1525	1526	1546	1547	1551				
MOV	675	820	866	950	1086	1114	1122	1159	1160	1163	1164	1165	1169	1172	1173

	1192	1232	1235	1249	1252	1261	1302	1412	1429	1483	1489	1512	1517	1549
NEG	808	918	1170	1231										
NOP	850	975	993	994	995									
RESET	682	991												
ROL	1176	1178	1179	1180	1182									
RTI	646	1091	1204	1271	1313	1397	1451	1499	1527					
RTS	1017	1026	1039	1053	1131	1320	1358	1552						
SUB	1238	1301												
TRAP	1554	1563	1564	1565	1566	1568	1570	1571	1572					
TST	669	684	961	1088	1096	1118	1187	1243	1253	1308	1362	1385	1443	1458
TSTB	1080	1120	1245	1259	1410	1426	1481	1487						1548
.ASCII	500	501												
.ASCII2	499	502	679	1000	1366	1531	1532	1533	1535	1575	1581	1587	1592	1546
.BLKB	1605	1610	1614	1618	1623	1627	1632							1600
.BLKB	1530													
.BLKB	1276													
.BYTE	471	472	477	478	486	487	495	496	497	498	999	1205	1206	1208
.DSABL	1528	1529												
.ENABL	1468													
.END	1645	315	1401											
.ENDC	1508	335	336	340	432	446	458	465	469	471	499	500	504	616
	1087	737	733	761	651	667	671	677	679	690	691	700	701	728
	1000	879	880	869	970	767	836	837	840	841	854	858	857	872
	1283	1003	1010	1014	1020	971	973	976	981	984	985	997	991	999
	1374	1380	1300	1305	1306	1023	1029	1034	1042	1046	1066	1088	1136	1213
	1508	1510	1513	1515	1531	1537	1541	1401	1402	1404	1432	1468	1472	1368
	1508	1509	1570	1571	1572	1573	1642	1547	1550	1562	1563	1564	1565	1566
.EQUIV	340	341	349	364	365	394	395	396	397	398	399	400	401	402
	422	423	424	425	426	427	428	429	430	431				403
.EVEN	679	1367	1537	1638	404	432	456	464	468	470	499			616
.IF	316	335	336	338	404	432	456	464	468	470	499			622
	627	629	631	633	635	651	666	667	668	671	678			700
	727	728	731	733	735	760	765	766	835	836	837			856
	957	871	872	878	879	968	969	970	971	972	973			985
	1199	991	997	999	1000	1009	1013	1019	1022	1028	1033			1086
	1338	1212	1279	1282	1292	1296	1303	1305	1306	1308	1312			1323
	1503	1354	1370	1373	1374	1379	1380	1390	1513	1514	1517			1402
.IFF	1503	1504	1532	1471	1472	1500	1508	1509	1570	1571	1572			1546
	728	736	738	765	766	766	766	767	627	666	666			701
	888	872	873	877	880	880	881	881	836	837	837			857
	1029	1034	1042	1046	1056	1136	1213	1280	1282	1296	1312			1023
	1514	1539	1541	1547	1644	1644	1398	1401	1404	1472	1474			1318
.IFT	1514	1539	1541	1547	1644	1644								1510
.IFTF	679	305	306	307	308	309								
.IIF	315	316	317	318	319	320								
	977	978	979	980	981	982								
	1361	1374	1378	1394	1398	1401	1422	1422	1531	1537	1552	1563	1564	1565

.IRP	616	975	1223	1263	1292	1379									
.LIST	1	315	446	453	499	616	635	667	668	679	976	991	1313	1374	1500
	1554	1562	1563	1564	1565	1566	1567	1568	1569	1570	1571	1572	1573		
.MACRO	1	336	462	584	1554										
.MCALL	315	446	635	668											
.NLIST	1	315	446	453	499	616	635	667	668	679	976	991	1313	1374	1500
	1554	1562	1563	1564	1565	1566	1567	1568	1569	1570	1571	1572	1573		
.PAGE	462	504	542	584	616	688	1007	1061							
.REM	1														
.REPT	453														
.SBTTL	329	336	447	456	462	504	616	621	663	668	688	726	758	834	852
	870	966	1007	1061	1063	1133	1210	1277	1321	1368	1398	1538	1554	1574	
.TITLE	315														
.WORD	453	454	455	460	470	473	474	475	476	479	480	481	482	483	484
	485	488	489	490	980	983	998	1130	1209	1347	1352				

ERRORS DETECTED: 0

* , DVDRBA.SEG/SOL/CRF/NL: TOC/DS:ERFZ=DVDRBA.SML, DVDRBA.P11
RUN-TIME: 40 45 3 SECONDS
CORE USED: 32K

