

Micro Fiche Scan

Name of device(s) tested:

KDJ11-DA

Test description:

KDJ11-DA CLUSTER DIAGNOSTIC

MAINDEC Number or Package Identifier (after SEP 1977):

COKDDB0

Fiche Document Part Number:

AH-FG68B-MC

Fiche preparation date unknown, using copyright year:

1986

Image resolution:

1-bit black&white, compressed for minimal file size

COPYRIGHT (C) 1986 by d|i|g|i|t|a|l

B 1

A
?

SEQ 0001

COKDDBO KDJ11-DA CLUSTER DIAG. MACRO V05.03 Wednesday 09 Apr 86 09:43

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

.REM &

IDENTIFICATION

PRODUCT CODE: AC FG67B-MC
PRODUCT NAME: COKDDBO KDJ11 DA CLUSTER DIAG.
PRODUCT DATE: APRIL, 1986
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 DOCUMENT.

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C) 1986 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

Table of contents

11-	1113	OPERATIONAL SWITCH SETTINGS
11	1122	MEMORY MANAGEMENT DEFINITIONS
11	1123	BASIC DEFINITIONS
11-	1125	TRAP CATCHER
11	1133	ACT11 HOOKS
11-	1134	APT PARAMETER BLOCK
12-	1135	COMMON TAGS
12-	1135	APT MAILBOX-ETABLE
13-	1135	ERROR POINTER TABLE
13-	1138	ERROR DEFINITIONS
13-	1468	GLOBAL VARIABLES AND REGISTER NAMES
13-	1541	GLOBAL DATA SECTION
13-	1731	INITIALIZE THE COMMON TAGS
13-	1743	GET VALUE FOR SOFTWARE SWITCH REGISTER
14-	1813	TEST - NATIVE REGISTER
15-	1892	TEST - 16 BIT ROM CHECKSUM TEST
16-	1993	TEST - KDJ11-DA DATA PATHS
17-	2071	TEST - MEMORY ACCESSABILITY
18-	2130	TEST - MEMORY ERROR REGISTER
19-	2313	TEST - DATA SHORTS AND STUCK AT BITS
20-	2463	TEST - QUICK VERIFY DATA AND ADDRESSING TEST
21-	2549	TEST - CHECK PARITY DETECT LOGIC AND RAMS
22-	2755	TEST - LTC BIT 7
23	2826	TEST - LKS INTERRUPT PRIORITY
24-	2935	TEST - RESETTING LKS
25-	2994	TEST - MAINTENANCE REGISTER TEST
26-	3028	TEST - SERIAL LINE UNIT REGISTERS
27-	3086	TEST - XCSR BIT 7
28-	3142	TEST - RCSR BIT 7 AND XCSR BIT 2
29-	3248	TEST - RESET AND XCSR<2!0>
30-	3287	TEST - RESET AND INTERRUPT ENABLE BITS
31-	3507	TEST - INTERRUPT PRIORITY FOR SLU
32-	3672	TEST - BREAK CONDITION
33-	3786	TEST - OVERRUN CONDITION
34-	3900	TEST - LED'S ON
35-	3958	TEST - DIFFRENT LEVELS OF INTERRUPTS
36-	4043	TEST - ARBIRATION BETWEEN PIRQ'S AND INTERRUPTS
38-	4110	GLOBAL ERROR MESSAGES
39-	4251	MODIFIED ERROR MESSAGE TYPEOUT ROUTINE
39-	4317	GLOBAL SUBROUTINES SECTION
42-	4464	Q22BE SIZE ROUTINE
42-	4526	Q22BE INTERRUPT INITIALISE ROUTINE
42-	4537	DMATRN DATO CYCLE THRU Q22BE
42	4556	DMARD DATI THRU Q22BE
50	4951	END OF PASS ROUTINE
50-	4952	SCOPE HANDLER ROUTINE
50-	4953	ERROR HANDLER ROUTINE
50-	4953	ABORT ROUTINE FOR LCP/ORION UFD MODE
50	4954	APT COMMUNICATIONS ROUTINE
50-	4955	TYPE ROUTINE
50	4956	BINARY TO OCTAL (ASCII) AND TYPE
50-	4957	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
50-	4958	TTY INPUT ROUTINE
50-	4959	READ AN OCTAL NUMBER FROM THE TTY
50-	4960	TRAP DECODER
50	4960	TRAP TABLE
50	4961	POWER DOWN AND UP ROUTINES

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
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

TABLE OF CONTENTS

- 1.0 PROGRAM ABSRACT
- 2.0 PROGRAMMING CONVENTIONS
 - 2.1 Implementation Language
 - 2.2 Program Generation
 - 2.3 Hardware Requirement
 - 2.4 Loading and Starting Procedures
 - 2.5 Special Environments
 - 2.6 Program Options
 - 2.7 Execution Times
 - 2.8 Error Information
 - 2.9 Examples
- 3.0 PROGRAM DESCRIPTION
 - 3.1 J11 Code
 - 3.2 Native Register
 - 3.3 Maintenance Register
 - 3.4 KDJ11-DA on board memory
 - 3.5 On-Board Rom Code
 - 3.6 Line Time Clock Code
 - 3.7 Serial Line Unit Code
 - 3.8 On Board LED
 - 3.9 Q22BE Code
 - 3.10 List of Subtests Performed
 - 3.11 Program Updates and Modifications
- 4.0 BIBLIOGRAPHY
- 5.0 GLOSSARY
- APPENDICES

94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124

1.0 PROGRAM ABSTRACT

This program tests out KDJ11 DA CPU board, including the J11 chip set, on board MEMORY, on-board ROM's, serial line unit, line time clocks, and the Bus Arbitration.

The KDJ11-DA is a PDP 11 CPU that incorporates the J11 chip set as the heart of the processor. It is a quad height Q22 bus module. It has 512 KB of on board memory. The memory has parity detection and is located on 18 256k x 1 RAM chips. There is a memory CSR to help determine parity errors.

The KDJ11-DA also has 2 on board ROM's. They contain the self-test and the boot codes.

The Serial Line Unit is implemented thru 2 D1art chips which provide the standard console interface to the CPU. It has internal loop back mode and has a user selectable baud rate of 300 to 38400 baud.

The line time clock functions are implemented using the BEVENT line.

As a program option the Q22BE module is used to test verify the interrupt arbitration of the KDJ11-DA, DMA protocol, and PMG counter.

Further details are explicitly mentioned test by test in the DESIGN DESCRIPTION section # 3.0

126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
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

2.0 PROGRAMMING CONVENTIONS

2.1 Implementation Language

The PDP/11 Assembly Language has been used to write this CPU diagnostic

2.2 Program Generation

This Diagnostic is developed and assembled on a VAX using the PDP/11 mode "MCR MAC" and the "ORION.MLB" libraries the following:

```
$ MCR MAC
MAC> NAME,NAME/ SP=ORION.MLB/ML,NAME.MAC/DS:GBL
```

Finally NAME.OBJ would be transferred to an XXDP+ media from the VAX by using the "XDT" or "SHARON" utility.

2.3 Hardware Requirements

To run successfully the diagnostic needs:

- A. KDJ11-DA CPU module
- B. console terminal

In DVT, and stage one manufacturing (module assembly) the Q22 Bus exerciser is needed to check Q22 Bus logic.

2.4 Loading and Starting Procedures

To start-up this program:

1. Boot XXDP+
2. Type "R NAME", where name is the name of the BIN or BIC file for this program.

The starting address of the program is 200.
Note: if trying to restart the program in an arbitrary place after HALT on Break the following registers should be set up:
1777572=0 to disable memory management

2.5 Special Environments

The program is APT compatible. It can also be run under the UFD monitor. In those cases none of the standard error printouts occur. Refer to corresponding documents on running procedures in APT and under UFD monitor.

2.6 Program Options

The Q22 Bus Exerciser is utilized if it is present in the

183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239

system and the diagnostic is not running in UFD mode.
Standard capabilities of looping on test and on error are provided.

SWITCH REGISTER SELECTION:

BIT NUMBER	USE
15	HALT ON ERROR
14	LOOP ON PRESENT TEST
13	INHIBIT ERROR TIMEOUT
12	ENABLE TEST TRACING
11	INHIBIT ITERATIONS
10	BELL ON ERROR
9	LOOP ON ERROR
8	LOOP ON TEST IN SWR<5-0
7	INHIBIT THE CHECK PARITY TEST
6	Not used
5 0	Subtest number to loop on (BIT 8)

2.7 Execution Times

Without check parity test, the diagnostic runs in under 20 seconds. With it, it takes about 2 minutes.

2.8 Error Information

In the case of errors, a failing PC and test numbers are given. Where it is possible, expected and received data are given. For an example, see section 2.9.

2.9 Examples

After booting XXDP+ and starting the program, the following will appear on the terminal:

* KDJ11-DA CPU DIAGNOSTIC - COKDDAO *

SWR = XXXXXX NEW =

where XXXXXX correspond to present software switch register setting.

After "NEW" an operator can do one of the following:

- 1) type in a new software switch register setting followed by carriage return or
- 2) just type in carriage return in which case the software register will remain unchanged.

Example of error printout:

ERROR DOING Q22BE INTERRUPTS

240
241
242
243
244

TEST	ERROR	ADDRESS	ADDRESS
*	PC	<21-16>	<15-0>
27	105620	66600	000000

Note: this may not correspond to the actual Program Counter.

246
 247
 248
 249
 250
 251
 252
 253
 254
 255
 256
 257
 258
 259
 260
 261
 262
 263
 264
 265
 266
 267
 268
 269
 270
 271
 272
 273
 274
 275
 276
 277
 278
 279
 280
 281
 282
 283
 284
 285
 286
 287
 288
 289
 290
 291
 292
 293
 294
 295

3.0 PROGRAM DESCRIPTION

3.1 J11 CODE

This portion of the code tests out the J11 chip set. It is broken into 3 pieces: CPU tests, which verify different instructions in different modes and different trap conditions; MMU tests, which verify different functions of MMU; and FFP tests, which do different floating point instructions.

This portion of the code have been written in close relationship with the J11 microcode. Therefore, even though not all possible instructions in all possible addressing mode have been tested, an attempt has been made to exercise all of the microcode.

Most of the CPU diagnostic tests have been taken from the KDJ11 B Cluster diagnostic and has not been rewritten due to similarities.

3.2 NATIVE REGISTER

TEST - NATIVE REGISTER

NATIVE REGISTER TEST

This test checks out the native register's existence and it's various bits.

BGNTST

- } Set up timeout vector PC to TIMOUT
- } Set up timeout vector PSW to 7
- } Read the Native register
- } Check out the bootstrap switch as read only
- } Check out the module functional revision as read only
- } Check out that the self-test enable bit works
- } Check out the indicators (i.e LEDES)

ENDTST

TIMOUT: Clean stack
 Error NATIVE register timed out

297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318

3.3 MAINTENANCE REGISTER

TEST MAINTENANCE REGISTER TEST

MAINTENANCE REGISTER TEST

THIS TEST WILL ADDRESS MAINTENANCE REGISTER AND CHECK BITS
7 4 TO BE 0010, 2 1 TO BE 10, AND READ BITS 10 08, 03, 00
FOR FUTURE USE. THOSE BITS REPRESENT THE FOLLOWING SIGNALS:
MULTIPROCESSOR SLAVE, UNIBUS SYSTEM, FPA AVAILABLE, HALT/TRAP
OPTION, AND AC POWER OKAY.

ROUTINE TEST

IF MAINT. REG. BITS <7 4> NE 0010 OR <2 1> NE 10 THEN
ERROR

ENDIF

READ MAINT.REG. BITS <10 08,03,00>

ENDROUTINE

320
 321
 322
 323
 324
 325
 326
 327
 328
 329
 330
 331
 332
 333
 334
 335
 336
 337
 338
 339
 340
 341
 342
 343
 344
 345
 346
 347
 348
 349
 350
 351
 352
 353
 354
 355
 356
 357
 358
 359
 360
 361
 362
 363
 364
 365
 366
 367
 368
 369
 370
 371
 372
 373
 374
 375
 376

3.4 KDJ11 DA ON BOARD MEMORY

The KDJ11 DA has on board memory of 512 Kbytes block mode, with parity, fixed start at 0

The KDJ11 DA board uses 256K x 1 chips which allows us to use two patterns per RAM only.

Our memory diagnostic starts with testing the data path using this pattern: 0, 177777, 177400, 170360, 007417, 052525, 125252. The 2nd test is checking the accessibility of each address and an error will flag if memory traps. The third test is to check the memory registers specific to the KDJ11-DA. Fourth, is the short and stuck on bits test which will test every single bit 0/1 for all addresses. The KNAIZUK HARTMANN algorithm which divides the memory into sections of three is used in test five as a quick verify of all stuck at faults in the data and addressing. Test six checks the parity detect logic of all memory locations on the KDJ11 DA.
 You can refer to this section for more details.

TEST KDJ11 DA DATA PATHS

DATA PATH TEST

This test checks out the data and address paths on the KDJ11-D Board. The patterns used will be:

0
 177777
 177400
 170360
 007417
 052525
 125252

BGNTST

```

} Set Timeout trap to TIMEOUT
} Set timeout priority to 7
} Read location 0
} FOR pattern := first to last
}   Write pattern
}   Read pattern
}   IF Pattern read <> Pattern Written THEN
}     ERROR
} ENDFOR
ENDTST
  
```

TEST - MEMORY ACCESSABILITY

ACCESSIBILITY TEST

This test will check the accessibility of each address of memory on the KDJ11-D Board. IF a memory address traps out then an error

377
 378
 379
 380
 381
 382
 383
 384
 385
 386
 387
 388
 389
 390
 391
 392
 393
 394
 395
 396
 397
 398
 399
 400
 401
 402
 403
 404
 405
 406
 407
 408
 409
 410
 411
 412
 413
 414
 415
 416
 417
 418
 419
 420
 421
 422
 423
 424
 425
 426
 427
 428
 429
 430
 431
 432
 433

will be flagged. A side effect of this test should be that all memory is cleared

```

BGNTST
} Set timeout trap to TIMOUT
} Set timeout priority to 7
} For MSB Address := 200000 to 177777 D0
: Contents of address := 0
} Go to the next test
}
}TIMOUT:
} Error Address should not have timed out
ENDTST
  
```

TEST MEMORY ERROR REGISTER

MEMORY ERROR REGISTER

This test will check the MEMORY ERROR REGISTER on the KDJ11-D Board.

```

BGNTST
} Setup timeout VECTOR PC to TIMOUT
} Setup timeout VECTOR Priority to 7
} Setup Parity abort VECTOR to PARINT
} Setup Parity abort VECTOR Priority to 7
} Do a bus reset
} Read the Memory Error Register (17772100)
} Make sure that the register bits are in the right state
} Check all R/W bits
ENDTST
  
```

TEST - DATA SHORTS AND STUCK AT BITS

DATA SHORTS AND STUCK AT BITS TEST

This test will check the DATA Rams for Data shorts and stuck at bits. Testing occurs as below:

1. A memory location is checked to be set to 0
2. IF NOT 0 error
3. The location is complemented
4. IF contents NOT -1 error
5. This is repeated for all addresses
6. Steps 1-5 are done from location 177777 to 0

Note: The KDJ11-DA board uses 256k X 1 chips, This allows us to use only 2 patterns per RAM. IF one bits is shorted to another we will detect this when we read for the initial state. If it has changed from the expected state then chances are that the bits are shorted together.

A side effect of this test should be that memory is cleared.

```

434
435      BGNTST
436      } FOR MSB Address :- 0 to 1777776 DO BY 2
437      }   Clear Address
438      }   If Contents <> 0 THEN
439      }     ERROR in memory
440      }   Complement the Contents of Address
441      }   IF contents <> 1 THEN
442      }     ERROR in memory
443      } ENDFOR
444      } FOR MSB Address := 1777776 DOWNT0 0 DO BY 2
445      }   IF contents <> 1 THEN
446      }     ERROR in memory
447      }   Complement the Contents of Address
448      }   IF contents <> 0 THEN
449      }     ERROR in memory
450      } ENDFOR
451      ENDTST

```

452
453
454
455 TEST QUICK VERIFY DATA AND ADDRESSING TEST

456 UNIQUE ADDRESSING TEST

457
458 This test will check the data and addressing of the memories.
459 It uses the KNAIZUK HARTMANN QUICK VERIFY TRAM TEST ALGORITHM.
460 This algorithm will test memory for all stuck at faults in the
461 data and addressing
462

463 Memory is split up into sections of 3.
464 I.E. 0,1,2 - 3,4,5 - 6,7,10 ...

465 Testing works as follows.

- 466 1. Write a 0 into the 2nd and 3rd address of all groups
- 467 write a -1 into the 1st address of all groups
- 468 2. make sure that the 2nd address of all groups contain 0
- 469 3. Write a -1 into the 2nd address of all groups
- 470 4. Make sure that the 3rd address of all groups contain 0
- 471 5. Make sure that the 1st and 2nd address of all groups
- 472 contain -1
- 473 6. Write a 0 into the 1st address of all groups
- 474 7. Make sure that the 1st address of all groups contain 0
- 475 8. Write a 1 into the 3rd address of all groups
- 476 9. Make sure that the 3rd address of all groups contain -1

477
478
479
480
481
482
483
484 TEST - CHECK PARITY DETECT LOGIC AND RAMS

485 CHECK PARITY DETECT LOGIC

486 This test will check out the parity detection logic on the KDJ11 DA
487 board. We will write wrong parity to all locations and then we will
488 Expect the a parity trap on a read.
489
490

```

491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547

```

```

BGNTST
} Set up Parity Vector (114) To PARINT
} Enable Write Bad parity (Set bit 2 in MER (17772100))
} Clear all of memory
} Read first location written to
} IF Parity Abort Occurs THEN
}   Error There should be no Parity Aborts when disabled
} Enable Parity Error (Set Bit 0 in MER (17772100))
} FOR First Address to Last address DO
} BEGIN
}   READ Address
}   NOP
}   IF No Interrupt THEN
}     Error Didn't Detect Bad parity
}   ELSE
}     Clear the interrupt flag
}     Read the MER and make sure the obtained address is the correct one
} ENDFOR
} PARINT:
}   Flag that an interrupt occurred
}   RTI
} ENDTST

```

3.5 ON-BOARD ROM CODE

The KDJ11-DA Native Firmware resides in a 2-16Kx8 EPROM's which are physically located on the KDJ11-DA module. The Firmware has a monitor to accept user commands, auto sequence boot, console boot, power up self tests, extended self tests invoked via user intervention, and support a text of basic error messages in all supported eleven foreign languages including extended error messages in English.

In this test we perform a sum test for the on Board ROM's which resides at address 17400000 - 17677777.

TEST - 16 BIT ROM CHECKSUM TEST

ROM'S CHECKSUMS

16 BIT ROM TEST

```

BGNTST
} INIT MMU REGISTERS
} POINT PAGE 2 TO SELF TEST
} ENABLE SELF TEST BIT

```

```

548      } READ FIRST 2 WORDS OF ROM
549      } GET EXPECTED CHECKSUM
550      }
551      } DO CALCULATE CHECKSUM
552      }   } ADD WORD TO REGISTER
553      }   } DO UNTIL ALL WORDS ADDED
554      } END CALCULATION
555      }
556      } NEGATE THE SUM OF WORDS
557      } IF NEG. SUM < > EXPECTED CHECKSUM
558      } ERROR
559      }
560      }
561      }
562      }
563      }
564      }
565      }
566      }
567      }
568      }
569      }
570      }
571      }
572      }
573      }
574      }
575      }
576      }
577      }
578      }
579      }
580      }
581      }
582      }
583      }
584      }
585      }
586      }
587      }
588      }
589      }
590      }
591      }
592      }
593      }
594      }
595      }
596      }
597      }
598      }
599      }
600      }
601      }
602      }
603      }
604      }

```

3.6 LINE TIME CLOCKS CODE

The line time clock control and status register is accessed at address 17777546. The EVENT interrupt thru vector 100 on interrupt request level 6 is received from the Q-bus signal BEVENT.

Note: in UFD mode only functions corresponding to Boot Rom selection are checked.

In this diagnostic we start with test#1: existence of the clock, test#2: interrupt priority max 5, test#3: resetting LKS.

TEST - LTC BIT 7

LTC BIT 7 TEST

This test check for the existence of the LTC register and it makes sure that the clock is ticking.

BGNTST

```

Set up timeout vector PC to TIMOUT
Set up timeout vector PSW to 7
Read the LTC CSR
IF not timed OUT THEN
: FOR a set amount of time
:   Check bit7
:   IF BIT7 is set THEN
:     Increment BIT7 set flag
:   ELSE
:     INCREMENT BIT7 Clear Flag
:   ENDFOR
: IF either flag has not been set at all THEN
:   Error Clock is not ticking

```

ENDTST

```

TIMOUT: Clean stack
        Error LTC register timed out

```

```

605
606
607 TEST LKS INTERRUPT PRIORITY
608
609 CHECK THAT LKS INTERRUPTS HAPPEN AT PRIORITY 5 CLEARING LKS<07>
610 AND DON'T HAPPEN AT PRIORITY 6.
611 ROUTINE TEST
612 IF UFD AND LKS IS DISABLED THEN
613 . EXIT TEST
614 .
615 . SET PRIORITY TO 5
616 . CLEAR INTERRUPT_FLAG
617 . LET LKS<06>=#1 (ENABLE INTERRUPTS)
618 . SET COUNTER TO WAIT FOR 3 INTERRUPTS
619 . REPEAT
620 . . DECREMENT COUNTER
621 . . UNTIL INTERRUPT_FLAG EQ #3 OR COUNTER EQ #0
622 . . CLEAR LKS<06>
623 . . IF LKS<07> EQ #1 THEN
624 . . . ERROR (WAS NOT CLEARED ON INTERRUPT)
625 . . ENDIF
626 . . IF COUNTER LT TIME_REQUIRED_FOR_3_INTERRUPTS_FOR_800HZ
627 . . . ERROR (INTERRUPTS NEVER GO LOW)
628 . . ENDIF
629 . . IF INTERRUPT_FLAG LT #3 THEN
630 . . . ERROR (INTERRUPTS DON'T HAPPEN)
631 . . ENDIF
632 . . CLEAR INTERRUPT_FLAG
633 . . WAIT FOR LKS<7>=#1
634 . . LET LKS<7>=#0
635 . . IF LKS<7> NE #0 THEN
636 . . . ERROR (LKS<7> NOT CLEARED)
637 . . ENDIF
638 . . SET PRIORITY TO 6
639 . . SET COUNTER TO 1 SLOW CLOCK INTERRUPT
640 . . SET LKS<06>
641 . . REPEAT
642 . . . DECREMENT COUNTER
643 . . . UNTIL COUNTER EQ #0 OR INTERRUPT_FLAG NE #0
644 . . . IF INTERRUPT_FLAG NE #0 THEN
645 . . . . ERROR (INTERRUPT WAS AT WRONG PRIORITY)
646 . . . ENDIF
647 . . . RESTORE ORIGINAL PRIORITY
648 . ENDROUTINE
649
650 ROUTINE LINE_CLOCK_INTERRUPT
651 . INCREMENT INTERRUPT_FLAG
652 . ENDROUTINE
653
654
655 TEST RESETTING LKS
656
657 RESETTING LKS(*)
658 THIS TEST WILL PROVE THAT RESET INSTRUCTION CLEARS LKS<06>.
659 ROUTINE TEST
660 IF UFD AND LKS IS DISABLED THEN
661

```



```

662          .EXIT TEST
663          .ENDIF
664          .POINT LKS VECTOR 100 TO ERROR_LKS_ILLEGAL_INTERRUPT
665          .SYNCHRONIZE LKS BY WAITING FOR 3 PULSES
666          .LET LKS<06>=#1
667          .EXECUTE "RESET"
668          .IF LKS<6> NE #0 THEN
669              .ERROR
670          .ENDIF
671          .IF ILLEGAL_LINE_CLOCK_INTERRUPT NE 0 THEN
672              .ERROR
673          .ENDIF
674          .ENDROUTINE
675
676          ROUTINE ERROR_LKS_ILLEGAL_INTERRUPT
677              FLAG_ILLEGAL_LINE_CLOCK_INTERRUPT
678          .RETURN
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718

```

3.7 SERIAL LINE UNIT CODE

The KDJ11-da board has two Serial Line Unit that have to be tested. When the processor halts it enters uODT and communicates over the SLU0 at addresses 17777560 thru 17777566. They considered the console addresses by the CPU. SLU1 is the General purpose Serial I/O and is addressed at 17776500 thru 17776506 and SLU1 does not have the halt on breake option.

In this diagnostic we verify the functionality of the SLU chip utilizing the maintenance mode of the chip. All serial line unit tests: interrupt level, loop back capability, and transmitter receiver bits registers, are performed on SLU0 and SLU1 except the BREAK CONDITION and the OVERRUN CONDITION tested on SLU1 only, because SLU0 is used as the console .

TEST - SERIAL LINE UNIT REGISTERS

```

SERIAL LINE UNIT TEST(*)
BCR<2-0> WILL BE READ TO FIND OUT BAUD RATE. SLU WILL BE PROG
RAMMED TO CHECK THE INTERRUPT LEVELS BY SETTING BIT<06> IN
RCSR AND XMIT. LOOP BACK CAPABILITIES WILL BE TESTED BY SETTING
TO 1 XCSR<02>. THE LINE CLOCK INTERRUPT SUBROUTINE WILL BE
USED TO RETURN TO THE EXECUTION OF THE DIAGNOSTICS, IF THE
PROGRAM HANGS IN THE LOOP BACK MODE.
ROUTINE TEST
IF UFO AND CONSOLE NOT PRESENT
    GO TO TEST_22
ENDIF
IF BCR<07> EQ #0 THEN

```

```

719           .               READ BCR<2 0> TO GET BAUD RATE
720           .               ENDF
721           .               LET 4=ADDRESS OF TIMEOUT ROUTINE
722           .               DO FOR RCSR,XCSR,RBUF,XBUF
723           .                   READ XCSR,XCSR,RBUF,XBUF
724           .                   IF TIMEOUT FLAG NE #0 THEN
725           .                       .
726           .                       ENDF
727           .               ENDDO
728           ENDRoutine
729
730 ROUTINE TIMEOUT
731           LET TIMEOUT FLAG=#1
732 ENDRoutine
733
734
735 TEST - XCSR BIT 7
736
737 CHECK THAT XCSR<07> CAN BE 0 AND 1.
738
739 XCSR   <07>          TRANSMITTER READY
740
741 ROUTINE TEST
742           WAIT FOR XCSR<07>=#1 NO MORE THAN 200MSEC
743           IF XCSR<07> NE #1 THEN
744               .
745               ERROR
746           .
747           ENDF
748           LET XBUF=#NULL
749           WAIT FOR XCSR<07>=#1
750           LET XBUF=#NULL
751           IF XCSR<07> NE 0 THEN
752               .
753               ERROR (READY DIDN'T GO LOW)
754           .
755           ENDF
756 ENDRoutine
757
758 TEST - RCSR BIT 7 AND XCSR BIT 2
759
760 CHECK THAT RCSR<07> CAN BE 0 AND 1 AND THAT XCSR<02> WORKS PROPERLY.
761
762 RCSR   <07>          RECEIVER DONE
763 RCSR   <02>          MAINTENANCE
764
765 ROUTINE TEST
766           .(CHECK RCSR<07> AND XCSR<07>)
767           WAIT FOR XCSR<07>=#1
768           LET XCSR<02>=#1 (LOOP BACK MODE)
769           LET XBUF=#125
770           WAIT FOR RCSR<07>=#1 NO MORE THAN 200MSEC
771           IF RCSR<07> NE #1 THEN
772               .
773               ERROR (RCSR<07> DOES NOT BECOME 1 OR XCSR<02> DOES NOT
774               .               WORK)
775           .
776           ENDF
777           IF RBUF NE #125 THEN
778               .
779               ERROR

```

```

776      .      ENDIF
777      .      IF RCSR<07> NE #0 THEN
778      .          ERROR (RCSR<07>DOES NOT GO LOW)
779      .      ENDIF
780      .      LET XCSR<02>=#0
781      .  ENDROUTINE
782
783
784
785      TEST   RESET AND XCSR<2!0>
786
787      CHECK THAT RESET CLEARS XCSR<0!2>.
788      ROUTINE TEST
789      .(CHECK RCSR<07> AND XCSR<07> AND RESET)
790      .      LET XCSR<02,00>=#1 (LOOP BACK MODE)
791      .      EXECUTE "RESET"
792      .      IF XCSR<02!00> NE #0 THEN
793      .          ERROR
794      .      ENDIF
795      .      LET XCSR<02>=#0
796      .  ENDROUTINE
797
798
799
800      TEST   - RESET AND INTERRUPT ENABLE BITS
801
802      CHECK THAT INTERRUPTS DON'T HAPPEN AT PRIORITY 4 AND THAT RESET
803      CLEARS XCSR<06> AND RCSR<06>.
804
805      RCSR   <06>   RECEIVER INTERRUPT ENABLE
806      XCSR   <06>   TRANSMITTER INTERRUPT ENABLE
807
808      ROUTINE TEST
809      .      LET 60=#ADDRESS_OF_ILLEGAL_INTERRUPT XRCSR
810      .      LET 64=#ADDRESS_OF_ILLEGAL_INTERRUPT XRCSR
811      .      SET PRIORITY TO 4
812      .      LET XCSR<02>=#1 (LOOPBACK MODE)
813      .      LET XCSR<06>=#1 (ENABLE TRANSMIT INTERRUPTS)
814      .      LET RCSR<06>=#1 (ENABLE RECEIVE INTERRUPTS)
815      .      WAIT FOR XCSR<07>=#1 (READY TO TRANSMIT)
816      .      LET XBUF=#NULL (SEND A CHARACTER)
817      .      WAIT FOR ILLEGAL INTERRUPTS (ABOUT 200MSEC)
818      .      EXECUTE "RESET"
819      .      IF XCSR<06> NE #0 OR RCSR<06> NE #0 OR XRCSR NE #0 THEN
820      .          ERROR
821      .      ENDIF
822      .      RESTORE PRIORITY TO NORMAL
823      .  ENDROUTINE
824
825      ROUTINE ILLEGAL_INTERRUPT_XRCSR
826      .      INCREMENT XRCSR
827      .  ENDROUTINE
828
829
830
831      TEST   - INTERRUPT PRIORITY FOR SLU
832

```

```

833 CHECK THAT INTERRUPTS HAPP' . AT PRIORITY 3 AND THAT THEY CLEAR
834 RCSR<06> AND XCSR<06>.
835
836 ROUTINE TEST
837 . LET 60=#ADDRESS OF LEGAL RINTERRUPT
838 . LET 64=#ADDRESS OF LEGAL XINTERRUPT
839 . LET XCSR<02>=#1
840 . SET PRIORITY TO #3
841 . WAIT FOR XINTERRUPT=#3
842 . IF XCSR<07> EQ #1 THEN
843 .     ERROR
844 . ENDIF
845 . WAIT FOR RINTERRUPT=#3
846 . IF RCSR<07> EQ #0 THEN
847 .     ERROR
848 . ENDIF
849 . LET XCSR<02>=#0
850 . SET PRIORITY TO NORMAL
851 ENDRoutine
852
853 ROUTINE LEGAL_XINTERRUPT
854 . LET XBUF=#CHARACTER
855 . INCREMENT XINTERRUPT
856 ENDRoutine
857
858 ROUTINE LEGAL_RINTERRUPT
859 . READ RCSR
860 . INCREMENT RINTERRUPT
861 ENDRoutine
862
863
864
865 TEST BREAK CONDITION
866
867 SLU #1 IS TESTED SINCE SLU #0 IS BEING USED AS THE CONSOLE
868
869 CHECK THAT SENDING BREAK CAUSES FRAMING ERROR.
870
871 RCSR <15> ERROR
872 <13> FRAMING ERROR
873 <11> RECEIVED BREAK
874
875 XCSR <00> TRANSMIT BREAK
876
877 ROUTINE TEST
878 . LET XCSR<02>=#1
879 . LET XCSR<00>=#1
880 . WAIT FOR RCSR<07>=#1
881 . IF RBUF<15!13!11> NE #1 THEN
882 .     ERROR (ERROR, FRAMING ERROR, RECEIVE BREAK NE 1)
883 . ENDIF
884 . LET XCSR<00>=#0
885 . IF XCSR<00> NE #0 THEN
886 .     ERROR (XCSR<00> DOES NOT GO LOW)
887 . ENDIF
888 . WAIT FOR XCSR<07>=#1
889 . LET XBUF=#NULL (SEND NULL CHARACTER TO SEE ERROR CLEARED)

```

```

890      .      WAIT FOR RCSR<07>=#1
891      .      IF RBUF<15!13!11> NE #0 THEN
892      .          ERROR
893      .      ENDIF
894      .      LET XCSR<00>=#1
895      .      EXECUTE "RESET"
896      .      IF XCSR<00> NE #0 THEN
897      .          ERROR
898      .      ENDIF
899      .      LET XCSR<02>=#0
900      .      ENDROUTINE
901
902
903      TEST - OVERRUN CONDITION
904
905      CHECK OVERRUN CONDITION
906
907      RCSR <14> OVERRUN ERROR
908
909      SLU #1 IS TESTED SINCE SLU #0 IS BEING USED AS THE CONSOLE
910
911      ROUTINE TEST
912      .      LET XCSR<02>=#1 (LOOPBACK MODE)
913      .      WAIT FOR XCSR<07>=#1
914      .      LET XBUF=#252
915      .      WAIT FOR XCSR<07>=#1
916      .      LET XBUF=#125 (SEND THE 2ND W/O READING THE 1ST CHARACTER)
917      .      WAIT FOR RCSR<07>=#1
918      .      STALL FOR LOWEST BAUD RATE TO GET 2ND CHARACTER
919      .      IF LOW BYTE OF RBUF NE #125 THEN
920      .          ERROR (1ST CHARACTER WASN'T OVERRUN)
921      .      ENDIF
922      .      IF RBUF<15!14> NE #1 THEN
923      .          ERROR (NO OVERRUN BIT SET)
924      .      ENDIF
925      .      WAIT FOR XCSR<07>=#1
926      .      LET XBUF=#NULL
927      .      WAIT FOR RCSR<07>=#1
928      .      IF RBUF<15!14> NE #0 THEN
929      .          ERROR (WASN'T CLEARED ON THE NEXT CHARACTER RECEIVED)
930      .      ENDIF
931      .      LET XCSR<02>=#0
932      .      ENDROUTINE
933
934
935
936
937
938
939
940
941
942
943
944      3.8 LED'S TEST CODE
945      This test will determine that the LED's situated on the CPU board
946

```

work correctly. This is done by sending a delayed turn on and off to the LED's so we can see their rotational pattern.

947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968

TEST LED'S ON

LED'S ON
THIS TEST WILL INITIALIZE BDR TO CONTAIN A ROTATING PATTERN
DISPLAYED IN LED'S.

ROUTINE TEST
· WHILE A KEY NOT RECEIVED FROM KEYBOARD DO
· · STALL ALLOWING TIME TO SEE PATTERN
· · ROTATE LEFT TO LIGHT UP NEXT LED'S
· · ENDDO
ENDROUTINE

970
 971
 972
 973
 974
 975
 976
 977
 978
 979
 980
 981
 982
 983
 984
 985
 986
 987
 988
 989
 990
 991
 992
 993
 994
 995
 996
 997
 998
 999
 1000
 1001
 1002
 1003
 1004
 1005
 1006
 1007
 1008
 1009
 1010
 1011
 1012
 1013
 1014
 1015
 1016
 1017
 1018
 1019
 1020
 1021
 1022
 1023
 1024
 1025
 1026

3.9 Q22BE CODE

The Q22 Bus Exerciser is a hardware option module that can work in conjunction with the CPU to test different levels of interrupt, arbitration between PIRQ's, and PMG counter of the KDJ11-DA, Direct Memory Access protocol.

More details about every test could be found in this section.

TEST - DIFFERENT LEVELS OF INTERRUPTS

DIFFERENT LEVELS OF INTERRUPTS
 THIS TEST WILL PROGRAM Q22 BUS EXERCISER TO INTERRUPT AT DIFFERENT LEVELS. ARBITRATION BETWEEN DIFFERENT LEVELS OF INTERRUPTS AND PIRQ'S WILL BE TESTED.

CHECK DIFFERENT LEVELS OF INTERRUPTS.

```

ROUTINE TEST
.   SET VECTOR TO INTERRUPT DMA
.   FOR INTERRUPTS FROM 4 TO 7 DO
.       ENABLE INTERRUPTS
.       SET PRIORITY=INTERUPT
.       IF INTERRUPT FLAG SET THEN
.           ERROR
.       ENDIF
.       ENABLE INTERRUPTS
.       SET PRIORITY=INTERRUPT-1
.       IF INTERRUPT FLAG NOTSET THEN
.           ERROR
.       ENDIF
.       LET INTERRUPT_DMA=0
.   ENDDO
ENDROUTINE
  
```

```

ROUTINE INTERUPT_DMA
.   LET INTERRUPT_FLAG=1
RETURN
ENDROUTINE
  
```

TEST - ARBITRATION BETWEEN PIRQ'S AND INTERRUPTS

CHECK PRIORITY ORDER BETWEEN PIRQ'S AND INTERRUPTS.

```

ROUTINE TEST
.   IF UFD THEN
.       EXIT TEST
.   ENDIF
.   DO FOR I FROM #6 DOWN TO #3
.       SET PRIORITY TO I
.       ENABLE INTERRUPT(I+1) AND PIRQ(I+1)
.       IF INTERRUPT(I+1) WAS BEFORE PIRQ(I+1) THEN
.           ERROR
.       ENDIF
.   
```

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
1061
1062
1063
1064
1065
1066
1067
1068
1069
1070
1071
1072

ENDDO
ENDROUTINE

3.10 LIST OF SUBTESTS PERFORMED

The following list represents the sequential order of subtests performed in COKDDAO..... subtests which are subject to the APT qualifications of section 3.2 are indicated by a Cache APT label.

TEST NO.	TEST NAME/FUNCTION
test 1	Base instruction set tests
test 32	KDJ11 DA on Board Memory
test 33	KDJ11-DA Data Path
test 34	Memory accessibility
test 35	Memory Error Register
test 36	Data short and stuck at bits
test 37	Quick verify Data and Addressing
test 38	Check Parity detect Logic and RAM's
test 39	Native Register
test 40	On Board ROM code
test 41	LTC bit 7
test 42	LKS interrupt Priority
test 43	Resetting LKS
test 44	Maintenance Register
test 45	Serial Line Unit Register
test 46	XCSR bit 7
test 47	RCSR bit 7 and XCSR bit 2
test 48	Reset and XCSR<2!0>
test 49	Reset and interrupt enable bit
test 50	Interrupt priority for SLU
test 51	Break Condition
test 52	Overrun Condition
test 53	LED's On
test 54	Different Level of interrupt
test 55	Arbitration between PIRQ's and interrupts

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

3.11 PROGRAM UPDATES AND MODIFICATIONS

Version COKDDAO	10-01-85	Michael Charchaflian
Version COKDD80	02-01 86	Michael Charchaflian

4.0 BIBLIOGRAPHY

N/A

5.0 GLOSSARY

N/A

APPENDICES

N/A

&

1115 167400
1116 000300
1117

```

$SWR=167400
$SWRMK=300
.TITLE COKDDBO KDJ11 DA CLUSTER DIAG.
;*COPYRIGHT (C) APR 86
;*DIGITAL EQUIPMENT CORP.
;*MAYNARD, MASS. 01754
;*
;*PROGRAM BY DIAG. ENG.
;*
;*THIS PROGRAM WAS ASSEMBLED USING THE PDP 11 MAINDEC SYSMAC
;*PACKAGE (MAINDEC-11 DZQAC-C8), OCT, 1982.

```

1118 000001

```

$TN=1
.SBTTL OPERATIONAL SWITCH SETTINGS
;*
;* SWITCH USE
;* ---
;* 15 HALT ON ERROR
;* 14 LOOP ON TEST
;* 13 INHIBIT ERROR TYPEOUTS
;* 11 INHIBIT ITERATIONS
;* 10 BELL ON ERROR
;* 9 LOOP ON ERROR
;* 8 LOOP ON TEST IN SWR<5:0>
;* 7 DO EXTENSIVE DATA RAM TEST
;* 6 DO EXTENSIVE TAG RAM TEST

```

1120
1121
1122

```

.SBTTL MEMORY MANAGEMENT DEFINITIONS

```

000250

```

;*KT11 VECTOR ADDRESS
MMVEC= 250
;*KT11 STATUS REGISTER ADDRESSES

```

177572
177574
177576
172516

```

SR0 177572
SR1 177574
SR2= 177576
SR3= 172516
;*USER "I" PAGE DESCRIPTOR REGISTERS

```

177600
177602
177604
177606
177610
177612
177614
177616

```

UIPDR0= 177600
UIPDR1= 177602
UIPDR2= 177604
UIPDR3= 177606
UIPDR4= 177610
UIPDR5= 177612
UIPDR6= 177614
UIPDR7= 177616
;*USER "D" PAGE DESCRIPTOR REGISTORS

```

177620
177622
177624
177626
177630
177632
177634
177636

```

UDPDR0= 177620
UDPDR1= 177622
UDPDR2= 177624
UDPDR3= 177626
UDPDR4= 177630
UDPDR5= 177632
UDPDR6= 177634
UDPDR7= 177636
;*USER "I" PAGE ADDRESS REGISTERS

```

177640
177642
177644
177646

```

UIPAR0= 177640
UIPAR1= 177642
UIPAR2= 177644
UIPAR3= 177646

```

MEMORY MANAGEMENT DEFINITIONS

177650	UIPAR4= 177650
177652	UIPAR5= 177652
177654	UIPAR6= 177654
177656	UIPAR7= 177656
	;*USER "D" PAGE ADDRESS REGISTERS
177660	UDPAR0= 177660
177662	UDPAR1= 177662
177664	UDPAR2= 177664
177666	UDPAR3= 177666
177670	UDPAR4= 177670
177672	UDPAR5= 177672
177674	UDPAR6= 177674
177676	UDPAR7= 177676
	;*SUPERVISOR "I" PAGE DESCRIPTOR REGISTERS
172200	SIPDR0= 172200
172202	SIPDR1= 172202
172204	SIPDR2= 172204
172206	SIPDR3= 172206
172210	SIPDR4= 172210
172212	SIPDR5= 172212
172214	SIPDR6= 172214
172216	SIPDR7= 172216
	;*SUPERVISOR "D" PAGE DESCRIPTOR REGISTERS
172220	SDPDR0= 172220
172222	SDPDR1= 172222
172224	SDPDR2= 172224
172226	SDPDR3= 172226
172230	SDPDR4= 172230
172232	SDPDR5= 172232
172234	SDPDR6= 172234
172236	SDPDR7= 172236
	;*SUPERVISOR "I" PAGE ADDRESS REGISTERS
172240	SIPAR0= 172240
172242	SIPAR1= 172242
172244	SIPAR2= 172244
172246	SIPAR3= 172246
172250	SIPAR4= 172250
172252	SIPAR5= 172252
172254	SIPAR6= 172254
172256	SIPAR7= 172256
	;*SUPERVISOR "D" PAGE ADDRESS REGISTERS
172260	SDPAR0= 172260
172262	SDPAR1= 172262
172264	SDPAR2= 172264
172266	SDPAR3= 172266
172270	SDPAR4= 172270
172272	SDPAR5= 172272
172274	SDPAR6= 172274
172276	SDPAR7= 172276
	;*KERNEL "I" PAGE DESCRIPTOR REGISTERS
172300	KIPDR0= 172300
172302	KIPDR1= 172302
172304	KIPDR2= 172304
172306	KIPDR3= 172306
172310	KIPDR4= 172310
172312	KIPDR5= 172312
172314	KIPDR6= 172314

MEMORY MANAGEMENT DEFINITIONS

```

172316      KIPDR7= 172316
            ;*KERNEL "D" PAGE DESCRIPTOR REGISTERS
172320      KDPDR0= 172320
172322      KDPDR1= 172322
172324      KDPDR2= 172324
172326      KDPDR3= 172326
172330      KDPDR4= 172330
172332      KDPDR5= 172332
172334      KDPDR6= 172334
172336      KDPDR7= 172336
            ;*KERNEL "I" PAGE ADDRESS REGISTERS
172340      KIPAR0= 172340
172342      KIPAR1= 172342
172344      KIPAR2= 172344
172346      KIPAR3= 172346
172350      KIPAR4= 172350
172352      KIPAR5= 172352
172354      KIPAR6= 172354
172356      KIPAR7= 172356
            ;*KERNEL "D" PAGE ADDRESS REGISTERS
172360      KDPAR0= 172360
172362      KDPAR1= 172362
172364      KDPAR2= 172364
172366      KDPAR3= 172366
172370      KDPAR4= 172370
172372      KDPAR5= 172372
172374      KDPAR6= 172374
172376      KDPAR7= 172376
1123      .SBTTL BASIC DEFINITIONS
            ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100      STACK= 1100
104000      ERROR= EMT                ;;BASIC DEFINITION OF ERROR CALL
000004      SCOPE= IOT                ;;BASIC DEFINITION OF SCOPE CALL
            ;*MISCELLANEOUS DEFINITIONS
000011      HT= 11                    ;;CODE FOR HORIZONTAL TAB
000012      LF= 12                    ;;CODE FOR LINE FEED
000015      CR= 15                    ;;CODE FOR CARRIAGE RETURN
000200      CRLF= 200                 ;;CODE FOR CARRIAGE RETURN-LINE FEED
177776      PS= 177776                ;;PROCESSOR STATUS WORD
177776      PSW= PS
177774      STKLMT= 177774            ;;STACK LIMIT REGISTER
177772      PIRQ= 177772              ;;PROGRAM INTERRUPT REQUEST REGISTER
177570      DSWR= 177570              ;;HARDWARE SWITCH REGISTER
177570      DDISP= 177570             ;;HARDWARE DISPLAY REGISTER
            ;*GENERAL PURPOSE REGISTER DEFINITIONS
000000      R0= *0                    ;;GENERAL REGISTER
000001      R1= *1                    ;;GENERAL REGISTER
000002      R2= *2                    ;;GENERAL REGISTER
000003      R3= *3                    ;;GENERAL REGISTER
000004      R4= *4                    ;;GENERAL REGISTER
000005      R5= *5                    ;;GENERAL REGISTER
000006      R6= *6                    ;;GENERAL REGISTER
000007      R7= *7                    ;;GENERAL REGISTER
000006      SP= *6                    ;;STACK POINTER
000007      PC= *7                    ;;PROGRAM COUNTER
            ;*PRIORITY LEVEL DEFINITIONS
000000      PRO= 0                    ;;PRIORITY LEVEL 0

```

BASIC DEFINITIONS

```

000040 PR1= 40 ;;PRIORITY LEVEL 1
000100 PR2= 100 ;;PRIORITY LEVEL 2
000140 PR3= 140 ;;PRIORITY LEVEL 3
000200 PR4= 200 ;;PRIORITY LEVEL 4
000240 PR5= 240 ;;PRIORITY LEVEL 5
000300 PR6= 300 ;;PRIORITY LEVEL 6
000340 PR7= 340 ;;PRIORITY LEVEL 7

```

;"SWITCH REGISTER" SWITCH DEFINITIONS

```

100000 SW15= 100000
040000 SW14= 40000
020000 SW13= 20000
010000 SW12= 10000
004000 SW11= 4000
002000 SW10= 2000
001000 SW09= 1000
000400 SW08= 400
000200 SW07= 200
000100 SW06= 100
000040 SW05= 40
000020 SW04= 20
000010 SW03= 10
000004 SW02= 4
000002 SW01= 2
000001 SW00= 1
001000 SW9= SW09
000400 SW8= SW08
000200 SW7= SW07
000100 SW6= SW06
000040 SW5= SW05
000020 SW4= SW04
000010 SW3= SW03
000004 SW2= SW02
000002 SW1= SW01
000001 SW0= SW00

```

;"DATA BIT DEFINITIONS (BIT00 TO BIT15)

```

100000 BIT15= 100000
040000 BIT14= 40000
020000 BIT13= 20000
010000 BIT12= 10000
004000 BIT11= 4000
002000 BIT10= 2000
001000 BIT09= 1000
000400 BIT08= 400
000200 BIT07= 200
000100 BIT06= 100
000040 BIT05= 40
000020 BIT04= 20
000010 BIT03= 10
000004 BIT02= 4
000002 BIT01= 2
000001 BIT00= 1
001000 BIT9= BIT09
000400 BIT8= BIT08
000200 BIT7= BIT07
000100 BIT6= BIT06
000040 BIT5= BIT05
000020 BIT4= BIT04

```

BASIC DEFINITIONS

```

000010 BIT3= BIT03
000004 BIT2= BIT02
000002 BIT1= BIT01
000001 BIT0= BIT00
;*BASIC "CPU" TRAP VECTOR ADDRESSES
000004 ERRVEC= 4 ;;TIME OUT AND OTHER ERRORS
000010 RESVEC= 10 ;;RESERVED AND ILLEGAL INSTRUCTIONS
000014 TBITVEC=14 ;; "T" BIT
000014 TRTVEC= 14 ;;TRACE TRAP
000014 BPTVEC= 14 ;;BREAKPOINT TRAP (BPT)
000020 IOTVEC= 20 ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
000024 PWRVEC= 24 ;;POWER FAIL
000030 EMTVEC= 30 ;;EMULATOR TRAP (EMT) **ERROR**
000034 TRAPVEC=34 ;; "TRAP" TRAP
000060 TKVEC= 60 ;;TTY KEYBOARD VECTOR
000064 TPVEC= 64 ;;TTY PRINTER VECTOR
000240 PIRQVEC=240 ;;PROGRAM INTERRUPT REQUEST VECTOR
1124 UFDSET= 1 ;FLAG FOR UFD
1125 .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 .=174
000174 000000 DISPREG: .WORD 0 ;;SOFTWARE DISPLAY REGISTER
000176 000000 SWREG: .WORD 0 ;;SOFTWARE SWITCH REGISTER
1126 .=200
1127 000200 005037 001160 CLR $TMPO
1128 000204 000137 004060 JMP @START
1129 000220 .=220
1130 000220 012737 000777 001160 MOV #777,$TMPO
1131 000226 000137 004060 JMP @START
1132 .SBTTL ACT11 HOOKS
1133 ;*****
;HOOKS REQUIRED BY ACT11
000232 $SVPC=. ;SAVE PC
000046 .=46
031732 $ENDAD ;:1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
000052 .=52
000000 .WORD 0 ;:2)SET LOC.52 TO ZERO
000232 .=$SVPC ;; RESTORE PC
1134 .SBTTL APT PARAMETER BLOCK
;*****
;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
;*****
000232 . $X=. ;;SAVE CURRENT LOCATION
000024 .=24 ;;SET POWER FAIL TO POINT TO START OF PROGRAM
000200 200 ;;FOR APT START UP
000044 .=44 ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044 $APTHDR ;;POINT TO APT HEADER BLOCK
000232 .=$X ;;RESET LOCATION COUNTER
;*****
;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
;INTERFACE SPEC.
000232 $APTHD:
000232 000000 $HIBTS: .WORD 0 ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.

```

E3

APT PARAMETER BLOCK

000234 001200
000236 000000
000240 000000
000242 000000
000244 000052

\$MBADR: .WORD \$MAIL ;; ADDRESS OF APT MAILBOX (BITS 0 15)
\$TSTM: .WORD ;; RUN TIM OF LONGEST TEST
\$PASTM: .WORD ;; RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
\$UNITM: .WORD ;; ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
\$ETEND \$MAIL/2 ;; LENGTH MAILBOX-ETABLE(WORDS)

COMMON TAGS

1135

```

.SBTTL COMMON TAGS
;*****
;THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
;USED IN THE PROGRAM.
.=1100
001100 001100 $CMTAG: .WORD 0 ;;START OF COMMON TAGS
001100 000000 $TSTNM: .BYTE 0 ;;CONTAINS THE TEST NUMBER
001102 000 $ERFLG: .BYTE 0 ;;CONTAINS ERROR FLAG
001103 000 $ICNT: .WORD 0 ;;CONTAINS SUBTEST ITERATION COUNT
001104 000000 $LPADR: .WORD 0 ;;CONTAINS SCOPE LOOP ADDRESS
001106 000000 $LPERR: .WORD 0 ;;CONTAINS SCOPE RETURN FOR ERRORS
001110 000000 $ERTTL: .WORD 0 ;;CONTAINS TOTAL ERRORS DETECTED
001112 000000 $ITEMB: .BYTE 0 ;;CONTAINS ITEM CONTROL BYTE
001114 000 $ERMAX: .BYTE 1 ;;CONTAINS MAX. ERRORS PER TEST
001115 001 $ERRPC: .WORD 0 ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001116 000000 $GDADR: .WORD 0 ;;CONTAINS ADDRESS OF 'GOOD' DATA
001120 000000 $BDADR: .WORD 0 ;;CONTAINS ADDRESS OF 'BAD' DATA
001122 000000 $GDDAT: .WORD 0 ;;CONTAINS 'GOOD' DATA
001124 000000 $BDDAT: .WORD 0 ;;CONTAINS 'BAD' DATA
001126 000000 .WORD 0 ;;RESERVED--NOT TO BE USED
001130 000000 .WORD 0
001132 000000 .WORD 0
001134 000 $AUTOB: .BYTE 0 ;;AUTOMATIC MODE INDICATOR
001135 000 $INTAG: .BYTE 0 ;;INTERRUPT MODE INDICATOR
001136 000000 .WORD 0
001140 177570 SWR: .WORD DSWR ;;ADDRESS OF SWITCH REGISTER
001142 177570 DISPLAY: .WORD DDISP ;;ADDRESS OF DISPLAY REGISTER
001144 177560 $TKS: 177560 ;;TTY KBD STATUS
001146 177562 $TKB: 177562 ;;TTY KBD BUFFER
001150 177564 $TPS: 177564 ;;TTY PRINTER STATUS REG. ADDRESS
001152 177566 $TPB: 177566 ;;TTY PRINTER BUFFER REG. ADDRESS
001154 000 $NULL: .BYTE 0 ;;CONTAINS NULL CHARACTER FOR FILLS
001155 002 $FILLS: .BYTE 2 ;;CONTAINS # OF FILLER CHARACTERS REQUIRED
001156 012 $FILLC: .BYTE 12 ;;INSERT FILL CHARS. AFTER A "LINE FEED"
001157 000 $TPFLG: .BYTE 0 ;;"TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)
. REPT 2
001160 000000 $TMPO: .WORD 0 ;;USER DEFINED
001162 000000 $TMP1: .WORD 0 ;;USER DEFINED
001164 000000 $TIMES: 0 ;;MAX. NUMBER OF ITERATIONS
001166 000000 $ESCAPE: 0 ;;ESCAPE ON ERROR ADDRESS
001170 207 377 377 $BELL: .ASCIZ <207><377><377> ;;CODE FOR BELL
001173 000
001174 077 $QUES: .ASCII /?/ ;;QUESTION MARK
001175 015 $CRLF: .ASCII <15> ;;CARRIAGE RETURN
001176 012 000 $LF: .ASCIZ <12> ;;LINE FEED
;*****
.SBTTL APT MAILBOX-ETABLE
;*****
.EVEN
001200 $MAIL: ;;APT MAILBOX
001200 000000 $MSGTY: .WORD AMSGTY ;;MESSAGE TYPE CODE
001202 000000 $FATAL: .WORD AFATAL ;;FATAL ERROR NUMBER
001204 000000 $TESTN: .WORD ATESTN ;;TEST NUMBER
001206 000000 $PASS: .WORD APASS ;;PASS COUNT
001210 000000 $DEVCT: .WORD ADEVCT ;;DEVICE COUNT
001212 000000 $UNIT: .WORD AUNIT ;;I/O UNIT NUMBER
001214 000000 $MSGAD: .WORD AMSGAD ;;MESSAGE ADDRESS

```


APT MAILBOX-ETABLE

001216	000000	\$MSGLG: .WORD	AMSGLG	::MESSAGE LENGTH
001220		\$ETABLE:		::APT ENVIRONMENT TABLE
001220	000	\$ENV: .BYTE	AENV	::ENVIRONMENT BYTE
001221	000	\$ENVM: .BYTE	AENVM	::ENVIRONMENT MODE BITS
001222	000000	\$SWREG: .WORD	ASWREG	::APT SWITCH REGISTER
001224	000000	\$USWR: .WORD	AUSWR	::USER SWITCHES
001226	000000	\$CPUOP: .WORD	ACPUOP	::CPU TYPE,OPTIONS
		*		BITS 15 11=CPU TYPE
		*		11/04=01,11/05=02,11/20=03,11/40=04,11/45=05
		*		11/70=06,PDQ=07,Q=10
		*		BIT 10=REAL TIME CLOCK
		*		BIT 9=FLOATING POINT PROCESSOR
		*		BIT 8=MEMORY MANAGEMENT
001230	000	\$MAMS1: .BYTE	AMAMS1	::HIGH ADDRESS,M.S. BYTE
001231	000	\$MTYP1: .BYTE	AMTYP1	::MEM. TYPE,BLK#1
		*		MEM.TYPE BYTE (HIGH BYTE)
		*		900 NSEC CORE=001
		*		300 NSEC BIPOLAR=002
		*		500 NSEC MOS=003
001232	000000	\$MADR1: .WORD	AMADR1	::HIGH ADDRESS,BLK#1
		*		MEM.LAST ADDR.=3 BYTES,THIS WORD AND LOW OF "TYPE" ABOVE
001234	000	\$MAMS2: .BYTE	AMAMS2	::HIGH ADDRESS,M.S. BYTE
001235	000	\$MTYP2: .BYTE	AMTYP2	::MEM.TYPE,BLK#2
001236	000000	\$MADR2: .WORD	AMADR2	::MEM.LAST ADDRESS,BLK#2
001240	000	\$MAMS3: .BYTE	AMAMS3	::HIGH ADDRESS,M.S.BYTE
001241	000	\$MTYP3: .BYTE	AMTYP3	::MEM.TYPE,BLK#3
001242	000000	\$MADR3: .WORD	AMADR3	::MEM.LAST ADDRESS,BLK#3
001244	000	\$MAMS4: .BYTE	AMAMS4	::HIGH ADDRESS,M.S.BYTE
001245	000	\$MTYP4: .BYTE	AMTYP4	::MEM.TYPE,BLK#4
001246	000000	\$MADR4: .WORD	AMADR4	::MEM.LAST ADDRESS,BLK#4
001250	000000	\$VECT1: .WORD	AVECT1	::INTERRUPT VECTOR#1,BUS PRIORITY#1
001252	000000	\$VECT2: .WORD	AVECT2	::INTERRUPT VECTOR#2BUS PRIORITY#2
001254	000000	\$BASE: .WORD	ABASE	::BASE ADDRESS OF EQUIPMENT UNDER TEST
001256	000000	\$DEVM: .WORD	ADEVM	::DEVICE MAP
001260	000000	\$CDW1: .WORD	ACDW1	::CONTROLLER DESCRIPTION WORD#1
001262	000000	\$CDW2: .WORD	ACDW2	::CONTROLLER DESCRIPTION WORD#2
001264	000000	\$DDW0: .WORD	ADDW0	::DEVICE DESCRIPTOR WORD#0
001266	000000	\$DDW1: .WORD	ADDW1	::DEVICE DESCRIPTOR WORD#1
001270	000000	\$DDW2: .WORD	ADDW2	::DEVICE DESCRIPTOR WORD#2
001272	000000	\$DDW3: .WORD	ADDW3	::DEVICE DESCRIPTOR WORD#3
001274	000000	\$DDW4: .WORD	ADDW4	::DEVICE DESCRIPTOR WORD#4
001276	000000	\$DDW5: .WORD	ADDW5	::DEVICE DESCRIPTOR WORD#5
001300	000000	\$DDW6: .WORD	ADDW6	::DEVICE DESCRIPTOR WORD#6
001302	000000	\$DDW7: .WORD	ADDW7	::DEVICE DESCRIPTOR WORD#7
001304	000000	\$DDW8: .WORD	ADDW8	::DEVICE DESCRIPTOR WORD#8
001306	000000	\$DDW9: .WORD	ADDW9	::DEVICE DESCRIPTOR WORD#9
001310	000000	\$DDW10: .WORD	ADDW10	::DEVICE DESCRIPTOR WORD#10
001312	000000	\$DDW11: .WORD	ADDW11	::DEVICE DESCRIPTOR WORD#11
001314	000000	\$DDW12: .WORD	ADDW12	::DEVICE DESCRIPTOR WORD#12
001316	000000	\$DDW13: .WORD	ADDW13	::DEVICE DESCRIPTOR WORD#13
001320	000000	\$DDW14: .WORD	ADDW14	::DEVICE DESCRIPTOR WORD#14
001322	000000	\$DDW15: .WORD	ADDW15	::DEVICE DESCRIPTOR WORD#15
001324		\$ETEND:		

ERROR POINTER TABLE

```

.SBTTL 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 $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
;*NOTE1: IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
;*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
$ERRTB:

```

001324

```

1136
1137
1138
1139
1140 001324 020617
1141 001326 025314
1142 001330 026502
1143 001332 000000
1144
1145 001334 020653
1146 001336 025314
1147 001340 026502
1148 001342 000000
1149
1150 001344 020665
1151 001346 025314
1152 001350 026502
1153 001352 000000
1154
1155 001354 020677
1156 001356 025314
1157 001360 026502
1158 001362 000000
1159
1160 001364 020735
1161 001366 025314
1162 001370 026502
1163 001372 000000
1164
1165 001374 020761
1166 001376 025314
1167 001400 026502
1168 001402 000000
1169
1170 001404 021073
1171 001406 025314
1172 001410 026502
1173 001412 000000
1174
1175 001414 021132
1176 001416 026145
1177 001420 026736
1178 001422 000000
1179
1180 001424 021202
1181 001426 025314

```

```

.SBTTL ERROR DEFINITIONS
;ITEM 1
EM1      ;CPU ERROR
DH1      ;TEST #, ERROR PC
DT1      ;$TMP1,$ERRPC
0
;ITEM 2
EM2      ;MMU ERROR
DH1      ;TEST #, ERROR PC
DT1      ;$TMP1,$ERRPC
0
;ITEM 3
EM3      ;FPP ERROR
DH1      ;TEST #, ERROR PC
DT1      ;$TMP1,$ERRPC
0
;ITEM 4
EM54     ;CHECKSUM ERROR IN 16 BIT ROM
DH1      ;TEST #, PC
DT1      ;$TMP1,$ERRPC
0
;ITEM 5
EM56     ;TIMEOUT READING LKS
DH1      ;TEST #, PC
DT1      ;$TMP1,$ERRPC
0
;ITEM 6
EM57     ;LKS<07> DOES NOT BECOME 1
DH1      ;TEST #, PC
DT1      ;$TMP1,$ERRPC
0
;ITEM 7
EM64     ;WRONG NUMBER OF LKS INTERRUPTS
DH1      ;TEST #, PC
DT1      ;$TMP1,$ERRPC
0
;ITEM 10
EM65     ;LKS INTERRUPTS HAPPEN AT WRONG PRIORITY
DH65     ;TEST #, PC, PRIORITY
DT65     ;$TMP1,$ERRPC,$GDDAT
0
;ITEM 11
EM71     ;RESET DOESN'T CLEAR LKS<06>
DH1      ;TEST #, PC

```

ERROR DEFINITIONS

```

1182 001430 026502          DT1          :$TMP1,$ERRPC
1183 001432 000000          0
1184
1185 001434 021236          ;ITEM 12  EM72          :TIMEOUT READING SLU REGISTERS
1186 001436 026211          DH72          :TEST #, PC, ADDRESS FAILED
1187 001440 026630          DT41          :$TMP1,$ERRPC,$BDDAT
1188 001442 000000          0
1189
1190 001444 021274          ;ITEM 13  EM73          :XMIT READY DIDN'T GO LOW
1191 001446 025314          DH1          :TEST #, PC
1192 001450 026502          DT1          :$TMP1,$ERRPC
1193 001452 000000          0
1194
1195 001454 021320          ;ITEM 14  EM74          :RCSR DOESN'T BECOME 1
1196 001456 025314          DH1          :TEST #, PC
1197 001460 026502          DT1          :$TMP1,$ERRPC
1198 001462 000000          0
1199
1200 001464 021351          ;ITEM 15  EM75          :WRONG CHARACTER RECEIVED
1201 001466 025341          DH4          :TEST #, PC, EXPECTED DATA, RECEIVED DATA
1202 001470 026746          DT75          :$TMP1,$ERRPC,$GDDAT,$BDDAT
1203 001472 000000          0
1204
1205 001474 021402          ;ITEM 16  EM76          :RCSR<07> NOT CLEARED AFTER READING RBUF
1206 001476 025314          DH1          :TEST #, PC
1207 001500 026502          DT1          :$TMP1,$ERRPC
1208 001502 000000          0
1209
1210 001504 021452          ;ITEM 17  EM77          :XCSR<07> NOT SET ON RESET
1211 001506 025314          DH1          :TEST #, PC
1212 001510 026502          DT1          :$TMP1,$ERRPC
1213 001512 000000          0
1214
1215 001514 021504          ;ITEM 20  EM100         :RCSR<07> NOT CLEARED ON RESET
1216 001516 025314          DH1          :TEST #, PC
1217 001520 026502          DT1          :$TMP1,$ERRPC
1218 001522 000000          0
1219
1220 001524 021542          ;ITEM 21  EM101         :SLU INTERRUPTS HAPPEN AT 4
1221 001526 025314          DH1          :TEST #, PC
1222 001530 026502          DT1          :$TMP1,$ERRPC
1223 001532 000000          0
1224
1225 001534 021575          ;ITEM 22  EM102         :RESET DOES NOT CLEAR XCSR<6> AND RSCR<6>
1226 001536 025314          DH1          :TEST #, PC
1227 001540 026502          DT1          :$TMP1,$ERRPC
1228 001542 000000          0
1229
1230 001544 021657          ;ITEM 23  EM103         :TRANSMIT INTERRUPT DOES NOT CLEAR XCSR<07>
1231 001546 025314          DH1          :TEST #, PC
1232 001550 026502          DT1          :$TMP1,$ERRPC
1233 001552 000000          0
1234
1235 001554 021732          ;ITEM 24  EM104         :RECEIVE INTERRUPTS DON'T CLEAR RCSR<07>
1236 001556 025314          DH1          :TEST #, PC
1237 001560 026502          DT1          :$TMP1,$ERRPC
1238 001562 000000          0

```

ERROR DEFINITIONS

1239					
1240	001564	022002	;ITEM 25	EM105	:BREAK CONDITION DOES NOT SET RBUF PROPERLY
1241	001566	026255		DH105	:TEST #, PC, RBUF
1242	001570	026760		DT105	:\$TMP1,\$ERRPC,RBUF
1243	001572	000000		0	
1244			;ITEM 26	EM106	:RBUF WASN'T CLEARED ON NEXT CHAR.
1245	001574	022055		DH105	:TEST #, PC, RBUF
1246	001576	026255		DT105	:\$TMP1,\$ERRPC,RBUF
1247	001600	026760		0	
1248	001602	000000	;ITEM 27	EM107	:ERROR IN WRITING TO XCSR<0>
1249				DH1	:TEST #, PC
1250	001604	022133		DT1	:\$TMP1,\$ERRPC
1251	001606	025314		0	
1252	001610	026502	;ITEM 30	EM110	:RESET DOES NOT CLEAR XCSR<00>
1253	001612	000000		DH1	:TEST #, PC
1254				DT1	:\$TMP1,\$ERRPC
1255	001614	022167		0	
1256	001616	025314	;ITEM 31	EM111	:FIRST CHARACTER WAS NOT OVERRUN BY THE SECOND
1257	001620	026502		DH4	:TEST #, PC, EXPECTED DATA, RECEIVED DATA
1258	001622	000000		DT75	:\$TMP1,\$ERRPC,\$GDDAT,\$BDDAT
1259				0	
1260	001624	022231	;ITEM 32	EM112	:OVERRUN CONDITION DOES NOT SET PROPER BITS IN RBUF
1261	001626	025341		DH105	:TEST #, PC, RBUF
1262	001630	026746		DT105	:\$TMP1,\$ERRPC,RBUF
1263	001632	000000		0	
1264			;ITEM 33	EM113	:RBUF WAS NOT CLEARED ON THE NEXT CHARACTER
1265	001634	022307		DH105	:TEST #, PC, RBUF
1266	001636	026255		DT105	:\$TMP1,\$ERRPC,RBUF
1267	001640	026760		0	
1268	001642	000000	;ITEM 34	EM114	:ERROR IN XCSR<2>
1269				DH1	:TEST #, PC
1270	001644	022372		DT1	:\$TMP1,\$ERRPC
1271	001646	026255		0	
1272	001650	026760	;ITEM 35	EM124	:PIRQ INTERRUPTS DON'T TAKE PRIORITY OVER Q BUS INTERRUPTS
1273	001652	000000		DH1	:TEST #, PC
1274				DT1	:\$TMP1,\$ERRPC
1275	001654	022456		0	
1276	001656	025314	;ITEM 36	EM125	:NO POWER DOWN TRAP TO 24 OCCUR
1277	001660	026502		DH1	:TEST #, PC
1278	001662	000000		DT1	:\$TMP1,\$ERRPC
1279				0	
1280	001664	022531	;ITEM 37	EM126	:ERROR IN INTERRUPTS FROM Q22BE
1281	001666	025314		DH1	:TEST #, PC
1282	001670	026502		DT1	:\$TMP1,\$ERRPC
1283	001672	000000		0	
1284			;ITEM 40	EM127	:ERROR IN PMG COUNTER
1285	001674	022623			
1286	001676	025314			
1287	001700	026502			
1288	001702	000000			
1289					
1290	001704	022662			
1291	001706	025314			
1292	001710	026502			
1293	001712	000000			
1294					
1295	001714	022717			

ERROR DEFINITIONS

1296	001716	025314	DH1	;TEST #, PC
1297	001720	026502	DT1	;\$TMP1,\$ERRPC
1298	001722	000000	0	
1299			:ITEM 41	
1300	001724	022761	EM130	;UNEXPECTED TIMFOUT
1301	001726	025314	DH1	;TEST #, PC
1302	001730	027004	DT130	;\$TMP1,\$ERRPC
1303	001732	000000	0	
1304			:ITEM 42	
1305	001734	023006	EM131	;ERROR WRITING TO LKS<6>
1306	001736	025314	DH1	;TEST #, PC
1307	001740	026502	DT1	;\$TMP1,\$ERRPC
1308	001742	000000	0	
1309			:ITEM 43	
1310	001744	023036	EM132	;MAINTENANCE REGISTER ERROR
1311	001746	025314	DH1	;TEST #, PC
1312	001750	026502	DT1	;\$TMP1,\$ERRPC
1313	001752	000000	0	
1314			:ITEM 44	
1315	001754	023074	EM135	; ERROR IN THE DATA PATHS
1316	001756	025341	DH4	;TEST #, PC, EXPECTED DATA, RECEIVED DATA
1317	001760	026746	DT75	;\$TMP1,\$ERRPC,\$GDDAT,\$BDDAT
1318	001762	000000	0	
1319			:ITEM 45	
1320	001764	023132	EM136	;TIMED OUT IN ACCESSING LOCATION 0
1321	001766	025314	DH1	;TEST #, PC
1322	001770	026502	DT1	;\$TMP1,\$ERRPC
1323	001772	000000	0	
1324			:ITEM 46	
1325	001774	023174	EM137	;TIMED OUT IN TRYING TO ACCESS MEMORY
1326	001776	025314	DH1	;TEST #, PC
1327	002000	026502	DT1	;\$TMP1,\$ERRPC
1328	002002	000000	0	
1329			:ITEM 47	
1330	002004	023241	EM140	;ERROR IN FUNCTIONAL REV BITS ON NATIVE REGISTER
1331	002006	025314	DH1	;TEST #, PC
1332	002010	026502	DT1	;\$TMP1,\$ERRPC
1333	002012	000000	0	
1334			:ITEM 50	
1335	002014	023332	EM141	;ERROR IN INDICATOR BITS ON THE NATIVE REGISTER
1336	002016	025314	DH1	;TEST #, PC
1337	002020	026502	DT1	;\$TMP1,\$ERRPC
1338	002022	000000	0	
1339			:ITEM 51	
1340	002024	023415	EM142	;ERROR IN THE BOOT SELECT SWITCHES ON THE NATIVE REGISTER
1341	002026	025314	DH1	;TEST #, PC
1342	002030	026502	DT1	;\$TMP1,\$ERRPC
1343	002032	000000	0	
1344			:ITEM 52	
1345	002034	023506	EM143	;TIMED OUT IN ACCESSING THE NATIVE REGISTER
1346	002036	025314	DH1	;TEST #, PC
1347	002040	026502	DT1	;\$TMP1,\$ERRPC
1348	002042	000000	0	
1349			:ITEM 53	
1350	002044	023561	EM144	;LTC MONITOR IS NOT TOGGLING
1351	002046	025314	DH1	;TEST #, PC
1352	002050	026502	DT1	;\$TMP1,\$ERRPC

ERROR DEFINITIONS

1353	002052	000000	0	
1354			:ITEM 54	
1355	002054	023636	EM145	:TIMEOUT IN ACCESSING THE LKS REGISTER
1356	002056	025314	DH1	:TEST #, PC
1357	002060	026502	DT1	:\$TMP1,\$ERRPC
1358	002062	000000	0	
1359			:ITEM 55	
1360	002064	023704	EM146	:ERROR IN STUCK AT ZERO BITS 0.1 MER
1361	002066	025314	DH1	:TEST #, PC
1362	002070	026502	DT1	:\$TMP1,\$ERRPC
1363	002072	000000	0	
1364			:ITEM 56	
1365	002074	023747	EM147	:COULD NOT SET ONE OF THE R/W BITS ON THE MER
1366	002076	025341	DH4	:TEST #, PC, EXPECTED DATA, RECEIVED DATA
1367	002100	026746	DT75	:\$TMP1,\$ERRPC,\$GDDAT,\$BDDAT
1368	002102	000000	0	
1369			:ITEM 57	
1370	002104	024024	EM150	:BITS 0,2,14,15 ON THE MER DID NOT CLEAR ON RESET
1371	002106	025314	DH1	:TEST #, PC
1372	002110	026502	DT1	:\$TMP1,\$ERRPC
1373	002112	000000	0	
1374			:ITEM 60	
1375	002114	024105	EM151	:TIMEOUT IN ACCESSING THE MER REGISTER
1376	002116	025314	DH1	:TEST #, PC
1377	002120	026502	DT1	:\$TMP1,\$ERRPC
1378	002122	000000	0	
1379			:ITEM 61	
1380	002124	024153	EM152	:ERROR IN THE DATA SHORTS AND STUCK AT MEMORY TEST
1381	002126	026364	DH134	:TEST #, PC, PATTERN, DATA, PAR, VIRTUAL ADDRESS
1382	002130	027012	DT134	:\$TMP1,\$ERRPC,\$GDDAT,\$BDDAT,KIPAR2,\$BDADR
1383	002132	000000	0	
1384			:ITEM 62	
1385	002134	024235	EM153	:PARITY ABORT OCCURED WITH PARITY ERROR DISABLED
1386	002136	025314	DH1	:TEST #, PC
1387	002140	026502	DT1	:\$TMP1,\$ERRPC
1388	002142	000000	0	
1389			:ITEM 63	
1390	002144	024315	EM154	:PARITY ABORT DID NOT OCCUR WITH PARITY ERROR ENABLED
1391	002146	025314	DH1	:TEST #, PC
1392	002150	026502	DT1	:\$TMP1,\$ERRPC
1393	002152	000000	0	
1394			:ITEM 64	
1395	002154	024402	EM155	:MER DIDN'T HAVE CORRECT ADDRESS BITS 11 17
1396	002156	025341	DH4	:TEST #, PC, EXPECTED DATA, RECEIVED DATA
1397	002160	026746	DT75	:\$TMP1,\$ERRPC,\$GDDAT,\$BDDAT
1398	002162	000000	0	
1399			:ITEM 65	
1400	002164	024455	EM156	:MER READ EXTENDED ADDRESS BITS HAS FAILED
1401	002166	025341	DH4	:TEST #, PC, EXPECTED DATA, RECEIVED DATA
1402	002170	026746	DT75	:\$TMP1,\$ERRPC,\$GDDAT,\$BDDAT
1403	002172	000000	0	
1404			:ITEM 66	
1405	002174	024527	EM157	:ERROR IN THE MEMORY
1406	002176	026364	DH134	:TEST #, PC, PATTERN, DATA, PAR, VIRTUAL ADDRESS
1407	002200	027012	DT134	:\$TMP1,\$ERRPC,\$GDDAT,\$BDDAT,KIPAR2,\$BDADR
1408	002202	000000	0	
1409			:ITEM 67	

ERROR DEFINITIONS

1410	002204	024575	EM160	:ERROR IN XCSR <6>
1411	002206	025314	DH1	:TEST #, PC
1412	002210	026502	DT1	:\$TMP1,\$ERRPC
1413	002212	000000	0	
1414			:ITEM 70	
1415	002214	024617	EM161	:NO XMIT INTERRUPTS HAVE OCCURED
1416	002216	025314	DH1	:TEST #, PC
1417	002220	026502	DT1	:\$TMP1,\$ERRPC
1418	002222	000000	0	
1419			:ITEM 71	
1420	002224	024657	EM162	:NO RECIEVE INTERRUPTS HAVE OCCURED
1421	002226	025314	DH1	:TEST #, PC
1422	002230	026502	DT1	:\$TMP1,\$ERRPC
1423	002232	000000	0	
1424			:ITEM 72	
1425	002234	024722	EM163	:UNEXPECTED PARITY ABORT HAS OCCURED
1426	002236	025314	DH1	:TEST #, PC
1427	002240	026502	DT1	:\$TMP1,\$ERRPC
1428	002242	000000	0	
1429			:ITEM 73	
1430	002244	024766	EM164	:MER DIDN'T HAVE CORRECT ADDRESS BITS 0 AND 15
1431	002246	025341	DH4	:TEST#, PC, EXTENDED DATA, RECEIVED DATA
1432	002250	026746	DT75	:\$TMP1,\$ERRPC,\$GDDAT,\$BDDAT
1433	002252	000000	0	
1434			:ITEM 74	
1435			EM165	:TOO MANY TRANSIVER INTERRUPTS HAPPENED
1436	002254	025045	DH1	:TEST #, PC
1437	002256	025314	DT1	:\$TMP1,\$ERRPC
1438	002260	026502	0	
1439	002262	000000	:ITEM 75	
1440			EM166	:TOO MANY RECEIVER INTERRUPTS HAPPENED
1441	002264	025114	DH1	:TEST #, PC
1442	002266	025314	DT1	:\$TMP1,\$ERRPC
1443	002270	026502	0	
1444	002272	000000	:item 76	
1445			em167	:error in ready bit of csr
1446	002274	025162	dh1	:test#, pc
1447	002276	025314	dt1	:\$tmp1,\$errpc
1448	002300	026502	0	
1449	002302	000000	:item 77	
1450			em170	:no character received
1451	002304	025214	dh1	:test#,pc
1452	002306	025314	dt1	:\$tmp1,\$errpc
1453	002310	026502	0	
1454	002312	000000	:item 100	
1455			em171	:wrong character received
1456	002314	025242	dh1	:test#,pc
1457	002316	025314	dt1	:\$tmp1,errpc
1458	002320	026502	0	
1459	002322	000000	:item 101	
1460			em172	:error rom size is zero
1461	002324	025273	dh1	:test#,pc
1462	002326	025314	dt1	:\$tmp1,errpc
1463	002330	026502	0	
1464	002332	000000		
1465				
1466				

ERROR DEFINITIONS

```

1467
1468      .SBTTL GLOBAL VARIABLES AND REGISTER NAMES
1469
1470      ;REGISTERS FOR THE FIRST Q22BE
1471 002334 000000 CSR1: .WORD 0 ;CONTROL REGISTER 1 FOR Q22BE
1472 002336 000000 CSR2: .WORD 0 ;CONTROL/STATUS REGISTER 2
1473 002340 000000 BA: .WORD 0 ;DMA ADDRESS FOR Q22BE
1474 002342 000000 WC: .WORD 0 ;WORD COUNT REGISTER
1475 002344 000000 DATA: .WORD 0 ;DMA DATA FOR Q22BE
1476 002346 000000 VQBE1: .WORD 0 ;ADDRESS OF VECTOR FOR Q22BE
1477 002350 000000 VQPR1: .WORD 0 ;PRIORITY
1478 002352 000000 SIMGOA: .WORD 0 ;SIMULTANEOUS GO ADDRESS REGISTER
1479
1480 002354 000000 ledcnt: .word 0 ;location for the led count ;mc
1481 002356 000000 RCOUNT: .WORD 0 ;RECEIVER INTERRUPT COUNT ;MC
1482 002360 000000 TCOUNT: .WORD 0 ;TRANSMITTER INTERRUPT COUNT ;MC
1483 002362 000000 lopbak: .word 0 ;flag for the loop back connector;mc
1484
1485      ;REGISTERS FOR THE SECOND Q22BE
1486 002364 000000 CSR12: .WORD 0 ;CONTROL REGISTER 1 FOR Q22BE
1487 002366 000000 CSR22: .WORD 0 ;CONTROL/STATUS REGISTER 2
1488 002370 000000 BA2: .WORD 0 ;DMA ADDRESS FOR Q22BE
1489 002372 000000 WC2: .WORD 0 ;WORD COUNT REGISTER
1490 002374 000000 DATA2: .WORD 0 ;DMA DATA FOR Q22BE
1491 002376 000000 VQBE2: .WORD 0 ;ADDRESS OF VECTOR FOR Q22BE
1492 002400 000000 VQPR2: .WORD 0 ;PRIORITY
1493
1494 002402 000000 LKSFL: .WORD 0
1495 002404 000000 lkcnt: .word 0
1496 002406 000000 savloc: .word 0
1497 002410 000000 ACTCHS: .WORD 0 ;ACTUAL CHECKSUM
1498 002412 000000 SAVPCR: .WORD 0
1499 002414 000000 SAVBR: .WORD 0
1500 002740 002740 .=2740
1501 002740 000000 TEMP: .WORD 0
1502 002742 000000 .BLKW 15. ;RESERVED FOR BLOCK MODE TRANSFER
1503 003000 000000 TIMEOUT: .WORD 0
1504 003002 000002 Q22EN: .WORD 2 ;PRIORITY 7-4 FOR Q22BE
1505
1506
1507 003004 377 021 377 tchar: .byte -1,21,-1,0 ;string for the loop back connector test
1508 003007 000
1508
1509 177524 BCR= 177524 ;BOOT/DIAGNOSTICS CONFIGURATION
1510 177524 BDR= 177524 ;BOOT/DIAGNOSTICS DISPLAY
1511 177520 BCSR= 177520 ;BOOT/DIAGNOSTICS STATUS
1512 177746 CCR= 177746 ;CACHE CONTROL REGISTER
1513 177752 HITMIS= 177752 ;HIT OR MISS REGISTER
1514 177734 KMCR= 177734 ;UNIBUS CONFIGURATION REGISTER
1515 177546 LKS= 177546 ;CLOCK STATUS REGISTER
1516 177750 MAIREG= 177750 ;MAINTENANCE REGISTER
1517 177520 NATREG= 177520 ;NATIVE REGISTER
1518 172100 MER= 172100 ;MEMORY ERROR REGISTER
1519 177744 MSER= 177744 ;MEMORY SYSTEM ERROR
1520 177522 PCR= 177522 ;PAGE CONTROL REGISTER
1521 177772 PIR= 177772 ;PROGRAM INTERRUPT REQUEST
1522 177560 RCSR= 177560 ;RECEIVER STATUS REGISTER

```


GLOBAL VARIABLES AND REGISTER NAMES

```

1523      177562      RBUF =          177562      ;RECEIVER DATA BUFFER
1524      177564      XCSR =          177564      ;TRANSMITTER STATUS REGISTER
1525      177566      XBUF =          177566      ;TRANSMITTER DATA BUFFER
1526      176500      RCSR1 =         176500      ;RECEIVER STATUS REGISTER
1527      176502      RBUF1 =         176502      ;RECEIVER DATA BUFFER
1528      176504      XCSR1 =         176504      ;TRANSMITTER STATUS REGISTER
1529      176506      XBUF1 =         176506      ;TRANSMITTER DATA BUFFER
1530
1531      177766      CPEREG =          177766      ;CPU ERROR REGISTER
1532
1533      177572      MMRO = SR0          ;MEMORY MANAGEMENT REG. 0
1534      177574      MMR1 = SR1          ;MEMORY MANAGEMENT REG. 1
1535      177576      MMR2 = SR2          ;MEMORY MANAGEMENT REG. 2
1536      172516      MMR3 = SR3          ;MEMORY MANAGEMENT REG. 3
1537      120001      POLY = 120001
1538      000000      NULL =              0
1539      100000      begrom =           100000
1540
1541      .SBTTL GLOBAL DATA SECTION
1542
1543      ;++
1544      ; THE GLOBAL DATA SECTION CONTAINS DATA THAT ARE USED
1545      ; IN MORE THAN ONE TEST.
1546      ;
1547
1548      ;THESE LOCATIONS ARE USED IN MORE THAN ONE TEST TO STORE VECTOR DATA
1549      ;WHEN THE TEST NEEDS TO HAVE AN ERROR CONDITION RESPOND DIFFERENTLY
1550      ;FROM THE DEFAULT RESPONCE.
1551      003010      000000      SLOC00: .WORD 0
1552      003012      000000      SLOC01: .WORD 0
1553
1554      ;THESE LOCATIONS ARE USED IN MORE THAN ONE TEST TO STORE WORKING DATA.
1555      003014      000000      PARPAT: .WORD 0      ;LOCATION TO SAVE PATTERN USED IN PARITY TEST
1556      003016      000000      SAVTIM: .WORD 0      ;LOCATION TO THE SAVE THE TIMEOUT TRAP
1557      003020      000000      LOWADD: .WORD 0      ;STORES LOW ADDRESS FOR RAM TESTS
1558      003022      000000      GOODAD: .WORD 0      ;STORES GOOD ADDRESS FOR RAM TESTS
1559      003024      000000      ERRCNT: .WORD 0      ; CONTAINS TOTAL NO. OF EEROM ERRORS
1560      003026      000000      TSTADD: .WORD 0      ;ADDRESS STORE FOR RAM TESTS
1561      003030      000000      NEWADD: .WORD 0      ;ADDRESS STORE FOR RAM TEST
1562      003032      000000      FLAG: .WORD 0      ;USED TO STORE "FLAG" CONDITIONS
1563      003034      000000      CCHPAS: .WORD 0      ; flag-counter for control of Cache subtests
1564      003036      000000      EEPAS: .WORD 0      ; flag-counter for control of EEROM subtest
1565      003040      000000      SAVSUP: .WORD 0      ;USED TO STORE SUPERVISOR STACK VALUE
1566      003042      000000      SAVUSE: .WORD 0      ;USED TO STORE USER STACK VALUE
1567      003044      000000      SAVMRO: .WORD 0      ;USED TO STORE MMU STATUS REGISTER 0 DATA
1568      003046      000000      SAVMR1: .WORD 0      ;USED TO STORE MMU STATUS REGISTER 1 DATA
1569      003050      000000      SAVMR2: .WORD 0      ;USED TO STORE MMU STATUS REGISTER 2 DATA
1570      003052      000000      SAVSWR: .WORD 0      ; SAVE SFTWRE SWTCH REG DURING EEROM TEST
1571      003054      000000      MERTAG: .WORD 0
1572      003056      .BLKW 4      ;USED TO STORE VALUES FOR MMU TESTS
1573      003066      .BLKW 4      ;USED TO STORE VALUES FOR MMU TESTS
1574      003076      000000      SEQ: .WORD 0      ;STORES SEQUENCE NUMBER FOR JUMP TESTS
1575      003100      000000      SPS: .WORD 0      ;STORES STACK POINTER FOR JUMP TESTS
1576      003102      000000      SPSJ: .WORD 0      ;STORES STACK POINTER FOR JUMP TESTS
1577      003104      .BLKW 4      ;STORES EXPONENT DURING BIT TESTS
1578      003114      .BLKW 4      ;STORES RECIEVED DATA FOR BIT TESTS
1579      003124      000000      COUNT: .WORD 0      ;ERROR INDICATOR FOR FLOATING POINT TESTS

```

C4

GLOBAL DATA SECTION

```

1580 003126      RECFC: .BLKW  4      ;RECIEVED FLOATING POINT EXCEPTION CODE
1581 003136      RECST: .BLKW  4      ;RECIEVED FLOATING POINT STATUS
1582 003146      RECDST: .BLKW  4     ;DESTINATION ADDRESS FOR FLOATING POINT TESTS
1583
1584             ;THESE LOCATIONS ARE USED BY MORE THAN ONE TEST AS LOOP COUNTERS
1585 003156 000000 ALLCTR: .WORD  0
1586 003160 000000 LOOPIN: .WORD  0
1587
1588             ;SOME MORE TEMPORARY STORAGE FOR RAM TESTS
1589 003162 000000 SAVPCS: .WORD  0      ;STORES TEMPORARY BIT POSITIONS FOR RAM TESTS
1590 003164 000000 MASK:   .WORD  0      ;STORES BIT MASK FOR ERROR ISOLATION
1591
1592             ;!!!!!!THIS IS IT. THE PROGRAM TEST LOCATION!!!!!!!!!!!!!!!!!!!!!!
1593 003166 000000 TSTLOC: .WORD  0
1594 003170             .BLKW  20
1595             ;FPP REGISTER DEFINITIONS
1596             AC0= *0
1597             AC1= *1
1598             AC2= *2
1599             AC3= *3
1600             AC4= *4
1601             AC5= *5
1602             AC6= *6
1603             AC7= *7
1604
1605             ;FPP INTERRUPT VECTOR
1606
1607             FPVEC=244
1608
1609
1610             STBOT= 1000
1611
1612
1614 003240 123456 TAB1:  .WORD  123456
1615 003242 000000      .WORD  000000
1616 003244 000000      .WORD  0
1617 003246 000001      .WORD  1
1618 003250 055555 TAB2:  .WORD  055555
1619 003252 177777      .WORD  177777
1620 003254 145671      .WORD  145671
1621 003256 100000      .WORD  100000
1622 003260 003000 TAB3:  .WORD  003000
1623 003262 123456      .WORD  123456
1624 003264 000000      .WORD  0
1625 003266 000000      .WORD  0
1626 003270 055555 TAB4:  .WORD  55555
1627 003272 177777      .WORD  -1
1628 003274 000000      .WORD  0
1629 003276 000000      .WORD  0
1630 003300 043243 TAB5:  .WORD  43243
1631 003302 000000      .WORD  0
1632 003304 000000      .WORD  0
1633 003306 000000      .WORD  0
1634 003310 162400 TAB5A: .WORD  162400
1635 003312 000000      .WORD  0
1636 003314 000000      .WORD  0
1637 003316 000000      .WORD  0

```

GLOBAL DATA SECTION

1638	003320	000000			TAB6:	.WORD	0
1639	003322	000000				.WORD	0
1640	003324	000000				.WORD	0
1641	003326	000000				.WORD	0
1642	003330	047050			TAB6A:	.WORD	47050
1643	003332	010000				.WORD	10000
1644	003334	000000				.WORD	0
1645	003336	000000				.WORD	0
1646	003340	000200			TAB7:	.WORD	200
1647	003342	000000				.WORD	0
1648	003344	000000				.WORD	0
1649	003346	000000				.WORD	0
1650	003350	000200			TAB8:	.WORD	200
1651	003352	000000				.WORD	0
1652	003354	000000				.WORD	0
1653	003356	000001				.WORD	1
1654	003360	000400	000000	000000	TAB9:	.WORD	400,0,0,0
	003366	000000					
1655	003370	030000			TAB10:	.WORD	30000
1656	003372	003000				.WORD	3000
1657	003374	000000				.WORD	0
1658	003376	000000				.WORD	0
1659	003400	016400			TAB11:	.WORD	16400
1660	003402	000000				.WORD	0
1661	003404	000000				.WORD	0
1662	003406	000000				.WORD	0
1663	003410	030000	003000	000002	TAB11A:	.WORD	30000,3000,2,0
	003416	000000					
1664	003420	016100	000000	000000	TAB12:	.WORD	16100,0,0,1
	003426	000001					
1665	003430	016200			TAB13:	.WORD	16200
1666	003432	000000				.WORD	0
1667	003434	000000				.WORD	0
1668	003436	000001				.WORD	1
1669	003440	030000	003000	000000	TAB13B:	.WORD	30000,3000,0,140000
	003446	140000					
1670	003450	030000			TAB14:	.WORD	30000
1671	003452	000000				.WORD	0
1672	003454	000000				.WORD	0
1673	003456	000000				.WORD	0
1674	003460	024700			TAB15:	.WORD	24700
1675	003462	000000				.WORD	0
1676	003464	000000				.WORD	0
1677	003466	000000				.WORD	0
1678	003470	025000			TAB16:	.WORD	25000
1679	003472	175363				.WORD	175363
1680	003474	123456				.WORD	123456
1681	003476	123456				.WORD	123456
1682	003500	030000			TAB17:	.WORD	30000
1683	003502	007020				.WORD	7020
1684	003504	000000	000000			.WORD	0,0
1685	003510	023456			TAB18:	.WORD	23456
1686	003512	000000				.WORD	0
1687	003514	000000				.WORD	0
1688	003516	000001				.WORD	1
1689	003520	100200	000000	000000	TAB21:	.WORD	100200,0,0,0
	003526	000000					

GLOBAL DATA SECTION

1690	003530	100400	000000	000000	TAB22:	.WORD	100400,0,0,0
	003536	000000					
1691	003540	000200	000000	000000	TAB23:	.WORD	200,0,0,1
	003546	000001					
1692	003550	062400	000000	000000	TAB24:	.WORD	62400,0,0,0
	003556	000000					
1693	003560	001100	000000	000000	TAB25:	.WORD	1100,0,0,0
	003566	000000					
1694	003570	100600	000000	000000	TAB26:	.WORD	100600,0,0,0
	003576	000000					
1695	003600	001000	000000	000000	TAB27:	.WORD	1000,0,0,0
	003606	000000					
1696	003610	000600	000000	000000	TAB28:	.WORD	600,0,0,0
	003616	000000					
1697	003620	010100	000000	000000	TAB29:	.WORD	10100,0,0,0
	003626	000000					
1698	003630	010100	000000	002000	TAB29A:	.WORD	10100,0,2000,0
	003636	000000					
1699							
1700	003640	000500	000000	000000	TAB30:	.WORD	500,0,0,0
	003646	000000					
1701	003650	100400	000000	000000	TAB31:	.WORD	100400,0,0,0
	003656	000000					
1702	003660	016000	000000	000000	TAB32:	.WORD	16000,0,0,0
	003666	000000					
1703	003670	011600	000000	000000	TAB33:	.WORD	11600,0,0,0
	003676	000000					
1704	003700	000640	000000	000000	TAB34:	.WORD	640,0,0,0
	003706	000000					
1705	003710	077600	000000	000000	TAB40:	.WORD	77600,0,0,0
	003716	000000					
1706	003720	100200	000000	000000	TAB41:	.WORD	100200,0,0,1
	003726	000001					
1707	003730	000340	000000	000000	TAB42:	.WORD	340,0,0,0
	003736	000000					
1708	003740	000077	177777	177777	TAB43:	.WORD	77,177777,177777,177776
	003746	177776					
1709	003750	000577	177777	177777	TAB45:	.WORD	577,-1,-1,-1
	003756	177777					
1710	003760	000577	177777	000000	TAB46:	.WORD	577,-1,0,0
	003766	000000					
1711	003770	173737	124242	052525	TAB47:	.WORD	173737,124242,052525,12346
	003776	012346					
1712	004000	000000	000000	052525	TAB47A:	.WORD	0,0,052525,12346
	004006	012346					
1713	004010	173737	124242	000000	TAB48:	.WORD	173737,124242,0,0
	004016	000000					
1714	004020	000600	000000	000000	TAB49:	.WORD	600,0,0,0
	004026	000000					
1715							
1716	004030	000000			DCOUNT	:	.WORD 0
1717	004032	000000			EXPDAT	:	.WORD 0
1718	004034	000000			RECDAT	:	.WORD 0
1719	004036	000000			PWDSEQ	:	.WORD 0
1720	004040	000000			ADLSB	:	.WORD 0
1721	004042	000000			RITEDA	:	.WORD 0
1722	004044	000000			NEWDAT	:	.WORD 0

GLOBAL DATA SECTION

1723 004046 000000
 1724 004050 000000
 1725 004052 000000
 1726 004054 000000
 1727 004056 000000
 1728
 1729
 1730 004060

CURDAT : .WORD 0
 FSTADD : .WORD 0
 LSTADD : .WORD 0
 CURADD : .WORD 0
 PARABT : .WORD 0 ;PARIY ABORT FLAG

004060 005737 004146
 004064 001034
 004066 032737 000040 000052
 004074 001430
 004076 012737 177777 004152
 004104 032737 000100 000052
 004112 001403
 004114 012737 177777 004154
 004122 104042
 004124 005060 000042
 004130 013737 000030 004146
 004136 013737 000032 004150
 004144 000404
 004146 000000
 004150 000000
 004152 000000
 004154 000000
 004156

START:
 ;; LCP/ORION ROUTINE TO SAVE EMULATOR AND PRIORITY
 EMTSAV: TST SAV30 ;; FIRST TIME THROUGH ?
 BNE VMKOR ;; BRANCH IF BEEN HERE ALREADY
 BIT #BIT5,@#52 ;; ARE WE IN UFD MODE ?
 BEQ VMKOR ;; LEAVE IF NOT
 MOV #1,UFDPLG ;; SET UFD FLAG
 BIT #BIT6,@#52 ;; ARE WE IN QUIET MODE ?
 BEQ 1\$;; BR IF NOT
 MOV #1,UQUIET ;; SET QUIET MODE
 1\$: EMT 42 ;; GET ADDRESS OF XXDP DCA TABLE
 CLR 42(R0) ;; CLR XXDP "DRSERR"
 MOV 30,SAV30 ;; SAVE EMULATOR ADDRESS
 MOV 32,SAV32 ;; SAVE EMULATOR PRIORITY LEVEL
 BR VMKOR ;; GET AROUND TAG AREA
 SAV30: .WORD 0 ;; PUT EMULATOR INFO HERE
 SAV32: .WORD 0 ;; PUT PRIORITY LOCATION HERE
 UFDPLG: .WORD 0 ;; USER FRIENDLY MODE FLAG
 UQUIET: .WORD 0 ;; UFD QUIET MODE FLAG
 VMKOR:

1731 004156

004156 012706 001100
 004162 005026
 004164 022706 001140
 004170 001374
 004172 012706 001100
 004176 012737 031766 000020
 004204 012737 000340 000022
 004212 012737 032270 000030
 004220 012737 000340 000032
 004226 012737 035014 000034
 004234 012737 000340 000036
 004242 012737 035076 000024
 004250 012737 000340 000026
 004256 013737 031700 031672
 004264 005037 001164
 004270 005037 001166
 004274 112737 000001 001115
 004302 012737 004302 001106
 004310 012737 004310 001110
 004316 013746 000004
 004322 012737 004356 000004
 004330 012737 177570 001140
 004336 012737 177570 001142

 1\$:
 .SBTTL INITIALIZE THE COMMON TAGS
 ;;CLEAR THE COMMON TAGS (\$CMTAG) AREA
 MOV #CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
 CLR (R6)+ ;;CLEAR MEMORY LOCATION
 CMP #SWR,R6 ;;DONE?
 BNE -6 ;;LOOP BACK IF NO
 MOV #STACK,SP ;;SETUP THE STACK POINTER
 ;;INITIALIZE A FEW VECTORS
 MOV #SCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
 MOV #340,@IOTVEC+2 ;;LEVEL 7
 MOV #ERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
 MOV #340,@EMTVEC+2 ;;LEVEL 7
 MOV #TRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
 MOV #340,@TRAPVEC+2;LEVEL 7
 MOV #PWRDN,@PWRVEC ;;POWER FAILURE VECTOR
 MOV #340,@PWRVEC+2 ;;LEVEL 7
 MOV \$ENDCT,\$EOPCT ;;SETUP END-OF-PROGRAM COUNTER
 CLR \$TIMES ;;INITIALIZE NUMBER OF ITERATIONS
 CLR \$ESCAPE ;;CLEAR THE ESCAPE ON ERROR ADDRESS
 MOVB #1,\$ERMAX ;;ALLOW ONE ERROR PER TEST
 MOV #.,\$LPADR ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
 MOV #.,\$LPERR ;;SETUP THE ERROR LOOP ADDRESS
 ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
 ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
 MOV @ERRVEC,-(SP) ;;SAVE ERROR VECTOR
 MOV #30000,\$ERRVEC ;;SET UP ERROR VECTOR
 MOV #DSWR,SWR ;;SETUP FOR A HARDWARE SWICH REGISTER
 MOV #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER

INITIALIZE THE COMMON TAGS

```

004344 022777 177777 174566      CMP      # 1,@SWR      ;; TRY TO REFERENCE HARDWARE SWR
004352 001012                      BNE      30002$      ;; BRANCH IF NO TIMEOUT TRAP OCCURRED
                                ;; AND THE HARDWARE SWR IS NOT = 1
004354 000403                      BR       30001$      ;; BRANCH IF NO TIMEOUT
004356 012716 004364 30000$:     MOV      #30001$, (SP) ;; SET UP FOR TRAP RETURN
004362 000002                      RTI
004364 012737 000176 001140 30001$: MOV      #SWREG, SWR    ;; POINT TO SOFTWARE SWR
004372 012737 000174 001142      MOV      #DISPREG, DISPLAY ;;
004400 012637 000004 30002$:     MOV      (SP)+, @#ERRVEC ;; RESTORE ERROR VECTOR
004404 005037 001206                      CLR      $PASS        ;; CLEAR PASS COUNT
004410 132737 000200 001221      BITB    #APTSIZE, $ENVM ;; TEST USER SIZE UNDER APT
004416 001403                      BEQ      30003$      ;; YES, USE NON-APT SWITCH
004420 012737 001222 001140      MOV      #SWREG, SWR   ;; NO, USE APT SWITCH REGISTER
004426                      30003$:
1732 004426 013737 004152 004154  MOV      UFDPLG, UQUIET ;; ABORT IN UFD ON ERROR
1733 004434 012737 030220 000004  MOV      #TOUT, @#ERRVEC ;; POINT TO TIMEOUT ROUTINE
1734 004442 012737 000340 000006  MOV      #340, @#ERRVEC+2 ;; AT PRIORITY 7
1735 004450 012737 027464 000114  MOV      #RAMPAR, @#114 ;; POINT PARITY ABORT
1736 004456 012737 000340 000116  MOV      #340, @#116   ;; AT PRIORITY 7
1737 004464 012737 030402 000250  MOV      #MMUTRP, @#250 ;; POINT MMU TRAP VECTOR
1738 004472 012737 000340 000252  MOV      #340, @#252   ;;
1739 004500 005037 177766                      CLR      @#177766     ;; CLEAR CPU ERROR REGISTER
1740 004504 032737 000100 000052  BIT      #BIT06, @#52 ;; IN UFD QUIET MODE ?
1741 004512 001164                      BNE      LOOP        ;; IF SO, SKIP PRINTOUT
1742 004514 104401 017750                      TYPE    ,SWTSEL      ;;
1743                      .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
004520 005737 000042                      TST     @#42         ;; ARE WE RUNNING UNDER XXDP/ACT?
004524 001012                      BNE     30004$       ;; BRANCH IF YES
004526 123727 001220 000001      CMPB   $ENV, #1     ;; ARE WE RUNNING UNDER APT?
004534 001406                      BEQ     30004$       ;; BRANCH IF YES
004536 023727 001140 000176      CMP    SWR, #SWREG  ;; SOFTWARE SWITCH REG SELECTED?
004544 001005                      BNE     30005$       ;; BRANCH IF NO
004546 104406                      GTSWR
004550 000403                      BR     30005$
004552 112737 000001 001134 30004$: MOVB   #1, $AUTOB    ;; SET AUTO-MODE INDICATOR
004560                      30005$:
1744 004560 000240                      NOP
1745 004562 123727 001220 000001  CMPB   $ENV, #1     ; debug aid
1746 004570 001020                      BNE     20$         ; running under APT?
1747 004572 013700 001224      MOV    $USWR, RO    ; default no-APT initialization
1748 004576 005700                      TST    RO          ; work copy pass calculation
1749 004600 001420                      BEQ     25$         ; if = 0 default value
1750                      ; setup default value
1751 004602 042700 000017      BIC    #17, RO     ; clear low order nibble
1752 004606 000241                      CLC
1753 004610 006000                      ROR    RO          ; assure no unknowns
1754 004612 006000                      ROR    RO          ; 4 rotates = divide
1755 004614 006000                      ROR    RO          ; by 16 (=pass time)
1756 004616 006000                      ROR    RO          ; this area subroutined
1757 004620 005700                      ROR    RO          ; with general purpose
1758 004622 001413                      TST    RO          ; divide, this test to
1759                      BEQ     30$         ; determine skip altogether
1760 004624 010037 003034      MOV    RO, CCHPAS  ; residue = no. of desired passes
1761 004630 000413                      BR     35$         ; continue on
1762 004632 012737 177777 003034 20$: MOV    # 1, CCHPAS  ; largest number no apt mode??
1763 004640 000407                      BR     35$
1764 004642 012737 000001 003034 25$: MOV    #1, CCHPAS  ; normal default = 1

```

GET VALUE FOR SOFTWARE SWITCH REGISTER

```

1765 004650 000403          BR      35$
1766 004652 012737 000000 003034 30$:  MOV    #0,CCHPAS      ; no cache tests included
1767 004660 000240          35$:  nop                ; debug a'd
1768
1769                          ;          BIT      #BIT07,@#52      ;UFD MODE?
1770                          ;          BNE     LOOP        ;IF YES, BRANCH
1771
1772
1773
1774
1775          :::::::::::::::::::::HALT ON ERROR JUMPER
1776
1777
1778
1779 004662 032737 000010 177750          BIT    #BIT03,MAIREG
1780 004670 001437          BEQ    99$
1781 004672 104401 004700          TYPE  ,30007$          ;;TYPE ASCIZ STRING
      004676 000433          BR     30006$          ;;GET OVER THE ASCIZ
      ;:30007$:          .ASCIZ <15><12>/TRAP ON HALT IS ENABLED, JUMPER IS NOT INSTALLED/<15><12>
      30006$:
1782
1783 004766 000434          BR     1000$
1784
1785 004770 99$:          TYPE  ,30009$          ;;TYPE ASCIZ STRING
      004774 104401 004776          BR     30008$          ;;GET OVER THE ASCIZ
      ;:30009$:          .ASCIZ <15><12>/TRAP ON HALT IS DISABLED, JUMPER IS INSTALLED/<15><12>
      30008$:
1786
1787 005060 004737 027546          1000$: JSR    PC,Q22SIZ      ;SIZE FOR Q22BE
1788
1789
1790
1791 005064 122737 000001 001220 LOOP:  CMPB  #APTENV,$ENV          ;ARE WE IN APT MODE?
1792 005072 001001          BNE   3000$          ;IF NOT: DO THIS TEST
1793 005074 000451          Br    TST1          ;;go to test 1
      ; under APT
1794
1795
1796 005076 005737 001206          3000$: TST    $PASS          ;FIRST PASS??
1797 005102 001401          BEQ   2000$          ;IF YES, DO IT
1798 005104 000445          Br    TST1          ;;go to test 1
1799 005106          2000$:
1800
1801 005106 005037 002362          clr   lopbak          ; clear loop back flag
1802 005112 104401 005120          TYPE ,30011$          ;;TYPE ASCIZ STRING
      005116 000431          BR    30010$          ;;GET OVER THE ASCIZ
      ;:30011$:          .ASCIZ <15><12><12>/IF LOOP BACK CONNECTOR INSTALLED TYPE:  Y/<15><12><12>
      30010$:
1803 005202 104410          rdchr          ;read the input character
1804 005204 022716 000131          cmp   #'Y,(sp)       ;is it yes ??
1805 005210 001003          bne   1$            ;no loop back connector
1806 005212 012737 000001 002362          mov   #1,lopbak      ;loop back installed, flag it.
1807 005220          1$:
1808
1809
1810
1811

```

TEST NATIVE REGISTER

1813
1814
1815
1816
1817
1818
1819
1820
1821
1822
1823
1824
1825
1826
1827
1828
1829
1830
1831
1832
1833
1834
1835
1836
1837

```

.SBTTL TEST - NATIVE REGISTER
;.....
; NATIVE REGISTER TEST
;
; This test checks out the native register's existence and it's
; various bits.
;
; BGNTST
; Set up timeout vector PC to TIMOUT
; Set up timeout vector PSW to 7
; Read the Native register
; Check out the bootstrap switch as read only
; Check out the module functional revision as read only
; Check out that the self test enable bit works
; Check out he indicators (i.e LEDES)
;
; ENDTST
;
; TIMOUT: Clean stack
; Error NATIVE register timed out
;
;-----

```

```

1838 005220 000004
1839 005222 032777 010000 173710
1840 005230 001424
005232 104401 005240
005236 000421
005302
1841 005302
1842 005302 013737 000004 003016
1843 005310 012737 005464 000004
1844 005316 012737 000340 000006
1845 005324 013700 177520
1846
1847
1848
1849
1850 005330 052737 160000 177520
1851 005336 013701 177520
1852 005342 020001
1853 005344 001402
1854 005346 104047
1855 005350 000447
1856
1857
1858
1859 005352 042737 000177 177520
1860 005360 032737 000177 177520
1861 005366 001403
1862 005370 104050
1863 005372 005037 172100
1864 005376 012701 000001
1865 005402 012702 000007

```

```

;.....
;NATIVE REGISTER TEST
;*****
TST1: SCOPE
; BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
; BEQ 1000$ ; THEN TYPE TEST TRACE
; TYPE ,30013$ ;;TYPE ASCIZ STRING
; BR 30012$ ;;GET OVER THE ASCIZ
;:30013$: .ASCIZ <15><12>/TEST 1 - NATIVE REGISTER TEST/
30012$:
1000$:
; MOV @44,SAVTIM ; SAVE UNEXPECTED TIMEOUT TRAP
; MOV #100$,@#4 ; SET UP TIMEOUT TRAP
; MOV #340,@#6
; MOV NATREG,R0 ; READ THE NATIVE REGISTER
;
; CHECK BITS 15-13 READ ONLY
;
; BIS #160000,NATREG ; TRY TO WRITE TO THE FUN REV BITS
; MOV NATREG,R1 ; READ THE NATIVE REGISTER AGAIN
; CMP R0,R1 ; IF IT CHANGED THEN
; BEQ 10$ ; ERROR IN NATIVE REGISTER
; ERROR +47
; BR 110$
;
; CHECK THE INDICATOR BITS
;
; 10$: BIC #177,NATREG ; CLEAR THE INDICATOR BITS
; BIT #177,NATREG ; IF THE BITS DIDN'T CLEAR
; BEQ 20$ ; THEN
; ERROR +50 ; ERROR IN THE NATIVE REGISTER
; CLR MER ; CLEAR MER AFTER REPORTING THE ERROR ;MC
; 20$: MOV #1,R1 ; START PATTERN IN FIRST BIT
; MOV #7,R2 ; SET LOOP COUNT TO 7

```


TEST NATIVE REGISTER

```

1866 005406          25$:          ; FOR ALL BITS THE ARE INDICATOR DO
1867 005406 050137 177520      BIS      R1,NATREG      ; : SET THE BIT
1868 005412 030137 177520      BIT      R1,NATREG      ; : CHECK THAT IT GOT SET
1869 005416 001002              BNE      30$              ; : IF NOT THEN
1870 005420 104050              ERROR    +50              ; : ERROR IN THE INDICATOR BITS
1871 005422 000422              BR      110$              ; :
1872 005424 006301      30$:      ASL      R1              ; : SHIFT PATTERN TO NEXT BIT
1873 005426 077211              SOB      R2,25$          ; : ENDFOR
1874 005430 042737 000177 177520 BIC      #177,NATREG    ; : CLEAR THE INDICATORS
1875
1876          ; CHECK THE BOOT SWITCH TO BE READ ONLY
1877
1878 005436 013700 177520      MOV      NATREG,R0      ; : READ THE NATIVE REGISTER
1879 005442 052737 007400 177520 BIS      #7400,NATREG    ; : TRY TO WRITE TO THE BOOT SELECT SWITCH
1880 005450 013701 177520      MOV      NATREG,R1      ; : READ IT AGAIN
1881 005454 020100              CMP      R1,R0          ; : IF THE BITS CHANGED THEN
1882 005456 001404              BEQ      110$          ; : ERROR IN THE NATIVE REGISTER
1883 005460 104051              ERROR    +51              ; :
1884 005462 000402              BR      110$          ; :
1885 005464 104052      100$:      ERROR    +52              ; : ERROR IN THE NATIVE REGISTER
1886 005466 022626              CMP      (SP)+,(SP)+    ; : ENDTST
1887
1888 005470 005037 172100      110$:   CLR      MER          ; : CLEAR MER AFTER REPORTING THE ERROR ;MC
1889 005474 013737 003016 000004 MOV      SAVTIM,#4      ; : RESTORE UNEXPECTED TIMEOUT ;MC
1890

```

TEST 16 BIT ROM CHECKSUM TEST

```

1892 .SBTTL TEST 16 BIT ROM CHECKSUM TEST
1893 ;ROM'S CHECKSUMS
1894 ;
1895 ;16 BIT ROM TEST
1896 ;:*****
1897 ;TST2: SCOPE
1898 005502 000004 BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
1899 005504 032777 010000 173426 BEQ 1000$ ; THEN TYPE TEST TRACE
1900 005512 001426 TYPE ,30015$ ;;TYPE ASCIZ STRING
005514 104401 005522 BR 30014$ ;;GET OVER THE ASCIZ
005520 000423 ;;30015$: .ASCIZ <15><12>/TEST 2 KDJ11 D ROM CHECKSUM TEST/
30014$:
1000$:
1901 005570 ;set trap catcher for 250 for aborts
1902
1903
1904
1905 005570 012737 006016 000250 mov #mmuerr,@#250
1906 005576 012737 000340 000252 mov #340,@#252
1907
1908 ;set up location 4
1909
1910 005604 013737 000004 003016 mov @#4,savt'm ;save unexpected timeout trap
1911 005612 012737 006050 000004 mov #timerr,@#4 ;set up the timeout trap
1912 005620 012737 000360 000006 mov #360,@#6
1913
1914 005626 004737 030676 jsr pc,setmmu
1915 005632 052737 000200 177520 bis #bit07,@#natreg ;enable rom in the native register
1916 005640 012737 174000 172350 mov #174000,@#kipar4 ;rom begins at pa= 17 400 000
1917 005646 005001 clr r1 ;init the temporary checksum
1918 005650 012700 100000 mov #begrom,r0 ;get address of beginning of rom
1919 ;begrom= 100 000
1920 005654 011002 mov (r0),r2 ;get the number of words covered
1921 ;by the checksum
1922 ;is the size zero ?
1923 005656 001014 bne 20$ ;no
1924
1925 005660 104101 error +101 ;yes error in rom size
1926 005662 005037 172100 clr mer
1927 005666 013737 003016 000004 mov savtim,@#4
1928 005674 005037 177572 clr @#sr0
1929 005700 042737 000200 177520 bic #bit07,@#natreg ;disable the rom
1930 005706 000475 Br TST3 ;;go to the next test
1931
1932
1933 005710 062001 20$: add (r0)+,r1 ;accumulate the checksum
1934 005712 020027 120000 cmp r0,#begrom+20000 ;finished the 4kw bank ?
1935 005716 103405 blo 30$ ;no
1936
1937 005720 062737 000200 172350 add #200,@#kipar4 ;yes bump address by 4 kw
1938 005726 012700 100000 mov #begrom,r0 ;get address of beginning of rom
1939
1940 005732 077212 30$: sob r2,20$ ;until all words are checked
1941
1942 005734 005701 tst r1 ;is the checksum correct ?
1943 005736 001414 beq 40$ ;yes, go to the next test
1944

```

TEST 16 BIT ROM CHECKSUM TEST

```

1945
1946 005740 104004          error +4          ;no, rom checksum error
1947 005742 005037 172100  clr      mer
1948 005746 013737 003016 000004  mov     savtim,@#4
1949 005754 005037 177572          clr     @#sr0
1950 005760 042737 000200 177520  bic     #bit07,@#natreg ;d'sable the rom
1951 005766 000445          Br      TST3          ;;go to the next test
1952
1953
1954
1955
1956 005770          40$:
1957
1958
1959 005770 005037 172100          clr     mer          ;end of test
1960 005774 013737 003016 000004  mov     savtim,@#4
1961 006002 005037 177572          clr     @#sr0
1962 006006 042737 000200 177520  bic     #bit07,@#natreg ;disable the rom
1963 006014 000432          Br      TST3          ;;go to the next test
1964
1965
1966
1967          ;handle trap to 250
1968
1969 006016 042737 000200 177520  mmuerr: bic     #bit07,@#natreg ;disable the rom
1970 006024 005037 177572          clr     @#sr0
1971 006030 104002          error   +2          ;mmu error
1972 006032 022626          cmp     (sp)+,(sp)+
1973 006034 005037 172100          clr     mer
1974 006040 013737 003016 000004  mov     savtim,@#4
1975 006046 000415          Br      TST3          ;;go to the next test
1976
1977
1978          ;handle trap to 4
1979
1980          timerr:
1981 006050 104041          error   +41
1982 006052 005037 172100          clr     mer
1983 006056 022626          cmp     (sp)+,(sp)+ ;clear stack
1984 006060 042737 000200 177520  bic     #bit07,@#natreg ;disable the rom
1985 006066 005037 177572          clr     @#sr0
1986 006072 013737 003016 000004  mov     savtim,@#4
1987 006100 000400          Br      TST3          ;;go to the next test
1988
1989
1990
1991

```

TEST KDJ11 DA DATA PATHS

1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
2011
2012
2013
2014
2015
2016
2017
2018
2019
2020
2021
2022

```

.SBTTL TEST - KDJ11 DA DATA PATHS
.....
: DATA PATH TEST
:
: This test checks out the data and address paths on the KDJ11-D
: Board. The patterns used will be:
:
:                                0
:                                177777
:                                177400
:                                170360
:                                007417
:                                052525
:                                125252
:
: BGNST
:   Set Timeout trap to TIMEOUT
:   Set timeout priority to 7
:   Read location 0
:   FOR pattern := first to last
:     Write pattern
:     Read pattern
:     IF Pattern read <> Pattern Written THEN
:       ERROR
:     ENDFOR
:   ENDTST
:
:-----

```

```

2023 006102 000004
2024 006104 032777 010000 173026
2025 006112 001425
2025 006114 104401 006122
2025 006120 000422
2026 006166
2026 006166 013737 000004 003016
2027 006166 013737 000004 003016
2028 006174 012737 006276 000004
2029 006202 012737 000340 000006
2030 006210 005737 000000
2031 006214 010701
2032 006216 062701 000106
2033 006222 012702 000007
2034 006226
2035 006226 011137 000000
2036 006232 023721 000000
2037 006236 001412
2038 006240 013737 000000 001126
2039 006246 016137 177776 001124
2040 006254 104044
2041 006256 005037 172100
2042 006262 000401
2043 006264 077220
2044
2045 006266 013737 003016 000004

```

```

:-----
:                                     ;DATA PATH TESTS
:*****
TST3: SCOPE
      BIT      #BIT12,@SWR           ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
      BEQ      1000$                 ; THEN TYPE TEST TRACE
      TYPE     ,30017$               ;;TYPE ASCIZ STRING
      BR       30016$                ;;GET OVER THE ASCIZ
:;30017$:
30016$: .ASCIZ <15><12>/TEST 3      KDJ11-D DATA PATH TEST/
1000$:
      MOV      @#4,SAV,IM             ; SAVE UNEXPECTED TIMEOUT TRAP ;MC
      MOV      #100$,@#4             ; SET UP THE TIMEOUT TRAP
      MOV      #340,@#6
      TST      @#0
      MOV      PC,R1
      ADD      #PATT-.,R1
      MOV      #7,R2
5$:
      MOV      (R1),@#0
      CMP      @#0,(R1)+
      BEQ      10$
      MOV      @#0,$BDDAT
      MOV      -2(R1),$GDDAT
      ERROR   +44
      CLR     MER
      BR      15$
10$: SOB      R2,5$
15$: MOV     SAVTIM,@#4
      ; RESTORE UNEXPECTED TIMEOUT TRAP
:
:   READ LOCATION 0
:   GET A POINTER TO THE PATTERNS
:   SET THE LOOP COUNT TO 7
:   FOR ALL OF THE PATTERNS
:   WRITE THE PATTERN TO 0
:   IF THE PATTERN IS NOT READ BACK
:   THEN
:     GET READ DATA
:     GET EXPECTED DATA
:     ERROR IN THE DATA PATHS
:   CLR MER AFTER ERROR REPORTING ;MC
:   END IF
: ENDFOR
: END TEST

```

N4

TEST KDJ11-DA DATA PATHS

```

2046 006274 000422          BR      TST4          ;;GO TO THE NEXT TEST
2047
2048
2049
2050          ; TIMEOUT ROUTINE
2051          ;
2052
2053 006276 012737 006306 000004 100$:  MOV      #102$,@#4
2054
2055 006304 104045          ERROR    +45
2056 006306 005037 172100 102$:  CLR      MER
2057
2058 006312 022626          CMP      (SP)+,(SP)+          ; CLEAN STACK
2059 006314 013737 003016 000004  MOV     SAVTIM,@#4          ; RESTORE UNEXPECTED TIMEOU* TRAP
2060 006322 000407          BR      TST4          ;;GO TO THE NEXT TEST
2061
2062 006324 000000          PATT:   .WORD    0
2063 006326 177777          .WORD    177777
2064 006330 177400          .WORD    177400
2065 006332 170360          .WORD    170360
2066 006334 007417          .WORD    007417
2067 006336 052525          .WORD    052525
2068 006340 125252          .WORD    125252
2069

```

TEST MEMORY ACCESSABILITY

2071
2072
2073
2074
2075
2076
2077
2078
2079
2080
2081
2082
2083
2084
2085
2086
2087
2088
2089
2090
2091
2092
2093

```

.SBTTL TEST MEMORY ACCESSABILITY
.....
:ACCESSIBILITY TEST
:
: This test will check the accessibility of each address of memory
: on the KDJ11 D Board. IF a memory address traps out then an error
: will be flagged. A side effect of this test should be that all memory
: is cleared
:
: BGNTST
:   Set timeout trap to TIMEOUT
:   Set timeout priority to 7
:   For MSB_Address := 200000 to 177777 D0
:     Contents of address := 0
:   Go to the next test
:
: TIMEOUT:
:   Error Address should not have timed out
: ENDTST

```

```

2094 006342 000004 010000 172566
2095 006344 032777 006362
2096 006352 001426
2096 006354 104401 006362
2096 006360 000423

2097 006430
2098 006430 013737 000004 003016
2099 006436 012737 006534 000004
2100 006444 012737 000340 000006
2101 006452 004737 030676
2102 006456 012701 040000
2103 006462 012737 001600 172344
2104 006470
2105 006470 142721 000377
2106 006474 020127 060000
2107 006500 103773
2108 006502 062737 000200 172344
2109 006510 012701 040000
2110 006514 023727 172344 020000
2111 006522 001362
2112
2113 006524 013737 003016 000004
2114 006532 000416
2115 006534
2116
2117
2118
2119
2120 006534 012737 006552 000004
2121 006542 005301
2122 006544 104046
2123 006546 005037 172100

```

```

:*****
TST4: SCOPE
      BIT      #BIT12,@SWR      ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
      BEQ      1000$            ; THEN TYPE TEST TRACE
      TYPE     ,30019$          ;;TYPE ASCIZ STRING
      BR       30018$           ;;GET OVER THE ASCIZ
      ;;30019$: .ASCIZ <15><12>/TEST 4 - MEMORY ACCESSIBILITY TEST/
      30018$:
      1000$:
      MOV      @#4,SAVTIM        ; SAVE TIMEOUT
      MOV      #100$,@#4        ; SET UP THE TIMEOUT TRAP
      MOV      #340,@#6         ;
      JSR      PC,SETMMU        ; GO SET UP THE MMU REGISTERS
      MOV      #40000,R1        ; WE WANT TO MAP THRU PAGE 2
      MOV      #1600,@#KIPAR2   ; SET UP KPAR 2
      1$:      ; FOR 160000 TO 2000000
      BICB    #377,(R1)+        ; : CLEAR OUT A BYTE
      CMP     R1,#60000         ; : IF WE HAVE PASSED THE PAGE BOUNDARY
      BLO     1$                ; : : THEN
      ADD     #200,@#KIPAR2     ; : : POINT KPAR2 TO A NEW PAGE
      MOV     #40000,R1         ; : : SET THE VIRTUAL ADDRESS TO PAGE 2
      CMP     @#KIPAR2,#20000   ; : : ENDF
      BNE     1$                ; :
      10$:   MOV      SAVTIM,@#4 ;
      BR      TST5              ;;GO TO THE NEXT TEST
      20$:
      ; TIMEOUT ROUTINE
      ;
      100$:  MOV      #102$,@#4
      DEC     R1                ;
      ERROR  +46                ; REPORT ERROR
      CLR     MER

```

C5

TEST MEMORY ACCESSABILITY

2124	006552	022626		1024:	CMP	(SP)+,(SP)+	:	CLEAN STACK
2125	006554	005037	177572		CLR	@SRO	:	
2126	006560	013737	003016	000004	MOV	SAVIM,@4	:	RESTORE TIMEOUT
2127	006566	000400			BR	TST5	::	GO TO THE NEXT TEST
2128								

TEST MEMORY ERROR REGISTER

```

2130 .SBTTL TEST MEMORY ERROR REGISTER
2131 .....
2132 ;MEMORY ERROR REGISTER
2133 ;
2134 ; This test will check the MEMORY ERROR REGISTER on the KDJ11 D
2135 ; Board.
2136 ;
2137 ; BGNTST
2138 ; Setup timeout VECTOR PC to TIMOUT
2139 ; Setup timeout VECTOR Priority to 7
2140 ; Setup Parity abort VECTOR to PARINT
2141 ; Setup Parity abort VECTOR Priority to 7
2142 ; Do a bus reset
2143 ; Read the Memory Error Register (17772100)
2144 ; Make sure that the register bits are in the right state
2145 ; Check all R/W bits
2146 ; ENDTST
2147 ;
2148 ;-----
2149 ;*****
2150 006570 000004 TST5: SCOPE
2151 006572 032777 010000 172340 BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
2152 006600 001427 BEQ 1000$ ; THEN TYPE TEST TRACE
2152 006602 104401 006610 TYPE ,30021$ ;;TYPE ASCIZ STRING
2152 006606 000424 BR 30020$ ;;GET OVER THE ASCIZ
2153 006660 ;;30021$: .ASCIZ <15><12>/TEST 5 - MEMORY ERROR REGISTER TEST/
2153 006660 30020$:
2154 006660 1000$:
2154 006660 005037 004056 clr parabt ;init parity abort.....mc
2155 006664 012737 000340 177776 mov #340,@#177776 ;set the psw to 7 .....mc
2156 006672 122737 000001 001220 CMPB #APTENV,$ENV ;ARE WE IN APT MODE?
2157 006700 001005 BNE 1001$ ;IF NOT: DO THIS TEST
2158 006702 005737 001206 TST $PASS ;FIRST PASS?
2159 006706 001402 BEQ 1001$ ; YES THEN PROCEED
2160 006710 000137 007452 JMP PAREND ; NO THEN GO TO THE NEXT TEST
2161
2162 006714 017737 000004 003016 1001$: MOV @#4,SAVTIM ; SAVE UNEXPECTED TIMEOUT TRAP
2163 006722 012737 007374 000004 MOV #100$,@#4 ; SET UP THE TIMEOUT TRAP
2164 006730 012737 000340 000006 MOV #340,@#6
2165 006736 013746 000114 MOV @#114,-(SP) ; SAVE VECTOR
2166 006742 013746 000116 MOV @#116,-(SP) ; SAVE PRIORITY
2167 006746 012737 007444 000114 MOV #PARINT,@#114
2168 006754 012737 000340 000116 MOV #340,@#116
2169 006762 005037 172100 CLR @#MER ; READ THE MER
2170 006766 013700 172100 MOV @#MER,R0
2171 ;
2172 ; CHECK OUT THE MER "ALWAYS READ AS 0" BITS
2173 ;
2174 006772 052737 030032 172100 BIS #30032,MER ; TRY TO WRITE THE STUCK AT 0 BITS
2175 007000 013701 172100 MOV MER,R1 ; READ THE MER
2176 007004 020001 CMP R0,R1 ; IF THE STUCK AT 0 BITS CHANGED
2177 007006 001403 BEQ 10$ ; : THEN
2178 007010 104055 ERROR +55 ; : ERROR IN THE MER
2179 007012 005037 172100 CLR MER ; ENDF
2180 ; CHECK THAT BITS 0,2,14,15 ARE CLEARED ON RESET
2181 ;
2182 007016 052737 140005 172100 10$: BIS #140005,MER ; SET ALL THE R/W BITS ON THE BOARD

```


E5

TEST MEMORY ERROR REGISTER

```

2183 007024 013700 172100      MOV      MER,R0      ; READ THE MER
2184 007030 042700 037772      BIC      #37772,R0   ; MASK OUT ALL THE UNWANTED BITS
2185 007034 020027 140005      CMP      R0,#140005  ; IF THE R/W BITS DID NOT GET SET THEN
2186 007040 001412              BEQ      15$         ; ERROR IN THE MER
2187 007042 005037 172100      CLR      MER        ;
2188 007046 010037 001126      MOV      R0,$BDDAT  ; GET RECIEVED DATA
2189 007052 012737 140005 001124  MOV      #140005,$GDDAT ; GET EXPECTED DATA
2190 007060 104056              ERROR    +56        ; REPORT ERROR
2191 007062 005037 172100      CLR      MER        ;
2192 007066 000005              15$: RESET          ; DO RESET
2193 007070 032737 140005 172100  BIT      #140005,MER ; MAKE SURE THE APPROPRIATE BITS GET CLEARED
2194 007076 001403              BEQ      20$         ; IF NOT THEN ERROR
2195 007100 104057              ERROR    +57        ;
2196 007102 005037 172100      CLR      MER        ;
2197
2198 ; CHECK OUT THE WRITE WRONG PARITY AND PARITY ERROR ENABLE BITS
2199
2200 007106              20$:
2201 007106 004737 030676      JSR      PC,SETMMU  ;
2202 007112 012737 017600 172344  MOV      #17600,@#KIPAR2 ; SET UP KPAR2
2203 007120 012701 057776      MOV      #57776,R1  ; POINT TO LAST VIRTUAL ADDRESS IN PAGE 2
2204 007124 042737 000001 172100  BIC      #BIT0,@#MER ; CLEAR PARITY ERROR ENABLE
2205 007132 052737 000004 172100  BIS      #BIT2,@#MER ; SET WRITE WRONG PARITY
2206
2207 ; WRITE WRONG PARITY TO LOCATION 1777776
2208
2209 007140 005011              CLR      (R1)        ; WRITE WRONG PARITY TO THAT LOCATION
2210 007142 042737 000004 172100  BIC      #BIT2,@#MER ; CLEAR WRITE WRONG PARITY
2211
2212 ; MAKE SURE NO PARITY ABORT OCCURS WHEN PARITY ERRORS ARE DISABLED
2213
2214 007150 005711              TST      (R1)        ; READ THE ADDRESS BACK
2215 007152 005737 003054      TST      MERTAG     ;
2216 007156 000240              NOP                ; SHOULD
2217 ; NOT GET
2218 ; A
2219 ; PARITY ABORT
2220
2221
2222
2223
2224
2225 007160 005737 004056      TST      PARABT     ; IF WE GOT A PARITY ABORT
2226 007164 001405              BEQ      25$         ; THEN
2227 007166 005037 004056      CLR      PARABT     ; CLEAR THE ABORT FLAG
2228 007172 104062              ERROR    +62        ; FLAG THE ERROR
2229 007174 005037 172100      CLR      @#MER      ; CLEAR MEMORY ERROR REGISTER ;MC
2230 ;
2231 ;
2232 ; MAKE SURE A PARITY ABORT OCCURS WHEN PARITY ERRORS ARE ENABLED
2233
2234 007200 052737 000001 172100  25$: BIS      #BIT0,@#MER ; NOW ENABLE PARITY ABORTS
2235 007206 005711              TST      (R1)        ; THIS HAS TO BE ONE WORD INTRUCTION
2236 ;
2237 ; READ THE ADDRESS BACK
2238 007210 005737 003054      TST      MERTAG     ;
2239 ; SHOULD

```

F5

SEQ 62

TEST - MEMORY ERROR REGISTER

```

2240                                     ; GET
2241                                     ; A
2242                                     ; PARITY ABORT
2243                                     ; SOON !!!
2244                                     ;
2245                                     ;
2246                                     ;
2247                                     ;
2248 007214 042737 000001 172100        BIC    #BIT0,@#MER    ; DISABLE PARITY ABORTS
2249 007222 022737 000001 004056        CMP    #1,PARABT    ; IF WE DIDN'T GET A PARITY ABORT
2250 007230 001405                       BEQ    30$          ; THEN
2251 007232 005037 004056                CLR    PARABT       ; CLEAR PARITY ABORT
2252 007236 104063                       ERROR  +63          ; ERROR DID NOT GET A PARITY ABORT
2253 007240 005037 172100                CLR    @#MER        ; CLEAR MEMORY ERROR REGISTER ;MC
2254
2255                                     ;
2256                                     ; CHECK OUT THE ADDRESS AND EXTENDED MER READ ENABLED
2257                                     ;
2258 007244 005037 004056                 30$:  CLR    PARABT    ; CLEAR THE ABORT FLAG
2259 007250 042737 100000 172100        BIC    #BIT15,@#MER ; CLEAR THE PARITY ERROR FLAG
2260 007256 013700 172100                MOV    @#MER,R0     ; GET THE MEMORY ERROR REGISTER
2261 007262 042700 170037                BIC    #170037,R0  ; CLEAR THE UNNEEDED BITS
2262 007266 020027 007740                CMP    R0,#7740    ; IF NOT CORRECT BITS SET THEN
2263 007272 001410                       BEQ    35$          ; ERROR IN ADDRESS 11 17
2264 007274 010037 001126                MOV    R0,$BDDAT   ; GET RECIEVED DATA
2265 007300 012737 007740 001124        MOV    #7740,$GDDAT ; GET EXPECTED DATA
2266 007306 104064                       ERROR  +64          ;
2267 007310 005037 172100                CLR    @#MER        ; CLEAR MEMORY ERROR REGISTER ;MC
2268
2269 007314 052737 040000 172100        35$:  BIS    #BIT14,@#MER ; SET ENABLE READ EXTENDED BIT
2270 007322 013702 172100                MOV    @#MER,R2    ; READ THE MER AGAIN
2271 007326 042702 177037                BIC    #177037,R2  ; CLEAR THE UNNEEDED BITS
2272 007332 020227 000040                CMP    R2,#40      ; IF NOT CORRECT BITS SET THEN
2273 007336 001410                       BEQ    40$          ; ERROR IN ADDR 18 21 OR READ EXT BIT
2274 007340 010237 001126                MOV    R2,$BDDAT   ; GET RECIEVED DATA
2275 007344 012737 000040 001124        MOV    #40,$GDDAT  ; GET EXPECTED DATA
2276 007352 104065                       ERROR  +65          ;
2277 007354 005037 172100                CLR    @#MER        ; ENDIF
2278                                     ; CLEAR MEMORY ERROR REGISTER ;MC
2279
2280 007360 005011                       40$:  CLR    (R1)      ;
2281 007362 005037 177572                CLR    @#SR0       ; DISABLE MMU
2282 007372 000407                       CLR    @#MER        ;
2283                                     ; BR    101$         ;
2284                                     ;
2285                                     ; TIMEOUT ROUTINE
2286                                     ;
2287 007374 104060                       100$: ERROR  +60    ; REPORT ERROR
2288 007376 005037 172100                CLR    @#MER        ; CLEAR MEMORY ERROR REGISTER ;MC
2289
2290                                     ;
2291 007402 022626                       CMP    (SP)+,(SP)+ ; CLEAN STACK
2292 007404 005011                       CLR    (R1)         ;
2293 007406 005037 177572                CLR    @#SR0       ; DISABLE MMU
2294 007412 012737 007424 000004        101$: MOV    #102$,@#4 ;
2295 007420 005037 172100                CLR    @#MER        ; CLEAR MEMORY ERROR REGISTER ;MC
2296 007424                                     102$:

```

G5

TEST MEMORY ERROR REGISTER

```

2297 007424 012637 000116          MOV    (SP)+, @#116          ;RESTORE PRIORITY
2298 007430 012637 000114          MOV    (SP)+, @#114          ;RESTORE VESTOR
2299 007434 013737 003016 000004  MOV    SAVTIM, @#4          ;RESTORE UNEXPECTED TIMEOUT
2300 007442 000403                    BR     TST6                  ;;GO TO THE NEXT TEST
2301                                     ;
2302                                     ; PARITY ABORT ROUTINE
2303                                     ;
2304                                     ;
2305 007444                    PARINT:
2306 007444 005237 004056          INC    PARABT                ; FLAG THAT WE CAME HERE
2307 007450 000002                    RTI                          ; RETURN
2308 007452                    PAREND:

```

TEST - MEMORY ERROR REGISTER

2310
2311
2312
2313
2314
2315
2316
2317
2318
2319
2320
2321
2322
2323
2324
2325
2326
2327
2328
2329
2330
2331
2332
2333
2334
2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354
2355

.SBTTL TEST DATA SHORTS AND STUCK AT BITS
:*****
:DATA SHORTS AND STUCK AT BITS TEST

: This test will check the DATA Rams for Data shorts and stuck at bits.
: Testing occurs as below:

1. A memory location is checked to be set to 0
2. IF NOT 0 error
3. The location is complemented
4. IF contents NOT -1 error
5. This is repeated for all addresses
6. Steps 1 5 are done from location 177777 to 0

: Note: The KDJ11 DA board uses 256k X 1 chips, This allows us to
: use only 2 patterns per RAM. IF one bits is shorted to
: another we will detect this when we read for the initial state.
: If it has changed from the expected state then chances are that
: the bits are shorted together.

: A side effect of this test should be that memory is cleared.

```

: BGNTST
:   ENABLE PARITY
:   FOR MSB Address := 0 to 177776 DO BY 2
:     Clear Address
:     IF Contents <> 0 THEN
:       ERROR in memory
:     Complement the Contents of Address
:     IF contents <> -1 THEN
:       ERROR in memory
:   ENDFOR
:   FOR MSB Address := 177776 DOWNT0 0 DO BY 2
:     IF contents <> -1 THEN
:       ERROR in memory
:     Complement the Contents of Address
:     IF contents <> 0 THEN
:       ERROR in memory
:   ENDFOR
: ENDTST

```

```

2356 007452 000004
2356 007454 032777 010000 171456
2357 007462 001433
2358 007464 104401 007472
      007470 000430

      007552
2359 007552
2360 007552 005037 177572
2361 007556 013737 000004 003016
2362 007564 012737 010130 000004

```

```

:*****
TST6: SCOPE
      BIT      #BIT12,@SWR      ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
      BEQ      1000$            ; THEN TYPE TEST TRACE
      TYPE     ,30023$          ;;TYPE ASCIZ STRING
      BR       30022$           ;;GET OVER THE ASCIZ
;;30023$: .ASCIZ <15><12>/TEST 6 DATA SHORTS AND STUCK AT BITS TEST/
30022$:
1000$:
      CLR      @#SRO            ;
      MOV      @#4,SAVTIM       ;STORE UNEXPECTED TIMEOUT
      MOV      #100$,@#4       ; SET UP THE TIMEOUT TRAP

```

TEST - DATA SHORTS AND STUCK AT BITS

```

2363 007572 012737 000340 000006      MOV      #340,@#6      ;
2364 007600 004737 030676      JSR      PC,SETMMU    ; GO SET UP THE MMU REGISTERS
2365 007604 012701 040000      MOV      #40000,R1    ; WE WANT TO MAP THRU PAGE 2
2366 007610 012737 001600 172344      MOV      #1600,@#KIPAR2 ; SET UP KPAR 2
2367 007616 013746 000114      MOV      @#114,-(SP)  ; SAVE VECTOR ;MC
2368 007622 013746 000116      MOV      @#116,-(SP)  ; SAVE PRIORITY ;MC
2369 007626 012737 010116 000114      MOV      #50,@#114    ; set up trap ;mc
2370 007634 012737 000340 000116      MOV      #340,@#116   ; " " " ;mc
2371 007642 052737 000001 172100      BIS      #BIT0,@#MER  ; ENABLE PARITY ABORTS ;mc
2372
2373 007650 10$:      MOV      (R1),#BDDAT  ; FOR ADDRESS 160000 TO 200000 DO
2374 007650 011137 001126      BEQ      15$          ; : READ MEMORY
2375 007654 001406      CLR      $GDDAT      ; : IF IT IS NOT ZERO THEN
2376 007656 005037 001124      MOV      R1,#BDADR   ; : GET THE EXPECTED DATA
2377 007662 010137 001122      ERROR   +61          ; : GET THE VIRTUAL ADDRESS
2378 007666 104061      BR       101$        ; : ERROR IN THE MEMORY
2379 007670 000531      ;                   ; RESTORE AND TO THE NEXT TEST
2380
2381 007672 005111 15$:      COM      (R1)        ; : COMPLEMENT MEMORY
2382 007674 011137 001126      MOV      (R1),#BDDAT ; : SAVE THE RECIEVED DATA
2383 007700 022127 177777      CMP      (R1)+,#177777 ; : IF ITS NOT = -1 THEN
2384 007704 001412      BEQ      20$          ; : ERROR IN THE MEMORY
2385 007706 012737 177777 001124      MOV      #-1,$GDDAT  ; : GET EXPECTED DATA
2386 007714 010137 001122      MOV      R1,#BDADR   ; : GET THE VIRTUAL ADDRESS
2387 007720 162737 000002 001122      SUB      #2,$BDADR   ; :
2388 007726 104061      ERROR   +61          ; :
2389 007730 000511      BR       101$        ; RESTORE AND TO THE NEXT TEST
2390
2391 007732 020127 060000 20$:      CMP      R1,#60000   ; : IF WE HAVE PASSED THE PAGE BOUNDARY
2392 007736 103744      BLO      10$          ; : : THEN
2393 007740 062737 000200 172344      ADD      #200,@#KIPAR2 ; : : POINT KPAR2 TO A NEW PAGE
2394 007746 012701 040000      MOV      #40000,R1   ; : : SET THE VIRTUAL ADDRESS TO PAGE 2
2395 007752 023727 172344 020000      CMP      @#KIPAR2,#20000 ; : : ENDFIF
2396 007760 001401      BEQ      25$          ; : ENDFOR
2397 007762 000732      BR       10$          ;
2398 007764 25$:      jsr      pc,delay    ; MOV      #200,R5 ; THIS LOOP
2399 007764 004737 030610
IS IN HERE TO TEST THE DATA RETENTION OF MEMORY.
2400
2401
2402
2403 007770 012701 060000      MOV      #60000,R1   ; GET A POINTER TO THE TOP OF A PAGE
2404 007774 162737 000200 172344      SUB      #200,@#KIPAR2 ; SET KIPAR TO POINT TO THE TOP PAGE
2405 010002 30$:      MOV      -(R1),#BDDAT ; FOR LAST ADDRESS TO FIRST BY 2
2406 010002 014137 001126      CMP      #BDDAT,#177777 ; : SAVE THE DATA
2407 010006 023727 001126 177777      BEQ      35$          ; : READ BACK MEMORY MAKE SURE IT IS 1
2408 010014 001407      MOV      #177777,$GDDAT ; : IF NOT = -1 THEN
2409 010016 012737 177777 001124      MOV      R1,#BDADR   ; : GET THE EXPECTED DATA
2410 010024 010137 001122      ERROR   +61          ; : GET THE VIRTUAL ADDRESS
2411 010030 104061      BR       101$        ; : ERROR IN MEMORY
2412 010032 000450      ;                   ; RESTORE AND TO THE NEXT TEST
2413
2414 010034 005111 35$:      COM      (R1)        ; : COMPLEMENT MEMORY
2415 010036 011137 001126      MOV      (R1),#BDDAT ; : IF NOT = 0 THEN
2416 010042 001406      BEQ      40$          ; : ERROR IN MEMORY
2417 010044 005037 001124      CLR      $GDDAT      ; : GET EXPECTED DATA
2418 010050 010137 001122      MOV      R1,#BDADR   ; : GET THE VIRTUAL ADDRESS
2419 010054 104061      ERROR   +61          ;

```

J5

TEST - DATA SHORTS AND STUCK AT BITS

```

2420 010056 000436          BR      101$          ; RESTORE AND TO THE NEXT TEST
2421
2422 010060 020127 040000    40$:  CMP      R1,#40000      ; : IF VIRTUAL ADDRESS ABOUT TO ENTER NEXT PAGE
2423 010064 101346          BHI      30$          ; : : THEN
2424 010066 162737 000200 172344  SUB      #200,@#KIPAR2 ; : : ADJUST THE PAR
2425 010074 012701 060000          MOV      #60000,R1    ; : : RESET THE VIRTUAL ADDRESS TO TOP OF PAGE
2426 010100 023727 172344 001400  CMP      @#KIPAR2,#1400 ; : : ENDFOR
2427 010106 001335          BNE      30$          ; : ENDFOR
2428 010110 005037 172100          CLR      MER          ; CLEAR THE MER
2429 010114 000417          BR      101$          ; RESTORE AND TO THE NEXT TEST
2430
2431          ; PARITY ABORT ROUTINE
2432          ;
2433
2434 010116 104072          50$:  ERROR   +72          ; REPORT ERROR
2435 010120 005037 172100          CLR      MER          ; INIT THE MER
2436 010124 022626          CMP      (SP)+,(SP)+ ; CLEAN STACK
2437
2438          ; VECTOR SAVE AND RESTORE
2439          ; UNEXPECTED PARITY ABORT ERROR ROUTINE
2440 010126 000412          BR      101$
2441
2442          ;
2443          ; TIMEOUT ROUTINE
2444          ;
2445
2446 010130 012737 010160 000004 100$:  MOV      #102$,@#4
2447 010136 005301          DEC      R1          ;
2448 010140 104046          ERROR   +46          ; REPORT ERROR
2449 010142 022626          CMP      (SP)+,(SP)+ ; CLEAN STACK
2450 010144 005037 172100          CLR      MER          ;
2451 010150 005037 177572          CLR      @#SRO       ;
2452
2453 010154 005037 172100          101$:  CLR      MER          ; INIT THE MER
2454
2455          102$:
2456 010160          MOV      (SP)+,@#116 ; RESTORE PRIORITY
2457 010164 012637 000116          MOV      (SP)+,@#114 ; RESTORE VECTOR
2458 010170 013737 003016 000004  MOV      SAVTIM,@#4  ; RESTORE UNEXPECTED TIMEOUT
2459 010176 000400          BR      TST7         ;;GO TO THE NEXT TEST
2460
2461

```

TEST QUICK VERIFY DATA AND ADRESSING TEST

2463
2464
2465
2466
2467
2468
2469
2470
2471
2472
2473
2474
2475
2476
2477
2478
2479
2480
2481
2482
2483
2484
2485
2486
2487
2488
2489
2490
2491
2492

```
.SBTTL TEST - QUICK VERIFY DATA AND ADRESSING TEST
:*****
:UNIQUE ADDRESSING TEST
:
: This test will check the data and addressing of the memories.
: It uses the KNAIZUK HARTMANN QUICK VERIFY TRAM TEST ALGORITHM.
: This algorithm will test memory for all stuck at faults in the
: data and addressing
:
: Memory is split up into sections of 3.
: I.E. 0,1,2 3,4,5 6,7,10 ...
:
: Testing works as follows.
:
: 1. Write a 0 into the 2nd and 3rd address of all groups
: write a 1 into the 1st address of all groups
: 2. make sure that the 2nd address of all groups contain 0
: 3. Write a 1 into the 2nd address of all groups
: 4. Make sure that the 3rd address of all groups contain 0
: 5. Make sure that the 1st and 2nd address of all groups
: contain -1
: 6. Write a 0 into the 1st address of all groups
: 7. Make sure that the 1st address of all groups contain 0
: 8. Write a 1 into the 3rd address of all groups
: 9. Make sure that the 3rd address of all groups contain 1
:
:-----
```

2493 010200 000004
2494 010202 032777 010000 170730
2495 010210 001434
010212 104401 010220
010216 000431

2496 010302
2497 010302 004737 030676
2498 010306 013737 000004 003016
2499 010314 013737 010476 000004
2500 010322 004737 031012
2501 010326 012704 000001
2502 010332 005005
2503 010334 005003
2504 010336 004737 031176
2505 010342 012704 000001
2506 010346 012705 177777
2507 010352 012703 000001
2508 010356 004737 031176
2509 010362 012704 000002
2510 010366 005005
2511 010370 005003
2512 010372 004737 031176
2513 010376 004737 031350
2514 010402 005004
2515 010404 005005

```
*****
TST: SCOPE
BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
BEQ 1000$ ; THEN TYPE TEST TRACE
TYPE 30025$ ;:TYPE ASCIZ STRING
BR 30024$ ;:GET OVER THE ASCIZ
;:30025$: .ASCIZ <15><12>/TEST 7 - QUICK VERIFY DATA AND ADDRESSING TEST/
30024$:
1000$:
JSR PC,SETMMU ; SET UP THE MMU REGISTERS
MOV @#4,SAVTIM ; STORE UNEXPECTED TIMEOUT
MOV 100$,@#4 ; TIMEOUT
CALL INIMEM ; INITIALIZE MEMORY
MOV #1,R4 ; SET ADDRESS INDEX TO 1
CLR R5 ; SET EXPECTED PATTERN TO 0
CLR R3 ; SET OPERATION TO READ
CALL RWTMEM ; GO READ AND CHECK MEMORY
MOV #1,R4 ; SET ADDRESS INDEX TO 1
MOV #-1,R5 ; SET WRITE PATTERN TO -1
MOV #1,R3 ; SET OPERATION TO WRITE
CALL RWTMEM ; GO WRITE THE MEMORY
MOV #2,R4 ; SET ADDRESS INDEX TO 2
CLR R5 ; SET EXPECTED PATTERN TO 0
CLR R3 ; SET OPERATION TO READ
CALL RWTMEM ; GO READ AND CHECK MEMORY
CALL RRMEM ; READ AND CHECK I AND I+1 INDEXES
CLR R4 ; SET ADDRESS INDEX TO 0
CLR R5 ; SET PATTERN TO 0
```

L5

TEST - QUICK VERIFY DATA AND ADRESSING TEST

```

2516 010406 012703 000001      MOV    #1,R3      ; SET OPERATION TO WRITE
2517 010412 004737 031176      CALL   RWTMEM     ; GO WRITE THE MEMORY
2518 010416 005004              CLR    R4         ; SET ADDRESS INDEX TO 0
2519 010420 005005              CLR    R5         ; SET EXPECTED PATTERN TO 0
2520 010422 005003              CLR    R3         ; SET OPERATION TO READ
2521 010424 004737 031176      CALL   RWTMEM     ; GO READ AND CHECK MEMORY
2522 010430 012704 000002      MOV    #2,R4     ; SET ADDRESS INDEX TO 2
2523 010434 012705 177777      MOV    #1,R5     ; SET PATTERN TO 1
2524 010440 012703 000001      MOV    #1,R3     ; SET OPERATION TO WRITE
2525 010444 004737 031176      CALL   RWTMEM     ; GO WRITE THE MEMORY
2526 010450 012704 000002      MOV    #2,R4     ; SET ADDRESS INDEX TO 2
2527 010454 012705 177777      MOV    #-1,R5    ; SET EXPECTED PATTERN TO 1
2528 010460 005003              CLR    R3         ; SET OPERATION TO READ
2529 010462 004737 031176      CALL   RWTMEM     ; GO READ AND CHECK MEMORY
2530 010466 013737 003016 000004  MOV    SAVTIM,@#4 ; RESTORE UNEXPECTED TIMEOUT
2531 010474 000416              BR     TST10     ;;GO TO THE NEXT TEST
2532
2533
2534 ; TIMEOUT ROUTINE
2535 ;
2536
2537 010476 012737 010522 000004 100$:  mov    #102$,@#4
2538 010504 005301              DEC    R1         ;
2539 010506 104046              ERROR  +46        ; REPORT ERROR
2540 010510 005037 172100              clr    mer
2541 010514 022626              CMP    (SP)+,(SP)+ ; CLEAN STACK
2542 010516 005037 177572              CLR    @#SR0     ;
2543 010522 013737 003016 000004 102$:  MOV    SAVTIM,@#4 ; RESTORE UNEXPECTED TIMEOUT
2544 010530 001400              BEQ   TST10     ;;GO TO THE NEXT TEST
2545

```


TEST - QUICK VERIFY DATA AND ADDRESSING TEST

2547
2548
2549
2550
2551
2552
2553
2554
2555
2556
2557
2558
2559
2560
2561
2562
2563
2564
2565
2566
2567
2568
2569
2570
2571
2572
2573
2574
2575
2576
2577
2578
2579
2580
2581

```

.SBTTL TEST  CHECK PARITY DETECT LOGIC AND RAMS
.....
:CHECK PARITY DETECT LOGIC
:
: This test will check out the parity detection logic on the KDJ11-DA
: board. We will write wrong parity to all locations and then we will
: Expect the a parity trap on a read.
:
: BGNTST
:   Set up Parity Vector (114) To PARINT
:   Enable Write Bad parity (Set bit 2 in MER (17772100))
:   Clear all of memory
:   Read first location written to
:   IF Parity Abort Occurs THEN
:     Error: There should be no Parity Aborts when disabled
:   Enable Parity Error (Set Bit 0 in MER (17772100))
:   FOR First Address to Last address DO
:     BEGIN
:       READ Address
:       NOP
:       IF No Interrupt THEN
:         Error Didn't Detect Bad parity
:       ELSE
:         Clear the interrupt flag
:         Read the MER and make sure the obtained address is the correct one
:     ENDFOR
:   PARINT:
:     Flag that an interrupt occurred
:   RTI
: ENDTST

```

```

2582 010532 000004
2583 010534 122737 0000C 001220
2584 010542 001005
2585 010544 005737 0012C
2586 010550 001402
2587 010552 000137 011552
2588 010556 032777 000200 170354 2000$
2589 010564 001402
2590 010566 000137 011552
2591 010572 032777 010000 170340 100$
2592 010600 001426
2593 010602 104401 010610
2594 010606 000423
2595 010656
2596 010656 005037 172100
2597 010662 005037 004056
2598 010666 013746 000114
2599 010672 013746 000116
2599 010676 012737 007444 000114
2599 010704 012737 000340 000116

```

```

.....
TST10: SCOPE
CMPB  #APTENV,$ENV ;ARE WE IN APT MODE?
BNE 2000$ ;IF NOT: DO THIS TEST
TST $PASS ;FIRST PASS??
BEQ 2000$ ;IF YES, DO IT
JMP NXTST ;ELSE GO TO THE NEXT TEST
BIT #BIT7,@SWR ; IF BIT7 SET IN SWR THEN SKIP THIS TEST
BEQ 100$
JMP NXTST
BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
BEQ 1000$ ; THEN TYPE TEST TRACE
TYPE ,30027$ ;:TYPE ASCIZ STRING
BR 30026$ ;:GET OVER THE ASCIZ
;:30027$: .ASCIZ <15><12>/TEST 10 CHECK PARITY DETECT TEST/
30026$:
1000$: CLR @MER ; INIT THE MEMORY ERROR REG.
CLR PARABT
MOV @#114, (SP) ; SAVE VECTORE
MOV @#116, (SP) ; SAVE PRIORITY
MOV #PARINT,@#114
MOV #340,@#116

```

TEST CHECK PARITY DETECT LOGIC AND RAMS

```

2600
2601
2602
2603 010712 004737 030676
2604 010716 105037 003014
2605 010722 012701 040000
2606 010726 012737 001600 172344
2607 010734 012703 000002
2608
2609
2610 010740 1$:
2611
2612 010740 012737 000004 172100
2613 010746 113721 003014
2614 010752 005037 172100
2615
2616 010756 020127 060000
2617 010762 103766
2618 010764 062737 000200 172344
2619 010772 012701 040000
2620 010776 023727 172344 020000
2621 011004 001355
2622
2623
2624
2625 011006 20$:
2626 011006 012737 001600 172344
2627 011014 012701 040000
2628 011020 22$:
2629 011020 042737 000004 172100
2630 011026 052737 000001 172100
2631 011034 105711
2632 011036 005737 003054
2633 011042 000240
2634
2635
2636
2637
2638
2639
2640
2641
2642 011044 023727 004056 000001
2643 011052 001410
2644 011054 005037 004056
2645 011060 005037 172100
2646 011064 104063
2647 011066 005037 172100
2648 011072 000465
2649
2650
2651
2652 011074 25$:
2653 011074 032737 100001 172100
2654 011102 001003
2655 011104 104073
2656 011106 005037 172100

```

```

; CLEAR ALL OF MEMORY WITH WRITE WRONG PARITY SET
;
; JSR PC,SETMMU ; GO SET UP THE MMU REGISTERS
; CLR PARPAT ; INITIALIZE FIRST PATTERN TO ZERO
; MOV #40000,R1 ; WE WANT TO MAP THRU PAGE 2
; MOV #1600,@#KIPAR2 ; SET UP KPAR 2
; MOV #2,R3 ; LOOP FOR ODD PARITY AND EVEN PARITY
;
; BGND0
; : ENABLE WRITE WRONG PARITY ;MC
; : CLEAR OUT A BYTE
; : DISABLE WRITE WRONG PARITY ;MC
; : FOR 160000 TO 2000000
; : : IF WE HAVE PASSED THE PAGE BOUNDARY
; : : : THEN
; : : : POINT KPAR2 TO A NEW PAGE
; : : : SET THE VIRTUAL ADDRESS TO PAGE 2
; : : : ENDIF
; : : : ENDFOR
;
; LET'S NOW MAKE SURE THAT THE ABORTS OCCUR
;
; 20$:
; MOV #1600,@#KIPAR2 ; : SET UP PAR 2
; MOV #40000,R1 ; : SET VIRTUAL ADDRESS TO PAGE #2
;
; 22$:
; BIC #BIT2,@#MER ; : : DISABLE WRITE WRONG PARITY
; BIS #BIT0,@#MER ; : : ENABLE PARITY ERRORS
; TSTB (R1) ; : : READ THE BYTE
; TST MERTAG ; : :
; NOP ; : :
; : : : PARITY
; : : : ABORT
; : : : TRAP
; : : : SHOULD
; : : : HAPPEN
; : : : SOON !!!
;
; : : : IF A PARITY ABORT DIDN'T OCCUR
; : : : THEN
; : : : (EITHER 0 OR >1 ABORTS HAVE OCCURED)
; : : : CLEAR OUT THE MER
; : : : ERROR DID NOT GET A PARITY ABORT
; : : : ELSE
;
; CHECK OUT THE ADDRESS AND EXTENDED MER READ ENABLED
;
; 25$:
; BIT #100001,@#MER ; : : : ARE BITS 0 AND 15 ARE SET ;MC
; BNE 30$ ; : : : YES THEY ARE ;MC
; ERROR +73 ; : : : NO THEY'RE NOT SET ;MC
; clr mer

```

Bc

TEST CHECK PARITY DETECT LOGIC AND RAMS

```

2657 011112 042737 100000 172100 30$: BIC #BIT15,@MER ; : : : CLEAR THE PARITY ERROR FLAG
2658 011120 010100 MOV R1,R0 ; : : : GET THE VIRTUAL ADDRESS
2659 011122 042700 160000 BIC #160000,R0 ; : : : MASK OUT THE PAGE BITS
2660 011126 072027 177772 ASH # 6,R0 ; : : : SHIFT OUT BITS 0-5
2661 011132 063700 172344 ADD @KIPAR2,R0 ; : : : CREATE BITS 6-21
2662 011136 010004 MOV R0,R4 ; : : : SAVE A COPY FOR LATER
2663 011140 042704 170037 BIC #170037,R4 ; : : : CLEAR OUT UNNEEDED BITS
2664 011144 013705 172100 MOV @MER,R5 ; : : : READ THE MER
2665 011150 042705 170037 BIC #170037,R5 ; : : : CLEAR THE UNNEEDED BITS
2666 011154 020405 CMP R4,R5 ; : : : IF NOT CORRECT BITS SET THEN
2667 011156 001407 BEQ 35$ ; : : : ERROR IN ADDRESS 11-17
2668 011160 010437 001124 MOV R4,+GDDAT ; : : : GET EXPECTED DATA
2669 011164 010537 001126 MOV R5,$BDUAT ; : : : GET RECIEVED DATA
2670 011170 104064 ERROR +64 ; : : :
2671 011172 005037 172100 clr mer ; : : :
2672 011176 052737 040000 172100 35$: BIS #BIT14,@MER ; : : : SET ENABLE READ EXTENDED BIT
2673 011204 013705 172100 MOV @MER,R5 ; : : : READ THE MER AGAIN
2674 011210 042705 177037 BIC #177037,R5 ; : : : CLEAR THE UNNEEDED BITS
2675 011214 072027 177771 ASH #-7,R0 ; : : : GET BITS 18 21 IN RIGHT PLACE
2676 011220 042700 177037 BIC #177037,R0 ; : : : MASK OUT UNNEEDED BITS
2677 011224 020005 CMP R0,R5 ; : : : IF NOT CORRECT BITS SET THEN
2678 011226 001407 BEQ 40$ ; : : : ERROR IN ADDR 18-21 OR READ EXT BIT
2679 011230 010537 001126 MOV R5,$BDDAT ; : : : GET RECIEVED DATA
2680 011234 010037 001124 MOV R0,$GDDAT ; : : : GET EXPECTED DATA
2681 011240 104065 ERROR +65 ; : : : ENDF
2682 011242 005037 172100 clr mer ; : : :
2683 011246 005037 172100 40$: CLR @MER ; : : : RESET MER
2684 011252 005037 004056 CLR PARABT ; : : : CLEAR THE PARITY ABORT TRAP
2685 011256 005201 INC R1 ; : : : BUMP UP THE ADDRESS
2686 011260 020127 060000 CMP R1,#60000 ; : : : IF WE HAVE PASSED THE PAGE BOUNDARY
2687 011264 103655 BLO 22$ ; : : : THEN
2688 011266 062737 000200 172344 ADD #200,@KIPAR2 ; : : : POINT KPAR2 TO A NEW PAGE
2689 011274 012701 040000 MOV #40000,R1 ; : : : SET THE VIRTUAL ADDRESS TO PAGE 2
2690 011300 023727 172344 020000 CMP @KIPAR2,#20000 ; : : : ENDF
2691 011306 001244 BNE 22$ ; : : : DOUNTIL WE HAVE READ ALL ADDRESSES
2692 011310 112737 000001 003014 MOVB #1,PARPAT ; : : : THIS TIME WE WANT AN ODD # OF 1 S
2693 011316 005303 DEC R3 ; : : : DECREMENT LOOP COUNT
2694 011320 001412 BEQ 50$ ; : : :
2695 011322 012701 040000 MOV #40000,R1 ; : : : WE WANT TO MAP THRU PAGE 2
2696 011326 012737 001600 172344 MOV #1600,@KIPAR2 ; : : : SET UP KPAR 2
2697 011334 052737 000004 172100 BIS #BIT2,MER ; : : : ENABLE WRITE WRONG PARITY
2698 011342 000137 010740 JMP 1$ ; : : : DOUNTIL WE DONE IT FOR EVEN AND ODD NUMBER
OF 1'S
2699 011346 50$: ; : :
2700 ; : : : CLEAR ALL OF MEMORY WITH WRITE W JNG PARITY CLEAR
2701 ; : : :
2702 ; : : :
2703 011346 004737 030676 JSR PC,SETMMU ; : : : GO SET UP THE MMU REGISTERS
2704 011352 012701 040000 MOV #40000,R1 ; : : : WE WANT TO MAP THRU PAGE 2
2705 011356 012737 001600 172344 MOV #1600,@KIPAR2 ; : : : SET UP KPAR 2
2706 011364 60$: ; : : FOR 160000 TO 2000000
2707 011364 105021 CLRB (R1)+ ; : : : CLEAR OUT A BYTE
2708 011366 020127 060000 CMP R1,#60000 ; : : : IF WE HAVE PASSED THE PAGE BOUNDARY
2709 011372 103774 BLO 60$ ; : : : THEN
2710 011374 062737 000200 172344 ADD #200,@KIPAR2 ; : : : POINT KPAR2 TO A NEW PAGE
2711 011402 012701 040000 MOV #40000,R1 ; : : : SET THE VIRTUAL ADDRESS TO PAGE 2
2712 011406 023727 172344 020000 CMP @KIPAR2,#20000 ; : : : ENDF
2713 011414 001401 BEQ 70$ ; : : : ENDFOR

```

C6

TEST CHECK PARITY DETECT LOGIC AND RAMS

```

2714 011416 000762          BR      60$          ;
2715                      ;
2716                      ; LET'S NOW MAKE SURE THAT THE ABORTS DON T OCCUR
2717                      ;
2718 011420          70$:
2719 011420 012737 001600 172344      MOV      #1600,@#KIPAR2      ; SET UP PAR 2
2720 011426 012701 040000              MOV      #40000,R1          ; SET VIRTUAL ADDRESS TO PAGE #2
2721 011432          75$:              BGND0
2722 011432 052737 000001 172100      BIS      #BIT0,@#MER        ; ENABLE PARITY ERRORS
2723 011440 105711              TSTB     (R1)              ; READ THE BYTE
2724 011442 012737 177777 003054      MOV      # 1,MERTAG        ;
2725                      ;
2726                      ; A
2727                      ; PARITY
2728                      ; ABORT
2729                      ; TRAP
2730                      ; SHOULD NOT
2731                      ; HAPPEN !!!
2732                      ;
2733                      ;
2734 011450 005737 004056          TST      PARABT            ; IF A PARITY ABORT HAS OCCURED
2735 011454 001407          BEQ      80$              ; : : THEN
2736 011456 005037 004056          CLR      PARABT            ; : :
2737 011462 005037 172100          CLR      MER                ; : : INITIALIZE THE MER
2738 011466 104063          ERROR   +63              ; : : ERROR A PARITY ABORT HAS OCCURED
2739 011470 005037 172100          clr     mer                ;
2740 011474 005201          80$:      INC      R1                ; : BUMP UP THE ADDRESS
2741 011476 020127 060000          CMP      R1,#60000         ; : IF WE HAVE PASSED THE PAGE BOUNDARY
2742 011502 103753          BLO     75$              ; : : THEN
2743 011504 062737 000200 172344      ADD      #200,@#KIPAR2     ; : : POINT KPAR2 TO A NEW PAGE
2744 011512 012701 040000              MOV      #40000,R1          ; : : SET THE VIRTUAL ADDRESS TO PAGE 2
2745 011516 023727 172344 020000      CMP      @#KIPAR2,#20000   ; : : ENDF
2746 011524 001401          BEQ     90$              ; DOUNTIL WE HAVE READ ALL ADDRESSES
2747 011526 000741          BR      75$              ;
2748 011530 005037 177572          90$:      CLR      @#SRO            ; DISABLE THE MMU
2749 011534 005037 172100          CLR      @#MER            ;
2750 011540 012637 000116          MOV     (SP)+,@#116        ;RESTORE PRIORITY
2751 011544 012637 000114          MOV     (SP)+,@#114        ;RESTORE VECTOR
2752 011550 000400          BR      TST11            ;:GO TO THE NEXT TEST
2753 011552          NXTST:

```

D6

TEST LTC BIT 7

2755
2756
2757
2758
2759
2760
2761
2762
2763
2764
2765
2766
2767
2768
2769
2770
2771
2772
2773
2774
2775
2776
2777
2778
2779
2780
2781
2782
2783
2784

```

.SBTTL TEST - LTC BIT 7
:*****
: LTC BIT 7 TEST
:
: This test check for the existance of the LTC register and it
: makes sure that the clock is ticking.
:
: BGNTST
: Set up timeout vector PC to TIMOUT
: Set up timeout vector PSW to 7
: Read the LTC CSR
: IF not timed OUT THEN
:   FOR a set amount of t'ne
:   : Check bit7
:   : IF BIT7 is set THEN
:   :   Increment BIT7 set flag
:   : ELSE
:   :   INCREMENT BIT7 Clear Flag
:   : ENDFOR
: : IF either flag has not been set at all THEN
: : Error Clock 's not ticking
:
: ENDTST
:
: TIMOUT: Clean stack
: Error LTC register timed out
:
: ---

```

```

2785 011552 000004
2786 011554 032777 010000 167356
2787 011562 001420
2787 011564 104401 011572
2787 011570 000415

2788 011624
2789 011624 013737 000004 003016
2790 011632 012737 011736 000004
2791 011640 012737 000340 000006
2792 011646 005737 177546
2793 011652 005000
2794 011654 005001
2795 011656 012703 000001
2796 011662 012702 177777
2797 011666 032737 000200 177546 1$:
2798 011674 001402
2799 011676 005200
2800 011700 000401
2801 011702 005201 5$:
2802 011704 077210 10$:
2803 011706 005700
2804 011710 001402
2805 011712 005701
2806 011714 001004
2807 011716 077315 20$:

```

```

:*****
:;LTC BIT7 TEST
:*****
TST11: SCOPE
BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
BEQ 1000$ ; THEN TYPE TEST TRACE
TYPE 30029$ ;:TYPE ASCIZ STRING
BR 30028$ ;:GET OVER THE ASCIZ
:;30029$: .ASCIZ <15><12>/TEST 11 - LTC BIT7 TEST/
30028$:
1000$:
MOV @#4,SAVTIM ; SAVE UNEXPECTED TIMEOUT
MOV #100$,@#4 ; SET UP THE TIMEOUT TRAP
MOV #340,@#6
TST LKS ; READ THE LKS REGISTER
CLR R0 ; CLEAR THE HIGH COUNTER
CLR R1 ; CLEAR THE LOW COUNTER
MOV #1,R3
MOV #177777,R2
BIT #BIT7,LKS ; SET UP A COUNT
BEQ 5$ ; FOR AN AMOUNT OF TIME
INC R0 ; : CHECK BIT 7 OF THE LKS
BR 10$ ; : IF IT IS A '1' THEN
; BUMP UP R0
INC R1 ; : ELSE
; BUMP UP R1
SOB R2,1$ ; ENDFOR
TST R0 ; IF R0 = 0 OR
BEQ 20$ ; R1 = 0 THEN
TST R1 ; ERROR WITH BIT7 OF LTC
BNE 25$ ; TOLERATE ONE ERROR ONLY
SOB R3,1$

```

E6

TEST LTC BIT 7

```
2808 011720 104053          ERROR +53          ; ENDTST
2809 011722 005037 172100  clr      mer
2810 011726 013737 003016 000004 25$:  MOV     SAVTIM,@#4  ; RESTORE UNEXPECTED TIMEOUT
2811 011734 000412          BR      TST12      ;;GO TO THE NEXT TEST
2812
2813          ;
2814          ; TIMEOUT ROUTINE
2815          ;
2816
2817 011736 012737 011752 000004 100$:  mov     #102$,@#4
2818 011744 104054          ERROR +54          ; REPORT ERROR
2819 011746 005037 172100  clr      mer
2820 011752 022626          CMP     (SP)+,(SP)+ ; CLEAN STACK
2821 011754 013737 003016 000004 102$:  MOV     SAVTIM,@#4  ; RESTORE UNEXPECTED TIMEOUT
2822
2823
2824
```

TEST LKS INTERRUPT PRIORITY

```

2826 .SBTTL TEST LKS INTERRUPT PRIORITY
2827 ;CHECK THAT LKS INTERRUPTS HAPPEN AT PRIORITY 5 CLEARING LKS<07>
2828 ;AND DON'T HAPPEN AT PRIORITY 6.
2829 ;ROUTINE TEST
2830 ;IF UFD AND LKS IS DISABLED THEN
2831 ;EXIT TEST
2832 ;ENDIF
2833 .
2834 . SET PRIORITY TO 5
2835 . CLEAR INTERRUPT FLAG
2836 . LET LKS<06>=#1 (ENABLE INTERRUPTS)
2837 . SET COUNTER TO WAIT FOR 3 INTERRUPTS
2838 . REPEAT
2839 .   DECREMENT COUNTER
2840 . UNTIL INTERRUPT FLAG EQ #3 OR COUNTER EQ #0
2841 . CLEAR LKS<06>
2842 . IF LKS<07> EQ #1 THEN
2843 .   ERROR (WAS NOT CLEARED ON INTERRUPT)
2844 . ENDIF
2845 . IF COUNTER LT TIME_REQUIRED_FOR_3_INTERRUPTS FOR 800HZ
2846 .   ERROR (INTERRUPTS NEVER GO LOW)
2847 . ENDIF
2848 . IF INTERRUPT FLAG LT #3 THEN
2849 .   ERROR (INTERRUPTS DON'T HAPPEN)
2850 . ENDIF
2851 . CLEAR INTERRUPT_FLAG
2852 . WAIT FOR LKS<7>=1
2853 . LET LKS<7>=0
2854 . IF LKS<7> NE #0 THEN
2855 .   ERROR (LKS<7> NOT CLEARED)
2856 . ENDIF
2857 . SET PRIORITY TO 6
2858 . SET COUNTER TO 1 SLOW CLOCK INTERRUPT
2859 . SET LKS<06>
2860 . REPEAT
2861 .   DECREMENT COUNTER
2862 . UNTIL COUNTER EQ #0 OR INTERRUPT_FLAG NE #0
2863 . IF INTERRUPT_FLAG NE #0 THEN
2864 .   ERROR (INTERRUPT WAS AT WRONG PRIORITY)
2865 . ENDIF
2866 . RESTORE ORIGINAL PRIORITY
2867 ;ENDROUTINE
2868 ;ROUTINE LINE_CLOCK_INTERRUPT
2869 ;INCREMENT INTERRUPT_FLAG
2870 ;ENDROUTINE
2871
2872 ;*****
TST12: SCOPE
2873 011762 000004 BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
2874 011764 032777 010000 167146 BEQ 1000$ ; THEN TYPE TEST TRACE
2875 011772 001425 TYPE ,30031$ ;;TYPE ASCIZ STRING
012000 000422 BR 30030$ ;;GET OVER THE ASCIZ
;;30031$: .ASCIZ <15><12>/TEST 12 - LKS INTERRUPT PRIORITY/
30030$:
1000$:
2876 012046 BIT #BIT06,@#52 ;UFD MODE?
2877 012046 032737 000100 000052 BEQ 1$ ;IF NOT, GO DO TEST
2878 012054 001400

```

G6

TEST LKS INTERRUPT PRIORITY

```

2879                                     ;
2880                                     ; WAIT FOR 3 INTERRUPTS AND CHECK LKS<7> TO BE 0 AFTER INTERRUPT
2881                                     ;
2882 012056 042737 000100 177546 1$:   BIC    #BIT06,LKS           ; FROM END OF TEST 42?? PROBLEM
2883 012064 005037 002402                CLR    LKSFL              ; CLEAR INTERRUPT FLAG
2884 012070 005037 002404                clr    lkcnt
2885 012074 013737 000100 002406        mov    @#100,savloc       ; save wht was in 100
2886 012102 012737 027530 000100        mov    #clkcnt,@#100     ; MOV    #LKSINT,100           ; POI
NT VECTOR TO ROUTINE
2887                                     ;
2888 012110 012701 077777                MOV    #77777,R1         ; COUNTER FOR SLOW CLOCK
2889 012114 052737 000100 177546        BIS    #BIT06,LKS       ; SET INTERRUPT ENABLE BIT
2890 012122 032737 000100 177546        BIT    #BIT06,LKS       ; BIT SET OK?
2891 012130 001003                BNE    2$                ; IF YES, BRANCH
2892 012132 104042                ERROR  +42              ; ERROR WRITING 1 TO LKS<6>
2893 012134 005037 172100                clr    mer
2894 012140 106427 000240                MTPS   #240             ; SET PRIORITY TO 5
2895 012144 004737 030610                2$:   jsr    pc,delay
2896 012150 022737 000003 002404        3$:   cmp    #3,lkcnt      ; CMP    #3,LKSFL           ; 3 I
NTERRUPTS HAPPENED?
2897 012156 001372                bne    3$                ; BEQ    4$                 ; IF
YES, BRANCH
2898                                     ;
2899                                     ;
2900                                     ; DISABLE INTERRUPTS AND CHECK THAT PROPER CONDITIONS ARE MET
2901                                     ;
2902 012160 106427 000340                4$:   MTPS   #340         ; RAISE PRIORITY
2903 012164 042737 000100 177546        BIC    #BIT06,LKS       ; DISABLE INTERRUPTS
2904 012172 032737 000100 177546        6$:   BIT    #BIT06,LKS       ; LKS<6>-0?
2905 012200 001403                BEQ    7$                ; IF YES, BRANCH
2906 012202 104042                ERROR  +42              ; ERROR WRITING 0 TO LKS<6>
2907 012204 005037 172100                clr    mer
2908 012210                7$:
2909 012210 022737 000003 002404        8$:   CMP    #3,lkcnt      ; DID 3 INTERRUPTS HAPPEN?
2910 012216 001406                BEQ    9$                ; IF YES, BRANCH
2911 012220 012737 000003 001124        MOV    #3,$GDDAT       ; 3 INTERRUPTS EXPECTED
2912 012226 104007                ERROR  +7                ; INTERRUPTS DON'T HAPPEN
2913 012230 005037 172100                clr    mer
2914                                     ;
2915                                     ; CHECK WHETHER INTERRUPTS HAPPEN AT PRIORITY 6
2916                                     ;
2917 012234 005037 002404                9$:   CLR    lkcnt         ; CLEAR INTERRUPT FLAG
2918 012240 106427 000300                MTPS   #300             ; RAISE PRIORITY TO 6
2919 012244 012701 077777                MOV    #77777,R1       ; COUNTER FOR SLOW CLOCK
2920 012250 052737 000100 177546        BIS    #BIT06,LKS       ; SET INTERRUPT ENABLE BIT
2921 012256                10$:
2922 012256 005737 002404                TST    lkcnt            ; ANY INTERRUPTS?
2923 012262 001002                bne    11$              ; IF YES, EXIT LOOP
2924 012264 004737 030610                jsr    pc,delay        ; SOB    R1,10$           ; CON
TIME WITH COUNT
2925                                     ;
2926 012270 005737 002404                11$:  TST    lkcnt         ; ANY INTERRUPTS?
2927 012274 001406                BEQ    12$              ; IF NO, BRANCH
2928 012276 012737 000006 001124        MOV    #6,$GDDAT       ; STORE PRIORITY FOR TYPE OUT
2929 012304 104010                ERROR  +10              ; INTERRUPTS HAPPEN AT WRONG PRIORITY
2930 012306 005037 172100                clr    mer
2931 012312 106427 000340                12$:  MTPS   #340             ; RESTORE PRIORITY
2932 012316 042737 000100 177546        BIC    #BIT6,LKS       ; DISABLE CLOC INTERRUPTS
2933 012324 013737 002406 000100        mov    @#savloc,@#100

```


TEST RESETTING LKS

```

2935 .SBTTL TEST RESETTING LKS
2936 ;RESETTING LKS(*)
2937 ;THIS TEST WILL PROVE THAT RESET INSTRUCTION CLEARS LKS<06>.
2938 ;ROUTINE TEST
2939 ;IF UFD AND LKS IS DISABLED THEN
2940 .EXIT TEST
2941 ;ENDIF
2942 ;. POINT LKS VECTOR 100 TO ERROR LKS ILLEGAL INTERRUPT
2943 ;. SYNCHRONIZE LKS BY WAITING FOR 3 PULSES
2944 ;. LET LKS<06>=#1
2945 ;. EXECUTE "RESET"
2946 ;. IF LKS<6> NE #0 THEN
2947 ;. ERROR
2948 ;. ENDF
2949 ;. IF ILLEGAL_LINE_CLOCK INTERRUPT NE 0 THEN
2950 ;. ERROR
2951 ;. ENDF
2952 ;ENDROUTINE
2953 ;
2954 ;ROUTINE ERROR_LKS_ILLEGAL_INTERRUPT
2955 ;. FLAG ILLEGAL_LINE_CLOCK INTERRUPT
2956 ;RETURN
2957
2958 ;*****
2959 012332 000004 TST13: SCOPE
2960 012334 032777 010000 166576 BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
2961 012342 001423 BEQ 1000$ ; THEN TYPE TEST TRACE
2961 012344 104401 012352 TYPE ,30033$ ;;TYPE ASCIZ STRING
2961 012350 000420 BR 30032$ ;;GET OVER THE ASCIZ
2962 012412 ;;30033$: .ASCIZ <15><12>/TEST 13 - RESETTING LKS TEST/
2962 012412 30032$:
2963 012412 005737 001206 1000$: TST $PASS ;FIRST PASS?
2964 012416 001053 BNE TST14 ;;IF NOT FIRST PASS, EXIT TEST
2965 012420 032737 000100 000052 BIT #BIT06,@#52 ;UFD MODE?
2966 012426 001400 BEQ 1$ ;IF NOT, BRANCH
2967
2968 ; SYNCHRONISE WITH LINE TIME CLOCK BY WAITING FOR 3 INTERRUPTS
2969
2970 012430 1$:
2971 012430 013737 000100 002406 mov @#100,savloc
2972 012436 005037 002404 clr lkcnt
2973 012442 012737 027530 000100 mov #clkcnt,@#100
2974 ;MOV #LKSINT,@#100 ;SET
2975 012450 012737 000340 000102 MOV #340,@#102 ;AT PROIRITY 7
2976 012456 052737 000100 177546 BIS #BIT06,LKS ;SET INTERRUPT ENABLE BIT
2977 012464 005037 002402 CLR LKSFL ;CLEAR INTERRUPTS FLAG
2978 012470 012702 077777 MOV #77777,R2 ;COUNTER TO WAIT FOR INTERRUPTS
2979 012474 106427 000240 MTPS #240 ;LOWER PRIORITY TO 5
2980 012500 004737 030610 2$: jsr pc,delay ;CMP #3,LKSFL ;3 I
2981 012504 022737 000003 002404 cmp #3,lkcnt ;BEQ 3$ ;EXI
2982 012512 001372 bne 2$ ;SOB R2,2$ ;OTH
2983
2984 012514 106427 000340 3$: MTPS #340 ;RAISE PRIORITY TO 7
2985 012520 000005 RESET ;EXECUTE RESET
2986
2987 012522 013737 002406 000100 4$: mov savloc,@#100

```

I6

TEST RESETTING LKS

2988 012530 032737 000100 177546
2989 012536 001403
2990 012540 104011
2991 012542 005037 172100
2992

BIT #BIT06,LKS
BEQ TST14
ERROR +11
clr mer

::IF SO, EXIT TEST ;INTERRUPT ENABLE BIT CLEARED?
;RESET DOESN'T CLEAR LKS

TEST - MAINTENANCE REGISTER TEST

```

2994 .SBTTL TEST - MAINTENANCE REGISTER TEST
2995 ;MAINTENANCE REGISTER TEST
2996 ;THIS TEST WILL ADDRESS MAINTENANCE REGISTER AND CHECK BITS
2997 ;7 4 TO BE 0010, 2 1 TO BE 10, AND READ BITS 10 08, 03, 00
2998 ;FOR FUTURE USE. THOSE BITS REPRESENT THE FOLLOWING SIGNALS:
2999 ;MULTIPROCESSOR SLAVE, UNIBUS SYSTEM, FPA AVAILABLE, HALT/TRAP
3000 ;OPTION, AND AC POWER OKAY.
3001 ;ROUTINE TEST
3002 ;. IF MAINT. REG. BITS <7 4> NE 0010 OR <2 1> NE 10 THEN
3003 ;. ERROR
3004 ;. ENDF
3005 ;. READ MAINT.REG. BITS <10-08,03,00>
3006 ;ENDROUTINE
3007
3008 ;*****
3009 012546 000004 TST14: SCOPE
3010 012550 032777 010000 166362 BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
3011 012556 001426 BEQ 1000$ ; THEN TYPE TEST TRACE
3011 012560 104401 012566 TYPE ,30035$ ;;TYPE ASCIZ STRING
3011 012564 000423 BR 30034$ ;;GET OVER THE ASCIZ
;;30035$: .ASCIZ <15><12>/TEST 14 MAINTENANCE REGISTER TEST/
30034$:
1000$:
3012 012634 BIT #177000,MAIREG ;UNUSED BITS ALL ZEROS? ;MC001
3013 012634 032737 177000 177750 BEQ 1$ ;CHANGED FROM 174000 FOR THE DA
3014 ;IF OK, BRANCH
3015 012642 001403 ERROR +43 ;MAINTENANCE REGISTER ERROR
3016 012644 104043 clr mer
3017 012646 005037 172100 1$: BIT #BIT3,MAIREG ;IF JUMPER IS IN
3018 012652 032737 000010 177750 BNE 2$ ;; THEN
3019 012660 001010 BIC #BIT3,MAIREG ;; SET HALT TO ODT
3020 012662 042737 000010 177750 CMP #105,MAIREG ;; MAINTENANCE REGISTER SHOULD BE SET UP TO
3021 012670 022737 000105 177750
105
3022 ;; CHANGED FROM <5,2> FOR THE DA
3023 012676 001401 BEQ 2$ ;IF SO, BRANCH
3024 012700 104043 ERROR +43 ;MAINTENANCE REGISTER ERROR
3025 012702 005037 172100 2$: clr mer
3026

```

TEST SERIAL LINE UNIT REGISTERS

```

3028 .SBTTL TEST SERIAL LINE UNIT REGISTERS
3029 ;SERIAL LINE UNIT TEST(*)
3030 ;BCR<2 0> WILL BE READ TO FIND OUT BAUD RATE. SLU WILL BE PROG
3031 ;RAMMED TO CHECK THE INTERRUPT LEVELS BY SETTING BIT<06> IN
3032 ;RCSR AND XMIT. LOOP BACK CAPABILITIES WILL BE TESTED BY SETTING
3033 ;TO 1 XCSR<02>. THE LINE CLOCK INTERRUPT SUBROUTINE WILL BE
3034 ;USED TO RETURN TO THE EXECUTION OF THE DIAGNOSTICS, IF THE
3035 ;PROGRAM HANGS IN THE LOOP BACK MODE.
3036 ;ROUTINE TEST
3037 ;IF UFD AND CONSOLE NOT PRESENT
3038 ; GO TO TEST 22
3039 ;ENDIF
3040 ; IF BCR<07> EQ #0 THEN
3041 ; READ BCR<2 0> TO GET BAUD RATE
3042 ;ENDIF
3043 ; LET 4=ADDRESS OF TIMEOUT ROUTINE
3044 ; DO FOR RCSR,XCSR,RBUF,XBUF
3045 ; READ XCSR,XCSR,RBUF,XBUF
3046 ; IF TIMEOUT FLAG NE #0 THEN
3047 ; ERROR
3048 ;ENDIF
3049 ; ENDDO
3050 ;ENDROUTINE
3051 ;ROUTINE TIMEOUT
3052 ; LET TIMEOUT FLAG=#1
3053 ;ENDROUTINE
3054
3055 ;*****
3056
3057 012706 000004 TST15: SCOPE
3058 012710 032777 010000 166222 BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
3059 012716 001426 BEQ 1000$ ; THEN TYPE TEST TRACE
3059 012720 104401 012726 TYPE ,30037$ ;;TYPE ASCIZ STRING
3059 012724 000423 BR 30036$ ;;GET OVER THE ASCIZ
3059 ;:30037$: .ASCIZ <15><12>/TEST 15 - SLU REGISTER ACCESS TEST/
3059 ;:30036$:
3060 012774
3061 012774 032737 000100 000052 BIT #BIT06,@#52 ;UFD MODE?
3062 013002 001402 BEQ 1$ ;IF NOT, GO DO THE TEST
3063 013004 000137 016714 JMP SLEND ;IF TRUE, SKIP ALL SLU TESTS
3064
3065 ; TRY TO ACCESS SLU REGISTERS
3066
3067 013010 013701 000004 1$: MOV ERRVEC,R1 ;SAVE TIMEOUT VECTOR
3068 013014 012737 013050 000004 MOV #3$,ERRVEC ;POINT NEW ONE TO PROGRAM AREA
3069 013022 012737 000340 000006 MOV #340,ERRVEC+2 ;AT PRIORITY 7
3070 013030 012702 177560 MOV #RCSR,R2 ;START ACCESSING WITH RCSR
3071 013034 012703 177566 MOV #XBUF,R3
3072 013040 012704 000002 MOV #2,R4
3073 013044 005712 2$: TST (R2) ;ACCESS SLU REGISTER
3074 013046 000405 BR 4$ ;IF NO TIMEOUT, CONTINUE
3075 013050 010237 001126 3$: MOV R2,$BDDAT ;STORE ADDRESS THAT TIMED OUT
3076 013054 104012 ERROR +12 ;TIMEOUT ACCESSING SLU REGISTER
3077 013056 005037 172100 CLR mer
3078 013062 020322 4$: CMP R3,(R2)+ ;LAST REGISTER ACCESSED?
3079 013064 103767 BLO 2$ ;IF NOT, BRANCH
3080 013066 012702 176500 MOV #RCSR1,R2 ;GET POINTERS TO SLU 1

```

TEST SERIAL LINE UNIT REGISTERS

3081 013072 012703 176506
3082 013076 077416
3083 013100 010137 000004
3084

MOV #XBUF1,R3
SOB R4,2\$
MOV R1,ERRVEC

;GET LAST ADDRESS ON SLU 1
;
;RESTORE TIMEOUT VECTOR

M6

TEST XCSR BIT 7

3086
3087
3088
3089
3090
3091
3092
3093
3094
3095
3096
3097
3098
3099
3100
3101
3102
3103
3104

```

.SBTTL TEST XCSR BIT 7
;CHECK THAT XCSR<07> CAN BE 0 AND 1.
;XCSR <07> TRANSMITTER READY
;ROUTINE TEST
;.. WAIT FOR XCSR<07>=#1 NO MORE THAN 200MSEC
;.. IF XCSR<07> NE #1 THEN
;.. ERROR
;.. ENDF
;.. LET XBUF=#NULL
;.. WAIT FOR XCSR<07>=#1
;.. LET XBUF=#NULL
;.. IF XCSR<07> NE 0 THEN
;.. ERROR (READY DIDN'T GO LOW)
;.. ENDF
;ENDROUTINE

```

3105 013104 000004
3106 013106 122737 000001 001220
3107 013114 001006
3108 013116 005737 001206
3109 013122 001403
3110 013124 012704 000002
3111 013130 000455
3112 013132 032777 010000 166000
3113 013140 001423
3114 013142 104401 013150
013146 000416

```

TST16: SCOPE
CMPB #APTENV,$ENV ;ARE WE IN APT MODE?
BNE 1002$ ;IF NOT: DO THIS TEST
TST $PASS ;FIRST PASS??
BEQ 1002$ ;IF YES, DO IT
MOV #2,R4 ; LOOP COUNT OF 1
BR 4$ ; GO SET UP POINTERS TO SLU 1

1002$: BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
BEQ 1000$ ; THEN TYPE TEST TRACE
TYPE ,30039$ ;;TYPE ASCIZ STRING
BR 30038$ ;;GET OVER THE ASCIZ
;;30039$: .ASCIZ <15><12>/TEST 16 XCSR BIT 7 TEST/
30038$:

```

3115 013204
3116 013204 004737 030610
AST CHARACTER
3117 013210
3118

```

;MOV #100000,R1 ; WAIT FOR L
;1001$: SOB R1,1001$
;MOV #1000,R1 ;OU

```

3119 013210 012702 177564
3120 013214 012703 177566
3121 013220 012704 000002
3122 013224 105712
3123 013226 100402
3124

```

MOV #XCSR,R2 ;
MOV #XBUF,R3 ;
MOV #2,R4 ;
1$: TSTB (R2) ;XCSR<7> READY 1?
BMI 2$ ;IF SO, EXIT WAIT LOOP
;SOB R1,1$ ;IF

```

3125 013230 004737 030610
3126

```

jsr pc,delay

```

3127 013234 105712
3128 013236 100403
3129 013240 104013
3130 013242 005037 172100
3131 013246 012713 000000
3132 013252 105712
3133 013254 100003
3134 013256 104013
3135 013260 005037 172100
3136 013264 012702 176504
3137 013270 012703 176506
3138 013274 012701 001000

```

2$: TSTB (R2) ;XCSR<7>=1?
BMI 3$ ;IF YES, BRANCH
ERROR +13 ;XCSR<7> DOES NOT BECOME 1
clr mer
3$: MOV #NULL,(R3) ;TRY TO TRANSMIT NULL CHARACTER
TSTB (R2) ;XCSR<7>=0
BPL 4$ ;
ERROR +13 ;XMIT READY DIDN'T GO LOW
clr mer
4$: MOV #XCSR1,R2 ;
MOV #XBUF1,R3 ;
MOV #1000,R1 ;

```

N6

Q 0078

TEST - XCSR BIT 7

3139 013300 077427
3140

S08 R4.1\$:

B

TEST RCSR BIT 7 AND XCSR BIT 2

```

3142 .SBTTL TEST RCSR BIT 7 AND XCSR BIT 2
3143 ;CHECK THAT RCSR<07> CAN BE 0 AND 1 AND THAT XCSR<02> WORKS PROPER Y.
3144 ;
3145 ;RCSR <07> RECEIVER DONE
3146 ;XCSR <02> MAINTENANCE
3147 ;
3148 ;ROUTINE TEST
3149 ;.(CHECK RCSR<07> AND XCSR<07>)
3150 ;. WAIT FOR XCSR<07>=#1
3151 ;. LET XCSR<02>=#1 (LOOP BACK MODE)
3152 ;. LET XBUF=#125
3153 ;. WAIT FOR RCSR<07>=#1 NO MORE THAN 200MSEC
3154 ;. IF RCSR<07> NE #1 THEN
3155 ;. . ERROR (RCSR<07> DOES NOT BECOME 1 OR XCSR<02>DOES NOT
3156 ;. . WORK)
3157 ;. ENDF
3158 ;. IF RBUF NE #125 THEN
3159 ;. . ERROR
3160 ;. ENDF
3161 ;. IF RCSR<07> NE #0 THEN
3162 ;. . ERROR (RCSR<07>DOES NOT GO LOW)
3163 ;. ENDF
3164 ;. LET XCSR<02>=#0
3165 ;ENDROUTINE
3166
3167 ;*****
3168 013302 000004 TST17: SCOPE
3169 013304 122737 000001 001220 CMPB #APTENV,$ENV ;ARE WE IN APT MODE?
3170 013312 001006 BNE 1002$ ;IF NOT: DO THIS TEST
3171 013314 005737 001206 TST $PASS ;FIRST PASS??
3172 013320 001403 BEQ 1002$ ;IF YES, DO IT
3173 013322 012703 000002 MOV #2,R3 ; LOOP COUNT OF 1
3174 013326 000555 BR 12$ ; GO SET UP POINTERS TO SLU 1
3175 013330 032777 010000 165602 1002$: BIT #BIT12,$SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
3176 013336 001431 BEQ 1000$ ; THEN TYPE TEST TRACE
3176 013340 104401 013346 TYPE ,30041$ ;:TYPE ASCIZ STRING
3176 013344 000426 BR 30040$ ;:GET OVER THE ASCIZ
3177 013422 ;:30041$: .ASCIZ <15><12>/TEST 17 RCSR BIT 7 AND XCSR BIT 2 TEST/
3178 ;:30040$: ;MOV #100000,R1 ; WAIT FOR L
3179 013422 1000$: ;1001$: SOB R1,1001$
3180 3181 013422 004737 030610 jsr pc,delay
3182 3183 013426 012701 000013 MOV #13,R1 ;COUNTER FOR ABOUT 200MICROSEC.
3184 013432 012702 177560 MOV #RCSR,R2 ;
3185 013436 012703 000002 MOV #2,R3
3186 013442 004737 030610 1$: jsr pc,delay
3187 013446 105762 000004 TSTB 4(R2) ;XCSR<7> READY 1?
3188 013452 100410 bmi 2$ ;BPL 1$ ;IF
3189 013454 004737 030610 jsr pc,delay
3190 013460 105762 000004 tstb 4(r2)
3191 013464 100403 bmi 2$
3192 013466 104076 error +76
3193 013470 005037 172100 clr mer
3194 013474 052762 000004 000004 2$: BIS #BIT02,4(R2) ;SET LOOP BACK MODE

```


TEST RCSR BIT 7 AND XCSR BIT 2

```

3195 013502 032762 000004 000004      BIT      #BIT02,4(R2)      ;GOT SET OK?
3196 013510 001006                      BNE      3$              ;IF YES, BRANCH
3197 013512 005062 000004      CLR      4(R2)          ;RESET TO PRINT ERROR
3198 013516 104034      ERROR   +34            ;XCSR<2> DOES NOT BECOME
3199 013520 005037 172100      clr     mer
3200 013524 000465      BR      TST20          ;;EXIT TEST
3201
3202      ; STALL FOR A WHILE IN CASE XCSR<2> CAUSES RCSR<7> TO BE 1
3203      ;
3204 013526      3$:              ;MOV     #60000,R1      ;STA
LL IN CASE XCSR<2> SETS READY
3205 013526 105712      4$:      TSTB     (R2)      ;IF RECEIVER READY SET?
3206 013530 100403      BMI     5$              ;IF SET, BRANCH
3207 013532 004737 030610      jsr     pc.delay        ;SOB    R1,4$      ;OTH
ERWISE, STAY FOR A WHILE
3208
3209
3210 013536 000402      BR      6$              ;IF NOT READY, BRANCH
3211 013540 005762 000002      5$:      TST     2(R2)      ;READ RBUF
3212
3213      ; TRANSMIT XON AND CHECK RCSR<7>
3214
3215 013544 012762 000021 000006      6$:      MOV     #21,6(R2)      ;TRANSMIT A CHARACTER
3216
3217 013552 105712      7$:      TSTB     (R2)      ;RCSR<7> READY 1?
3218 013554 100402      BMI     8$              ;IF YES, EXIT WAIT LOOP
3219 013556 004737 030610      jsr     pc.delay        ;SOB    R1,7$      ;OTH
ERWISE, CONTINUE WAITING
3220
3221
3222 013562 105712      8$:      TSTB     (R2)      ;RCSR<7>=1?
3223 013564 100405      BMI     9$              ;IF YES, BRANCH
3224 013566 005062 000004      CLR     4(R2)          ;RESET XCSR<2>
3225 013572 104014      ERROR   +14            ;RECEIVER READY DIDN'T COME UP
3226 013574 005037 172100      clr     mer
3227 013600 016237 000002 001126      9$:      MOV     2(R2), $BDDAT    ;STORE RECEIVED DATA
3228 013606 022737 000021 001126      CMP     #21, $BDDAT     ;DATA RECEIVED OK?
3229 013614 001410      BEQ     10$            ;IF YES, BRANCH
3230 013616 012737 000021 001124      MOV     #21, $GDDAT
3231 013624 005062 000004      CLR     4(R2)          ;RESET TO ENABLE SLU
3232 013630 104015      ERROR   +15            ;WRONG CHARACTER RECEIVED
3233 013632 005037 172100      clr     mer
3234 013636 105712      10$:     TSTB     (R2)      ;RCSR<7>=0?
3235 013640 100005      BPL     11$            ;IF ZERO, BRANCH
3236 013642 005062 000004      CLR     4(R2)          ;RESET TO ENABLE SLU
3237 013646 104016      ERROR   +16            ;RCSR<07><<0 AFTER READING RBUF
3238 013650 005037 172100      clr     mer
3239 013654 042762 000004 000004      11$:     BIC     #BIT02,4(R2)    ;DISABLE LOOP BACK MODE
3240 013662 012702 176500      12$:     MOV     #RCSR1,R2    ;POINT TO SLU 1
3241 013666 012701 000013      MOV     #13,R1         ;SET UP COUNTER
3242
3243 013672 005303      dec     r3
3244 013674 005703      tst     r3
3245 013676 001261      bne     1$
3246

```

TEST RESET AND XCSR<2!0>

```

3248 .SBTTL TEST RESET AND XCSR<2!0>
3249 ;CHECK THAT RESET CLEARS XCSR<0!2>.
3250 ;ROUTINE TEST
3251 ;.(CHECK RCSR<07> AND XCSR<07> AND RESET)
3252 ;. LET XCSR<02,00>=#1 (LOOP BACK MODE)
3253 ;. EXECUTE "RESET"
3254 ;. IF XCSR<02!00> NE #0 THEN
3255 ;. ERROR
3256 ;. ENDF
3257 ;. LET XCSR<02>=#0
3258 ;ENDROUTINE
3259
3260 ;*****
3261 013700 000004 TST20: SCOPE
3262 013702 122737 000001 001220 CMPB #APTENV,$ENV ;ARE WE IN APT MODE?
3263 013710 001006 BNE 1001$ ;IF NOT: DO THIS TEST
3264 013712 005737 001206 TST $PASS ;FIRST PASS??
3265 013716 001403 BEQ 1001$ ;IF YES, DO IT
3266 013720 012703 000002 MOV #2,R3 ; LOOP COUNT OF 1
3267 013724 000451 BR 10$ ; GO SET UP POINTERS TO SLU 1
3268 013726 032777 010000 165204 1001$: BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
3269 013734 001426 BEQ 1000$ ; THEN TYPE TEST TRACE
3270 013736 104401 013744 TYPE ,30043$ ;:TYPE ASCIZ STRING
3271 013742 000423 BR 30042$ ;:GET OVER THE ASCIZ
3272 014012 ;:30043$
3273 014012 30042$: .ASCIZ <15><12>/TEST 20 - RESET AND XCSR<0!2> TEST/
3274 1000$:
3275 MOV #XCSR,R2 ;
3276 MOV #2,R3 ;
3277 1$: BIS #BIT02!BIT00,(R2) ;LOOP BACK MODE
3278 ; EXECUTE RESET AND VALIDATE THAT XCSR<7,2> BECOMES <1,0>
3279 ;
3280 RESET ;EXECUTE RESET
3281 BIT #BIT02!BIT00,(R2) ;XCSR<2,0> CLEAR?
3282 BEQ 10$
3283 BIC #BIT02!BIT00,(R2) ;CLEAR THE BITS SO WE CAN REPORT
3284 ERROR +22 ;XCSR<2,0> NOT CLEARED ON RESET
3285 clr mer
3286 10$: MOV #XCSR1,R2 ;
3287 SOB R3,1$ ;

```

E7

TEST RESET AND INTERRUPT ENABLE BITS

3287
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298
3299
3300
3301
3302
3303
3304
3305
3306
3307
3308
3309
3310
3311
3312
3313
3314
3315

```

.SBTTL TEST RESET AND INTERRUPT ENABLE BITS
;CHECK THAT INTERRUPTS DON'T HAPPEN AT PRIORITY 4 AND THAT RESET
;CLEARS XCSR<06> AND RCSR<06>.
;RCSR <06> RECEIVER INTERRUPT ENABLE
;XCSR <06> TRANSMITTER INTERRUPT ENABLE
;ROUTINE TEST
... LET 60=#ADDRESS_OF_ILLEGAL_INTERRUPT_XRCSR
... LET 64=#ADDRESS_OF_ILLEGAL_INTERRUPT_XRCSR
... SET PRIORITY TO 4
... LET XCSR<02>=#1 (LOOPBACK MODE)
... LET XCSR<06>=#1 (ENABLE TRANSMIT INTERRUPTS)
... LET RCSR<06>=#1 (ENABLE RECEIVE INTERRUPTS)
... WAIT FOR XCSR<07>=#1 (READY TO TRANSMIT)
... LET XBUF=#NULL (SEND A CHARACTER)
... WAIT FOR ILLEGAL INTERRUPTS (ABOUT 200MSEC)
... EXECUTE "RESET"
... IF XCSR<06> NE #0 OR RCSR<06> NE #0 OR XRCSR NE #0 THEN
...     ERROR
... ENDIF
... RESTORE PRIORITY TO NORMAL
;ENDROUTINE
;ROUTINE ILLEGAL_INTERRUPT_XRCSR
... INCREMENT XRCSR
;ENDROUTINE

```

```

3316 014056 000004
3317 014060 122737 000001 001220
3318 014066 001015
3319 014070 005737 001206
3320 014074 001412
3321 014076 012705 000002
3322 014102 013737 000004 003016
3323 014110 012737 015000 000004
3324 014116 000137 014642
3325 014122 032777 010000 165010
3326 014130 001433
3327 014132 104401 014140
3328 014136 000430
3329 014220
3330 014220 005037 002360
3331 014224 005037 002356
3332 014230 012737 015000 000004
3333 014236 012701 000060
3334 014242 012704 177560
3335 014246 012705 000002
3336 014252 005037 002360
3337 014256 005037 002356

```

```

;*****
TST21: SCOPE
CMPB #APTENV,$ENV ;ARE WE IN APT MODE?
BNE 1001$ ;IF NOT: DO THIS TEST
TST $PASS ;FIRST PASS??
BEQ 1001$ ;IF YES, DO IT
MOV #2,R5 ; LOOP COUNT OF 1
MOV @#4,$AVTIM ; STORE UNEXPECTED TIMEOUT
MOV #100,$@#4 ; SET UP TIMEOUT
JMP 20$ ; GO SET UP POINTERS TO SLU 1
1001$: BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
BEQ 1000$ ; THEN TYPE TEST TRACE
TYPE ,30045$ ;;TYPE ASCIZ STRING
BR 30044$ ;;GET OVER THE ASCIZ
;;30045$: .ASCIZ <15><12>/TEST 21 - SLU RESET AND INTERRUPT ENABLE TEST/
30044$:
1000$:
clr tcount
clr rcount
MOV #100,$@#4 ;
MOV #60,R1 ; R1 POINTS TO SLU 0 INTERRUPT VECTOR
MOV #RCSR,R4 ; R4 POINTS TO SLU 0 REGISTERS
MOV #2,R5 ; R5 IS THE LOOP COUNT
;
; CHECK THAT INTERRUPTS ENABLE BITS FOR RECEIVER AND TRANSMITTER OF SLU
; ARE CLEARED BY RESET
;
1$: clr tcount
clr rcount

```

F7

TEST RESET AND INTERRUPT ENABLE BITS

```

3340 014262 052764 000100 000004      BIS      #BIT06,4(R4)      ;SET INTERRUPT ENABLE BIT IN XCSR
3341 014270 032764 000100 000004      BIT      #BIT06,4(R4)      ;GOT SET OK?
3342 014276 001003                      BNE     2$                ;IF YES, BRANCH
3343 014300 104067                      ERROR   +67                ;IN BIT 6 OF XCSR
3344 014302 005037 172100          clr     mer
3345 014306 052714 000100 2$:      BIS      #BIT06,(R4)      ;SET INTERRUPT ENABLE BIT IN RCSR
3346 014312 032714 000100          BIT      #BIT06,(R4)      ;GOT SET OK?
3347 014316 001003                      BNE     3$                ;IF YES, BRANCH
3348 014320 104067                      ERROR   +67                ;IN BIT 6 OF RCSR
3349 014322 005037 172100          clr     mer
3350 014326 000005                      RESET
3351 014330 032764 000100 000004 3$:      BIT      #BIT06,4(R4)      ;INLINE BUS RESET
3352 014336 001403                      BEQ     4$                ;XMIT INTERRUPT ENABLE BIT CLEARED?
3353 014340 104022                      ERROR   +22                ;IF CLEARED, BRANCH
3354 014342 005037 172100          clr     mer                ;INTERRUPT ENABLE NOT CLEARED ON RESET
3355 014346 032714 000100 4$:      BIT      #BIT06,(R4)      ;RECEIVE INTERRUPT ENBLE CLEARED?
3356 014352 001403                      BEQ     5$                ;IF CLEARED, BRANCH
3357 014354 104022                      ERROR   +22                ;INTERRUPT ENABLE NOT CLEARED ON RESET
3358 014356 005037 172100          clr     mer
3359
3360          ; CHECK THAT TRANSMIT INTERRUPTS DON'T HAPPEN AT PRIORITY HIGHER THAN 3
3361          ;
3362 014362 5$:
3363 014362 012761 015036 000004      MOV     #200$,4(R1)        ;POINT XMIT VECTOR TO PROGRAM AREA
3364 014370 012761 000340 000006      MOV     #340,6(R1)        ;AT PRIORITY 7
3365 014376 052764 000100 000004      BIS     #BIT06,4(R4)      ;SET INTERRUPT ENABLE BIT IN XCSR
3366 014404 012702 000340          MOV     #340,R2           ;SET PRIORITY TO 7
3367 014410 000402                      BR      7$                ;GO WAIT IN CASE OF INTERRUPTS
3368 014412 162702 000040 6$:      SUB     #40,R2            ;LOWER PRIORITY LEVEL
3369 014416 106402 7$:      MTPS   R2                ;SET PRIORITY
3370 014420 004737 030610          JSR     PC,DELAY           ;TIMER ROUTINE
3371          ;           MOV     #41,R3           ;MC           ;TIME DELAY
3372 014424 005737 002360          tst     tcount
3373 014430 001403          beq     10$              ;8$:      SOB     R3,8$           ;WAIT FOR IN
TERRUPTS
3374          ;BR     10$           ;IF INTERRUPTS DIDN'
T HAPPENED, BRANCH
3375 014432 104021 9$:      ERROR   +21                ;INTERRUPTS HAPPEN AT WRONG PRIORITY
3376 014434 005037 172100          clr     mer
3377          ;CMP     (SP)+,(SP)+           ;CLEAN UP THE STACK
3378 014440 022702 000200 10$:     CMP     #200,R2           ;AT PRIORITY 4?
3379 014444 001362                      BNE     6$                ;IF NOT LAST ONE, CONTINUE
3380 014446 106427 000340          MTPS   #340              ;RESTORE PRIORITY 7
3381 014452 042764 000100 000004      BIC     #BIT06,4(R4)      ;CLEAR INTERRUPT ENABLE BIT
3382
3383          ; CHECK THAT RECEIVE INTERRUPTS DON'T HAPPEN AT PRIORITY HIGHER THAN 3
3384          ;
3385 014460 012711 015060 000002      MOV     #202$,2(R1)        ;POINT RECEIVE VECTOR TO PROGRAM AREA
3386 014464 012761 000340          MOV     #340,2(R1)        ;AT PRIORITY 7
3387 014472 105764 000004          TSTB   4(R4)             ;TRANSMITTER READY
3388 014476 100410          bmi    11$              ;BPL     11$           ;IF NOT, WAIT
3389 014500 004737 030610          jsr     pc,delay
3390 014504 105764 000004          tstb   4(r4)
3391 014510 100403          bmi    11$
3392 014512 104076          error  +76
3393 014514 005037 172100          clr     mer
3394
3395 014520 052714 000100 11$:     BIS     #BIT06,(R4)      ;SET INTERRUPT ENABLE BIT IN RCSR
3396 014524 012764 000000 000006      MOV     #NULL,6(R4)      ;TRY TO TRANSMIT NULL

```

G7

TEST RESET AND INTERRUPT ENABLE BITS

```

3397 014532 012702 000340      MOV      #340,R2      ;SET PRIORITY TO 7
3398 014536 000402              BR       13$         ;GO WAIT IN CASE OF INTERRUPTS
3399 014540 162702 000040      SUB      #40,R2      ;LOWER PRIORITY LEVEL
3400 014544 052764 000004 000004 12$:  BIS      #BIT02,4(R4) ;SET LOOP BACK MODE
3401 014552 106402              MTPS     R2          ;SET PRIORITY
3402 014554 004737 030610      JSR      PC,DELAY    ;TIMER ROUTINE ;MC
3403 014560 005737 002356      tst      rcount      ;
3404 014564 001406              beq      16$         ;14$: SOB      R3,14$ ;TIME DELAY ;WAIT FOR IN

TERRUPTS
3405 ;BR      16$         ;IF INTERRUPTS DIDN'
T HAPPENED, BRANCH
3406 014566 042764 000004 000004 15$:  BIC      #BIT02,4(R4) ;CLEAR LOOP BACK MODE
3407 014574 104021              ERROR    +21         ;INTERRUPTS HAPPEN AT WRONG PRIORITY
3408 014576 005037 172100      clr      mer
3409 ;CMP     (SP)+,(SP)+ ;CLEAN UP THE STACK
3410 014602 022702 000200      16$:  CMP      #200,R2 ;AT PRIORITY 4?
3411 014606 001354              BNE      12$         ;IF NOT LAST ONE, CONTINUE
3412 ;
3413 ; CLEAN UP BEFORE NEXT TEST
3414 ;
3415 014610 106427 000340      MTPS     #340        ;RESTORE PRIORITY 7
3416 014614 042714 000100      BIC      #BIT06,(R4) ;CLEAR INTERRUPT ENABLE BIT
3417 ;MOV     #400,R2 ;STALL DELAY
3418 014620 105714      17$:  TSTB     (R4)      ;RECEIVE READY?
3419 014622 100402              BMI      18$         ;STOP WAITING, IF SO
3420 014624 004737 030610      jsr      pc,delay    ;SOB     R2,17$ ;OTHERWISE, STAY IN

THE LOOP
3421 ;
3422 014630 005764 000002      18$:  TST      2(R4)    ;READ CHARACTER TRANSMITTED
3423 014634 042764 000004 000004 20$:  BIC      #BIT02,4(R4) ;CLEAR LOOP BACK MODE
3424 014642 012701 000300      MOV      #300,R1    ; POINT TO SLU #1 VECTOR
3425 014646 012704 176500      MOV      #RCSR1,R4  ; POINT TO SLU #1 REGISTER
3426 014652 005305              DEC      R5
3427 014654 001402              BEQ      19$
3428 014656 000137 014252      JMP      1$
3429 014662              19$:
3430 ;
3431 ;
3432 ;
3433 ;
3434 ; loop back connector test
3435 ;
3436 ;
3437 ;
3438 014662 012702 000002      mov      #2,r2      ;counter
3439 ;
3440 014666 032702 000001      110$: bit      #1,r2    ; odd or even ??
3441 014672 001005              bne      112$        ;nope even
3442 014674 022737 000001 002362  cmp      #1,lobbak  ;yep odd. connector installed ???
3443 014702 001031              bne      120$        ;no it's not
3444 014704 000403              br       114$
3445 ;
3446 014706 052737 000004 176504 112$: bis      #bit02,xcsr1 ;set the maintenance bit in the
3447 ;csr to do the internal loop back
3448 ;
3449 014714 012703 003004      114$: mov      #tchar,r3 ;pointer of .byte : -1, 21, -1, 0
3450 014720 004737 030610      jsr      pc,delay    ;allow last character from terminal
3451 014724 113700 176502      movb     @#rbuf1,r0 ;suck any extra character
3452 ;
3453 014730 112300      116$: movb     (r3)+,r0 ;put the first character in r0

```

H7

TEST RESET AND INTERRUPT ENABLE BITS

```

3454 014732 001415          beq    120$          ;done with this pass of th's test
3455 014734 110037 176506    movb   r0,xbuf1     ;transmit
3456 014740 004737 030610    jsr    pc,delay     ;allow some time
3457 014744 105737 176500    tstb  @#rcsr1       ;receiver ready ??
3458 014750 100401          bmi    118$
3459 014752 104077          error  +77          ;no character received
3460
3461 014754 123737 176502 176506 118$:  cmpb  @#rbuf1,@#xbuf1
3462 014762 001762          beq    116$
3463 014764 104100          error  +100        ;wrong character received
3464
3465 014766          120$:
3466 014766 077241          sob   r2,110$
3467
3468 014770          500$:
3469 014770 013737 003016 000004    MOV   SAVTIM,@#4
3470 014776 000440          BR    TST2         ;;GO TO THE NEXT TEST
3471
3472          ;
3473          ; DEBUG PURPOSES
3474          ;
3475
3476 015000 106427 000340    100$:  MTPS   #340        ;RESTORE PRIORITY 7
3477 015004 042714 000100    BIC   #BIT06,(R4)  ;CLEAR INTERRUPT ENABLE BIT
3478          ;
3479 015010 105714          170$:  TSTB   (R4)        ;MOV #400,R2 ;STALL DELAY
3480 015012 100403          BMI   180$        ;RECEIVE READY?
3481 015014 004737 030610    jsr    pc,delay     ;STOP WAITING, IF SO
3482          ;SOB R2,170$ ;OTHERWISE, STAY IN
3483          ;
3484          ;
3485          ;
3486          ;
3487          ;
3488          ;
3489          ; INTERRUPT SERVICE ROUTINES
3490          ;
3491
3492 015036 005737 002360    200$:  tst    TCOUNT      ;INCREMENT TRANS. COUNTER ;MC
3493 015042 001003          bne   61$
3494 015044 042764 000100 000004    bic   #bit6,4(r4)
3495 015052 005237 002360    61$:   inc    tcount
3496 015056 000002          RTI
3497          ;RETURN FROM XMIT INTERRUPT ;MC
3498 015060 005737 002356    202$:  tst    RCOUNT      ;INCREMENT RECEIVER COUNTER;MC
3499 015064 001002          bne   81$
3500 015066 042714 000100    bic   #bit6,(r4)
3501 015072 005237 002356    81$:   inc    rcount
3502 015076 000002          RTI
3503          ;RETURN FROM RECEIVER INTERRUPT
3504
3505

```

TEST INTERRUPT PRIORITY FOR SLU

```

3507 .SBTTL TEST INTERRUPT PRIORITY FOR SLU
3508 ;CHECK THAT INTERRUPTS HAPPEN AT PRIORITY 3 AND THAT THEY CLEAR
3509 ;RCSR<06> AND XCSR<06>.
3510 ;
3511 ;ROUTINE TEST
3512 ;. LET 60=#ADDRESS OF LEGAL RINTERRUPT
3513 ;. LET 64=#ADDRESS OF LEGAL XINTERRUPT
3514 ;. LET XCSR<02>=#1
3515 ;. SET PRIORITY TO #3
3516 ;. WAIT FOR XINTERRUPT=#3
3517 ;. IF XCSR<07> EQ #1 THEN
3518 ;. ERROR
3519 ;. ENDF
3520 ;. WAIT FOR RINTERRUPT=#3
3521 ;. IF RCSR<07> EQ #0 THEN
3522 ;. ERROR
3523 ;. ENDF
3524 ;. LET XCSR<02>=#0
3525 ;. SET PRIORITY TO NORMAL
3526 ;ENDROUTINE
3527 ;
3528 ;ROUTINE LEGAL_XINTERRUPT
3529 ;. LET XBUF=#CHARACTER
3530 ;. INCREMENT XINTERRUPT
3531 ;ENDROUTINE
3532 ;
3533 ;ROUTINE LEGAL_RINTERRUPT
3534 ;. READ RCSR
3535 ;. INCREMENT RINTERRUPT
3536 ;ENDROUTINE
3537 ;
3538 ;*****
3539 015100 000004 TST22: SCOPE
3540 015102 122737 000001 001220 CMPB #APTENV,$ENV ;ARE WE IN APT MODE?
3541 015112 005737 001206 BNE 1002$ ;IF NOT: DO THIS TEST
3542 015116 001404 TST $PASS ;FIRST PASS??
3543 015120 012705 000002 BEQ 1002$ ;IF YES, DO IT
3544 015124 000137 015454 MOV #2,R5 ; LOOP COUNT OF 1
3545 015130 032777 010000 164002 1002$: JMP 20$ ; GO SET UP POINTERS TO SLU 1
3546 015136 001431 015146 BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
3547 015140 104401 015146 BEQ 1000$ ; THEN TYPE TEST TRACE
015144 000424 TYPE ,30047$ ;;TYPE ASCIZ STRING
3548 015216 004737 030610 BR 30046$ ;;GET OVER THE ASCIZ
3549 ;:30047$: .ASCIZ <15><12>/TEST 22 SLU INTERRUPT PRIORITY TEST/
3550 1000$: ;:30046$: jsr pc,delay ;:MOV #100000,R1
3551 ;:1001$: SOB R1,1001$ ; WAIT FOR L
3552 ;
3553 ;
3554 ; GET READY FOR INTERRUPTS
3555 ;
3556 015222 100$: MOV #60,R1 ;
3557 015222 012701 000060 MOV #RCSR,R4 ;
3558 015226 012704 177560 MOV #2,R5 ;
3559 015232 012705 000002

```

TEST INTERRUPT PRIORITY FOR SLU

```

3560
3561 015236          10$:
3562 015236 011146      MOV      (R1), (SP)          ;SAVE PREVIOUS VECTOR ,MC
3563 015240 016146 000004  MOV      4(R1), (SP)      ;" " " " " "
3564 015244 012711 015634      MOV      #8$, (R1)        ;STORE RECEIVER VECTOR
3565 015250 012761 015612 000004  MOV      #6$, 4(R1)        ;STORE TRANSMITTER VECTOR
3566 015256 012761 000340 000002  MOV      #340, 2(R1)       ;AT PRIORITY 7
3567 015264 012761 000340 000006  MOV      #340, 6(R1)       ;FOR RECEIVER AND TRANSMITTER
3568 015272 052764 000004 000004  BIS      #BIT02, 4(R4)     ;SET LOOP BACK MODE
3569 015300 012701 000140      MOV      #140, R1         ;DELAY FOR UNEXPECTED CHARACTERS
3570 015304 105714          1$:      TSTB     (R4)              ;RECEIVER READY?
3571 015306 100401          BMI      2$                ;IF YES, BRANCH
3572 015310 077103          SOB      R1, 1$           ;OTHERWISE, WAIT JUST IN CASE
3573 015312 005764 000002          2$:      TST      2(R4)        ;READ RECEIVER
3574
3575      ; SET PRIORITIES AND XMIT INTERRUPTS
3576
3577 015316 012702 000200          MOV      #200, R2         ;START WITH PRIORITY 3
3578
3579 015322 162702 000040          3$:      SUB      #40, R2        ;LOWER PRIORITY
3580 015326 005037 002360          CLR      TCOUNT        ;CLEAR TRANSMITTER COUNTER ;MC
3581 015332 005037 002356          CLR      RCOUNT        ;CLEAR RECEIVER COUNTER ;MC
3582
3583      ; TRANSMITTER INTERRUPT HERE
3584
3585
3586 015336 106402          4$:      MTPS     R2            ;TRY TO DO AT LOWER PRIORITY
3587 015340 052764 000100 000004  BIS      #BIT06, 4(R4)    ;LOOP BACK & INTERRUPT ENABLE
3588 015346 004737 030610          JSR      PC, DELAY        ;WAIT DELAY FOR INTERRUPT ;MC
3589 015352 042764 000100 000004  BIC      #BIT06, 4(R4)    ;CLEAR INTERRUPT ENABLE
3590 015360 022737 000001 002360  CMP      #1, TCOUNT     ;ANY INTERRUPT HAPPENED ? ;MC
3591 015366 101041          BHI      5$                ;NO XMIT INTERRUPT ;MC
3592 015370 103452          BLO      200$             ;TOO MANY XMIT INTERRUPTS ;MC
3593
3594      ; RECEIVER INTERRUPT HERE
3595
3596
3597
3598 015372 052714 000100          BIS      #BIT06, (R4)     ;SET RECEIVE INTERRUPT
3599 015376 012764 000000 000006  MOV      #NULL, 6(R4)     ;TRANSMIT NULL
3600 015404 004737 030610          JSR      PC, DELAY        ;WAIT DELAY FOR INTERRUPT ;MC
3601 015410 042714 000100          BIC      #BIT06, (R4)    ;CLEAR INTERRUPT ENABLE
3602 015414 022737 000001 002356  CMP      #1, RCOUNT     ;ONE INTERRUPT HAPPENED ? ;MC
3603 015422 101047          BHI      7$                ;NO RECEIVER INTERRUPTS ;MC
3604 015424 103460          BLO      201$             ;2 MANY RECEIVER INTERRUPTS ;MC
3605
3606      ; IF DONE GET OUT, IF NOT LOOP
3607
3608
3609 015426 005764 000002          9$:      TST      2(R4)        ;READ RECEIVER BUFFER
3610 015432 005702          TST      R2              ;PRIORITY 0
3611 015434 001332          BNE      3$                ;IF NOT YET, CONTINUE
3612 015436 106427 000340          MTPS     #340            ;RAISE PRIORITY TO 7
3613 015442 005064 000004          CLR      4(R4)           ;CLEAR XCSR
3614 015446 012661 000004          MOV      (SP)+, 4(R1)     ;GET PREVIOUS VECTOR BACK
3615 015452 012611          MOV      (SP)+, (R1)     ;" " " " " "
3616 015454 012704 176500          20$:     MOV      #RCSR1, R4      ;
  
```


TEST INTERRUPT PRIORITY FOR SLU

```

3617 015460 012701 000300      MOV      #300,R1      ;
3618 015464 005305      DEC      R5          ;
3619 015466 001263      BNE     10$         ;
3620 015470 000471      BR      TST23       ;;IF ERROR, EXIT TEST
3621
3622
3623      ;ERROR ROUTINES
3624      ;
3625
3626 015472 012661 000004      5$:      MOV      (SP)+,4(R1)      ;GET PREVIOUS VECTOR BACK
3627 015476 012611          MOV      (SP)+,(R1)        ;" " " " "
3628 015500 042764 000104 000004      BIC     #BIT02!BIT06,4(R4) ;CLEAR LOOP BACK BIT
3629 015506 104070          ERROR   +70             ;NO XMIT INTERRUPTS
3630 015510 005037 172100      clr     mer
3631 015514 000457          BR      TST23       ;;IF ERROR, EXIT TEST
3632
3633 015516 012661 000004      200$:   MOV      (SP)+,4(R1)      ;GET PREVIOUS VECTOR BACK
3634 015522 012611          MOV      (SP)+,(R1)        ;" " " " "
3635 015524 042764 000104 000004      BIC     #BIT02!BIT06,4(R4) ;CLEAR LOOP BACK MODE BIT
3636 015532 104074          ERROR   +74             ;2 MANY XMIT INTERRUPTS ;MC
3637 015534 005037 172100      clr     mer
3638 015540 000445          BR      TST23       ;;IF ERROR, EXIT TEST
3639
3640 015542 012661 000004      7$:      MOV      (SP)+,4(R1)      ;GET PREVIOUS VECTOR BACK
3641 015546 012611          MOV      (SP)+,(R1)        ;" " " " "
3642 015550 042764 000104 000004      BIC     #BIT02!BIT06,4(R4) ;CLEAR LOOP BACK MODE BIT
3643 015556 104071          ERROR   +71             ;NO RECEIVE INTERRUPTS
3644 015560 005037 172100      clr     mer
3645 015564 000433          BR      TST23       ;;IF ERROR, EXIT TEST
3646
3647 015566 012661 000004      201$:   MOV      (SP)+,4(R1)      ;GET PREVIOUS VECTOR BACK
3648 015572 012611          MOV      (SP)+,(R1)        ;" " " " "
3649 015574 042764 000104 000004      BIC     #BIT02!BIT06,4(R4) ;CLEAR LOOP BACK MODE BIT
3650 015602 104075          ERROR   +75             ;2 MANY RECEIVER INTERRUPTS
3651 015604 005037 172100      clr     mer
3652 015610 000421          BR      TST23       ;;IF ERROR, EXIT TEST
3653
3654      ;
3655      ;INTERRUPT SERVICE ROUTINES
3656      ;
3657
3658 015612 005737 002360      6$:      tst      TCOUNT        ;INCREMENT TRANS. COUNTER ;MC
3659 015616 001003          bne     61$         ;
3660 015620 042764 000100 000004      bic     #bit6,4(r4)
3661 015626 005237 002360      61$:     inc     tcount
3662 015632 000002          RTI                    ;RETURN FROM XMIT INTERRUPT ;MC
3663
3664 015634 005737 002356      8$:      tst      RCOUNT        ;INCREMENT RECEIVER COUNTER;MC
3665 015640 001002          bne     81$         ;
3666 015642 042714 000100          bic     #bit6,(r4)
3667 015646 005237 002356      81$:     inc     rcount
3668 015652 000002          RTI                    ;RETURN FROM RECEIVER INTERRUPT
3669
3670

```

L7

TEST BREAK CONDITION

3672
3673
3674
3675
3676
3677
3678
3679
3680
3681
3682
3683
3684
3685
3686
3687
3688
3689
3690
3691
3692
3693
3694
3695
3696
3697
3698
3699
3700
3701
3702
3703
3704
3705
3706
3707
3708
3709

```

.SBTTL TEST BREAK CONDITION
; SLU #1 IS TESTED SINCE SLU #0 IS BEING USED AS THE CONSOLE
; CHECK THAT SENDING BREAK CAUSES FRAMING ERROR.
;RCSR <15> ERROR
; <13> FRAMING ERROR
; <11> RECEIVED BREAK
;XCSR <00> TRANSMIT BREAK
;ROUTINE TEST
;.. LET XCSR<02>=#1
;.. LET XCSR<00>=#1
;.. WAIT FOR RCSR<07>=#1
;.. IF RBUF<15!13!11> NE #1 THEN
;.. ERROR (ERROR, FRAMING ERROR, RECEIVE BREAK NE 1)
;.. ENDIF
;.. LET XCSR<00>=#0
;.. IF XCSR<00> NE #0 THEN
;.. ERROR (XCSR<00> DOES NOT GO LOW)
;.. ENDIF
;.. WAIT FOR XCSR<07>=#1
;.. LET XBUF=#NULL (SEND NULL CHARACTER TO SEE ERROR CLEARED)
;.. WAIT FOR RCSR<07>=#1
;.. IF RBUF<15!13!11> NE #0 THEN
;.. ERROR
;.. ENDIF
;.. LET XCSR<00>=#1
;.. EXECUTE "RESET"
;.. IF XCSR<00> NE #0 THEN
;.. ERROR
;.. ENDIF
;.. LET XCSR<02>=#0
;ENDROUTINE

```

015654 000004
3710 015656 032777 010000 163254
3711 015664 001426
3712 015666 104401 015674
015672 000423
015742
3713 015742
3714
3715
3716
3717
3718
3719 015742 052737 000004 176504
3720 015750 052737 000001 176504
3721 015756 032737 000001 176504
3722 015764 001003
3723 015766 104027
3724 015770 005037 172100

```

;*****
TST23: SCOPE
BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
BEQ 1000$ ; THEN TYPE TEST TRACE
TYPE ,30049$ ;;TYPE ASCIZ STRING
BR 30048$ ;;GET OVER THE ASCIZ
;;30049$: .ASCIZ <15><12>/TEST 23 - SLU BREAK CONDITION TEST/
30048$:
1000$:
; SEND BREAK AND CHECK ERROR BITS IN RBUF
;
1$: BIS #BIT02,XCSR1 ;TRANSMIT IN LOOP BACK
BIS #BIT00,XCSR1 ;SET SEND BREAK BIT
BIT #BIT00,XCSR1 ;GOT SET OK?
BNE 2$ ;IF YES, BRANCH
ERROR +27 ;WRITING 1 TO XCSR<0>
clr mer

```

TEST - BREAK CONDITION

```

3725 015774 012701 000100      2$:  MOV    #100,R1          ;STALL DELAY
3726 016000 105737 176500      4$:  TSTB   RCSR1          ;RECEIVER READY?
3727 016004 100401              BMI    5$              ;IF YES, BRANCH
3728 016006 077104              SOB   R1,4$           ;WAIT JUST IN CASE OF A CHARACTER
3729 016010 005737 176502      5$:  TST    RBUF1          ;READ A CHARACTER
3730 016014 052737 000001 176504  BIS    #BIT00,XCSR1   ;TRANSMIT BREAK
3731 016022 004737 030610      JSR    pc,delay       ;      ; MOV #1000,R1
;ANOTHER DELAY TO GET BREAK
3732
;WAIT A WHILE
3733 016026 105737 176500      TSTB   RCSR1          ;RECEIVER READY?
3734 016032 100410              BMI    7$              ;BPL    7$              .IF
NOT, WAIT
3735 016034 004737 030610      JSR    pc,delay
3736 016040 105737 176500      TSTB   rcsr1
3737 016044 100403              BMI    7$
3738 016046 104076              error  +76
3739 016050 005037 172100      CLR    mer
3740
3741 016054 013737 176502 001126 7$:  MOV    RBUF1,$BDDAT   ;STORE WHATEVER RECEIVED
3742 016062 022737 124000 001126  CMP    #BIT15!BIT13!BIT11,$BDDAT ;ALL ERROR BITS SET?
3743 016070 001407              BEQ    8$              ;IF YES, BRANCH
3744 016072 042737 000004 176504  BIC    #BIT02,XCSR1   ;RESET TO ENABLE SLU
3745 016100 104025              ERROR  +25             ;BREAK DOES NOT CAUSE ERRORS
3746 016102 005037 172100      CLR    mer
3747 016106 000467              BR     TST24           ;;EXIT
3748 016110 042737 000001 176504 8$:  BIC    #BIT00,XCSR1   ;CLEAR TRANSMIT BREAK
3749 016116 032737 000001 176504  BIT    #BIT00,XCSR1   ;GOT CLEARED OK?
3750 016124 001407              BEQ    9$              ;IF YES, BRANCH
3751 016126 042737 000004 176504  BIC    #BIT02,XCSR1   ;RESET TO ENABLE SLU
3752 016134 104027              ERROR  +27             ;ERROR WRITING 0 TO XCSR<0>
3753 016136 005037 172100      CLR    mer
3754 016142 000451              BR     TST24           ;;EXIT
3755
; CHECK THAT BREAK CONDITION IS CLEARED
3756
3757
3758 016144      9$::  MOV    SAVBR,BCSR     ;RESTORE BCSR
3759 016144 105737 176504      TSTB   XCSR1          ;XMIT READY?
3760 016150 100410              BMI    10$            ;BPL    10$            .IF
NOT, WAIT
3761 016152 004737 030610      JSR    pc,delay
3762 016156 105737 176504      TSTB   xcsr1
3763 016162 100403              BMI    10$
3764 016164 104076              error  +76
3765 016166 005037 172100      CLR    mer
3766
3767 016172 012737 000177 176506 10$:  MOV    #177,XBUF1     ;TRY TO TRANSM  DELETE
3768 016200 105737 176500      TSTB   RCSR1          ;RECEIVER READY?
3769 016204 100410              BMI    11$            ;BPL    11$            .IF
NOT, WAIT
3770 016206 004737 030610      JSR    pc,delay
3771 016212 105737 176500      TSTB   rcsr1
3772 016216 100403              BMI    11$
3773 016220 104076              error  +76
3774 016222 005037 172100      CLR    mer
3775
3776 016226 013737 176502 001126 11$:  MOV    RBUF1,$BDDAT   ;STORE RECEIVE BUFFER
3777 016234 032737 124000 001126  BIT    #BIT15!BIT13!BIT11,$BDDAT ;ERRORS CLEARED?
3778 016242 001406              BEQ    12$            ;IF YES, BRANCH
3779 016244 042737 000004 176504  BIC    #BIT02,XCSR1   ;RESET TO ENABLE SLU
3780 016252 104025              ERROR  +25             ;BREAK NOT CLEARED ON NEXT CHARACTER
3781 016254 005037 172100      CLR    mer

```

N7

TEST BREAK CONDITION

3782 016260 042737 000004 176504 124: BIC #BIT02,XCSR1 ;CLEAR LOOP BACK MODE
3783
3784

B5

TES* OVERRUN CONDITION

```

3786 .SBTTL TEST OVERRUN CONDITION
3787 ;CHECK OVERRUN CONDITION
3788
3789 ;RCSR <14> OVERRUN ERROR
3790
3791 ; SLU #1 IS TESTED SINCE SLU #0 IS BEING USED AS THE CONSOLE
3792
3793 ;ROUTINE TEST
3794 .. LET XCSR<02>=#1 (LOOPBACK MODE)
3795 .. WAIT FOR XCSR<07>=#1
3796 .. LET XBUF=#252
3797 .. WAIT FOR XCSR<07>=#1
3798 .. LF, XBUF=#125 (SEND THE 2ND W/O READING THE 1ST CHARACTER)
3799 .. WAIT FOR RCSR<07>=#1
3800 .. STALL FOR LOWEST BAUD RATE TO GET 2ND CHARACTER
3801 .. IF LOW BYTE OF RBUF NE #125 THEN
3802 .. ERROR (1ST CHARACTER WASN'T OVERRUN)
3803 ..
3804 .. ENDF
3805 .. IF RBUF<15!14> NE #1 THEN
3806 .. ERROR (NO OVERRUN BIT SET)
3807 ..
3808 .. ENDF
3809 .. WAIT FOR XCSR<07>=#1
3810 .. LET XBUF=#NULL
3811 .. WAIT FOR RCSR<07>=#1
3812 .. IF RBUF<15!14> NE #0 THEN
3813 .. ERROR (WASN'T CLEARED ON THE NEXT CHARACTER RECEIVED)
3814 ..
3815 .. ENDF
3816 .. LET XCSR<02>=#0
3817 .. ENDRoutine
3818
3819 *****
3820 TST24: SCOPE
3821 BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
3822 BEQ 100$ ; THEN TYPE TEST TRACE
3823 TYPE ,30051$ ;;TYPE ASCIZ STRING
3824 BR 30050$ ;;GET OVER THE ASCIZ
3825 ;;30051$: .ASCIZ <15><12>/TEST 24 SLU OVERRUN CONDITION TEST/
3826 30050$:
3827 MOV #100000,R1
3828 1001$: SOB R1,1001$ ; WAIT FOR LAST CHARACTER
3829 100$: BIS #BIT02,XCSR1 ;SET LOOP BACK MODE
3830 tstb xcsr1 ;1$: TSTB XCSR1
3831 bmi 1$ ;BPL 1$ .IF
3832 NOT WAIT
3833 jsr pc,delay
3834 tstb xcsr1
3835 bmi 1$
3836 error +76
3837 clr mer
3838 176506 1$: MOV #21,XBUF1 ;TRANSMIT A CHARACTER
3839 tstb rcsr1 ;2$: TSTB RCSR1
3840 bmi 2$ ;BPL 2$ .IF
3841 NOT WAIT
3842 jsr pc,delay
3843 tstb rcsr1
3844 bmi 2$
3845 error +76
3846 clr mer
3847
3848
3849
3850
3851
3852
3853
3854
3855
3856
3857
3858

```

TEST OVERRUN CONDITION

```

3839 016454          2$:          tstb   xcsr1          ;3$:   TSTB   XCSR1
3840 016454 105737 176504          bmi   3$          ;BPL   3$          ;IF
;READY TO TRANSMIT?
3841 016460 100410          jsr   pc, delay
NOT WAIT          tstb   xcsr1
3842 016462 004737 030610          bmi   3$
3843 016466 105737 176504          error +76
3844 016472 100403          clr   mer
3845 016474 104076          3$:          MOV    #177,XBUF1          ;TRANSMIT THE 2ND CHARACTER
3846 016476 005037 172100          ;MOV    #175000,R3          ;STA
3847 016502          jsr   pc, delay
3848 016502 012737 000177 176506          ;4$:   SOB    R3,4$
3849          ;STORE RECEIVED DATA
LL FOR THE 2ND CHARACTER $$$          ;EXPETED PATTERN
3850 016510 004737 030610          MOV    RBUF1,$BDDAT          ;2ND CHARACTER RECEIVED?
;WAIT A WHILE          MOV    #140177,$GDDAT          ;IF YES, BRANCH
3851 016514 013737 176502 001126          CMPB  #177,$BDDAT          ;RESET TO ENABLE SLU
3852 016522 012737 140177 001124          BEQ    5$          ;2ND CHARACTER DIDN'T OVERRUN 1ST
3853 016530 122737 000177 001126          BIC   #BIT02,XCSR1
3854 016536 001406          ERROR +31
3855 016540 042737 000004 176504          clr   mer
3856 016546 104031          5$:          CMPB  #BIT7!BIT6,$BDDAT+1          ;OVERRUN ERROR BITS SET?
3857 016550 005037 172100          BEQ    6$          ;IF YES, BRANCH
3858 016554 122737 000300 001127          CLR   XCSR1
3859 016562 001406          ERROR +32
3860 016564 005037 176504          clr   mer
3861 016570 104032          BR    TST25          ;;EXIT
3862 016572 005037 172100          ; SEND NEXT CHARACTER TO CLEAR OVERRUN CONDITIONS
3863 016576 000450          ;
3864          ;
3865          ;
3866          ;
3867 016600          6$:          tstb   xcsr1          ;TSTB   XCSR1          ;TRA
3868 016600 105737 176504          bmi   7$          ;BPL   6$          ;IF
NSMITTER READY?
3869 016604 100410          jsr   pc, delay
NOT BRANCH AND WAIT          tstb   xcsr1
3870 016606 004737 030610          bmi   7$
3871 016612 105737 176504          error +76
3872 016616 100403          clr   mer
3873 016620 104076          7$:          MOV    #NULL,XBUF1          ;TRANSMIT NULL CHARACTER
3874 016622 005037 172100          tstb   rcsr1          ;7$:   TSTB   RCSR1
3875 016626          bmi   8$          ;BPL   7$          ;IF
;RECEIVER READY?
3876          ;
3877 016626 012737 000000 176506          ;
3878 016634 105737 176500          ;
3879 016640 100410          ;
NOT BRANCH AND WAIT          ;
3880 016642 004737 030610          jsr   pc, delay
3881 016646 105737 176500          tstb   rcsr1
3882 016652 100403          bmi   8$
3883 016654 104076          error +76
3884 016656 005037 172100          clr   mer
3885          ;
3886 016662 032737 140000 176502 8$:          BIT   #BIT15!BIT14,RBUF1          ;ANY ERRORS SET?
3887 016670 001406          BEQ    9$          ;IF NOT, BRANCH
3888 016672 042737 000004 176504          BIC   #BIT02,XCSR1          ;RESET TO ENABLE SLU
3889 016700 104033          ERROR +33          ;OVERRUN NOT CLEARED ON NEXT CHAR.
3890 016702 005037 172100          clr   mer
3891 016706 042737 000004 176504 9$:          BIC   #BIT02,XCSR1          ;CLEAR LOOP BACK MODE BIT
3892          ;
3893 016714          SLEND:          ;LAST SLU TEST
3894 016714 000401          BR    TST25          ;GO TO THE NEXT TEST
3895

```

D8

TEST OVERRUN CONDITION

3896 016716 000240
3897
3898

NOSLU: NOP

;WE SKIPPED ALL SLU TEST IN APT MODE ON MORE THAN 1 PASS

E8

TEST LED'S ON

```

3900 .SBTTL TEST - LED'S ON
3901 ;LED'S ON
3902 ;THIS TEST WILL INITIALIZE BDR TO CONTAIN A ROTATING PATTERN
3903 ;DISPLAYED IN LED'S.
3904 ;
3905 ;ROUTINE TEST
3906 ;. WHILE A KEY NOT RECEIVED FROM KEYBOARD DO
3907 ;. STALL ALLOWING TIME TO SEE PATTERN
3908 ;. ROTATE LEFT TO LIGHT UP NEXT LED'S
3909 ;. ENDDO
3910 ;ENDROUTINE
3911
3912
3913
3914
3915 ;*****
3916 016720 000004 TST25: SCOPE
3917 016722 032777 010000 162210 BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
3918 016730 001423 BEQ 1000$ ; THEN TYPE TEST TRACE
3918 016732 104401 016740 TYPE ,30053$ ;;TYPE ASCIZ STRING
3918 016736 000420 BR 30052$ ;;GET OVER THE ASCIZ
;;30053$: .ASCIZ <15><12>/TEST 25 - KDJ11-DA LED TESTS/
30052$:
3919 017000
3920 017000 1000$:
3921 017000 005005 CLR R5 ;FLAG IN NO INTERRUPT MODE
3922 017002 032737 000001 000052 BIT #BIT00,@#52 ;IF RUNNING IN CHAIN MODE
3923 017010 001413 BEQ 1$ ;SKIP PRINTOUTS
3924 017012 005737 001206 TST $PASS ;1ST PASS?
3925 017016 001010 BNE 1$ ;IF NOT, SKIP PRINTOUTS
3926 017020 122737 000001 001220 CMPB #APTENV,$ENV ;APT MODE?
3927 017026 001404 BEQ 1$ ;YES, SKIP PRINTOUT'S
3928 017030 005105 COM R5 ;CLEAR FLAG IN INTERRUPT MODE
3929 017032 012737 000005 002354 MOV #5,LEDCNT ;DO PATTERN 5 TIMES ;MC
3930
3931 017040 012704 000020 1$: MOV #20,R4 ;FOR EACH LOOP
3932 017044 012701 000000 MOV #0,R1 ;START WITH 1
3933 017050 110137 177520 2$: MOVB R1,@#NATREG ;TURN OFF FIRST LED
3934 017054 012703 000004 MOV #4,R3 ;STALL DELAY
3935 017060 012702 177777 3$: MOV #177777,R2 ;STALL DELAY
3936 017064 077201 4$: SOB R2,4$ ;WAIT A WHILE
3937 017066 077304 SOB R3,3$ ;WAIT A WHILE
3938 017070 000241 CLC ;
3939
3940
3941
3942 017072 7$:
3943 017072 005201 INC R1 ; 1.....19 OCTAL ;MC
3944 017074 077413 SOB R4,2$ ; 20.....0 OCTAL ;MC
3945 017076 005705 TST R5 ;RUNNING IN INTERACTIVE MODE?
3946 017100 001411 BEQ 6$ ;IF NOT, EXIT
3947 017102 104401 001170 TYPE , $BELL ;
3948 017106 005337 002354 DEC LEDCNT ;REPEAT PATTERN 5 TIMES
3949 017112 001352 BNE 1$
3950
3951
3952 017114 005737 177562 5$: TST RBUF ;READ BUFFER

```


F8

TEST - LED'S ON

```
3953 017120 062706 000004      ADD      #4,SP      ;ADJUST STACK
3954 017124 112737 000000 177520 6$:  MOVB     #0,NATREG  ;NO MORE
3955 017132 000400          BR       TST26      ;;EXIT TEST
3956
```

TEST DIFFERENT LEVELS OF INTERRUPTS

3958
3959
3960
3961
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987

```

.SBTTL TEST DIFFERENT LEVELS OF INTERRUPTS
;DIFFERENT LEVELS OF INTERRUPTS
;THIS TEST WILL PROGRAM Q22 BUS EXERCISER TO INTERRUPT AT DIFFERENT
;LEVELS. ARBITRATION BETWEEN DIFFERENT LEVELS OF INTERRUPTS AND
;PIRQ'S WILL BE TESTED.
;
;CHECK DIFFERENT LEVELS OF INTERRUPTS.
;ROUTINE TEST
;.. SET VECTOR TO INTERRUPT DMA
;.. FOR INTERRUPTS FROM 4 TO 7 DO
;.. . ENABLE INTERRUPTS
;.. . SET PRIORITY=INTERUPT
;.. . IF INTERRUPT FLAG SET THEN
;.. . . ERROR
;.. . . ENDF
;.. . . ENABLE INTERRUPTS
;.. . . SET PRIORITY-INTERRUPT 1
;.. . . IF INTERRUPT FLAG NOTSET THEN
;.. . . . ERROR
;.. . . . ENDF
;.. . . LET INTERRUPT_DMA=0
;.. ENDDO
;ENDROUTINE
;ROUTINE INTERUPT_DMA
;.. LET INTERRUPT_FLAG=1
;RETURN
;ENDROUTINE

```

017134 000004

3988
3989
3990
3991
3992
3993
3994

017136 005737 002334
017142 001534
017144 032777 010000 161766
017152 001444
017154 104401 017162
017160 000441

```

;*****
TST26: SCOPE
; BIT #BIT07,@#52 ;UFD MODE?
; SKIP NE,<IF SO, EXIT TEST>
; TST CSR1 ;AT LEAST ONE Q22BE FOUND?
; BEQ TST27 ;; IF NOT, EXIT TEST
; BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
; BEQ 1000$ ; THEN TYPE TEST TRACE
; TYPE ,30055$ ;;TYPE ASCIZ STRING
; BR 30054$ ;;GET OVER THE ASCIZ
;;30055$: .ASCIZ <15><12>/TEST 26 - ARBITRATION BETWEEN CPU PRIORITY AND Q BUS INTERR

```

UPTS/

017264
3995 017264
3996
3997
3998
3999 017264 013703 002334
4000 017270 012777 017340 163050
4001 017276 012777 000340 163044
4002 017304 012700 003002
4003 017310 012701 000340
4004
4005
4006
4007 017314
4008 017314 106427 000200
4009 017320 004737 030026
4010 017324 012077 163006

```

30054$:
1000$:
; SETUP INITIAL PRIORITY TO 7
;
; MOV CSR1,R3 ;DO FOR FIRST FOUND Q22BE
; MOV #5$,@VQBE1 ;POINT INTERRUPT VECTOR TO PROGRAM
; MOV #340,@VQPR1 ;AT PRIORITY 7
; MOV #Q22EN,R0 ;START WITH 4 FOR INTERRUPTS
; MOV #340,R1 ;LOW BOUNDARY FOR NO INTERRUPTS
;
; CHECK THAT INTERRUPTS DON'T HAPPEN AT PRIORITY HIGHER THAN BR
;
2$:
4$: MTPS #200 ;SET PRIORITY NOT TO INTERRUPT
; JSR PC,Q22INT ;ENABLE INTERRUPTS
; MOV (R0)+,@CSR2 ;CLEAR GO BIT

```

TEST DIFFERENT LEVELS OF INTERRUPTS

```

4011 017330 000240      NOP
4012 017332 000240      NOP
4013 017334 000240      NOP
4014 017336 000405      BR          6$
4015 017340 104037      5$: ERROR    +37      ;IF NO INTERUPT, BRANCH
4016 017342 005037 172100  CLR          mer      ;INTERRUPTS HAPPEN
4017 017346 005726      TST         (SP)+
4018 017350 005726      TST         (SP)+      ;RESTORE STACK
4019 017352
4020
4021      ; INTERRUPT AT ALL LEVELS
4022
4023 017352 012777 017424 162766 INQ22: MOV     #5$,@VQBE1      ;POINT INTERRUPT VECTOR TO PROGRAM
4024 017360 012777 000340 162762  MOV     #340,@VQPR1      ;AT PRIORITY 7
4025 017366 012700 003002      MOV     #Q22EN,R0      ;START WITH 4 FOR INTERRUPTS
4026
4027      ; CHECK THAT INTERRUPTS HAPPEN AT PRIORITY LOWER THAN BR
4028
4029 017372 106427 000140 4$: MTPS    #140      ;SET PRIORITY TO INTERRUPT
4030 017376 004737 030026      JSR     PC,Q22INT      ;ENABLE INTERRUPTS
4031 017402 011077 162730      MOV     (R0),@CSR2    ;CLEAR GO BIT
4032 017406 000240      NOP
4033 017410 000240      NOP
4034 017412 000240      NOP
4035 017414 104037      ERROR    +37      ;INTERRUPTS DON'T HAPPEN
4036 017416 005037 172100  CLR          mer
4037 017422 000402      BR          6$      ;DON T RESTORE STACK
4038 017424 005726      5$: TST     (SP)+      ;RESTORE STACK
4039 017426 005726      TST     (SP)+
4040 017430 106427 000340 6$: MTPS    #340      ;BACK TO 7
4041

```

TEST ARBITRATION BETWEEN PIRQ'S AND INTERRUPTS

```

4043 .SBTTL TEST ARBITRATION BETWEEN PIRQ'S AND INTERRUPTS
4044 ;CHECK PRIORITY ORDER BETWEEN PIRQ'S AND INTERRUPTS.
4045 ;ROUTINE TEST
4046 ;. IF UFD THEN
4047 ;. EXIT TEST
4048 ;. ENDF
4049 ;. DO FOR I FROM #6 DOWN TO #3
4050 ;. SET PRIORITY TO I
4051 ;. ENABLE INTERRUPT(I+1) AND PIRQ(I+1)
4052 ;. IF INTERRUPT(I+1) WAS BEFORE PIRQ(I+1) THEN
4053 ;. ERROR
4054 ;. ENDF
4055 ;. ENDDO
4056 ;ENDROUTINE
4057
4058 ;:*****
4059 017434 000004 TST27: SCOPE
4060 ; BIT #BIT07,@#52 ;UFD MODE?
4061 ; SKIP NE,<IF SO, EXIT TEST>
4061 017436 005737 002334 TST CSR1 ;AT LEAST ONE Q22BE FOUND?
4062 017442 001523 BEQ TST30 ;;IF NOT, EXIT TEST
4063 017444 032777 010000 161466 BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGIST
ER 4064 017452 001441 BEQ 1000$ ; THEN TYPE TEST TRACE
4065 017454 104401 017462 TYPE ,30057$ ;;TYPE ASCIZ STRING
017460 000436 BR 30056$ ;;GET OVER THE ASCIZ
;:30057$: .ASCIZ <15><12>/TEST 27 ARBITRATION BETWEEN PIRQ'S AND Q-BUS INTERRUPTS/
30056$:
1000$:
4066 017556 MOV #3$,@VQBE1 ;SETUP Q22BE VECTOR
4067 017556 012777 017656 162562 MOV #340,@VQPR1 ;AT PRIORITY 7
4068 017564 012777 000340 162556 MOV #4$,PIRQVEC ;SETUP PIRQ VECTOR
4069 017572 012737 017672 000240 MOV #340,PIRQVEC+2 ;AT PRIORITY 7
4070 017600 012737 000340 000242 MOV #Q22EN,R0 ;POINT THRU PRIORITIES FOR Q22BE
4071 017606 012700 003002 MOV #PIRQT,R4 ;POINTER THRU PIRQ'S
4072 017612 012704 017710 MOV CSR1,R3 ;DO FOR FIRST Q22BE
4073 017616 013703 002334 MOV #140,R2 ;START WITH CPU PRIORITY AT 7
4074 017622 012702 000140 MTPS #340 ;RAISE PRIORITY TO 7
4075 017626 106427 000340 2$: MOV (R4)+,PIRQ ;SET PRIORITY FOR PIRQ'S
4076 017632 012437 177772 JSR PC,Q22INT ;INITIALISE Q22BE TO INTERRUPT
4077 017636 004737 030026 MOV (R0)+,@CSR2 ;SET DONE BIT
4078 017642 012077 162470 MTPS R2 ;LOWER PRIORITY
4079 017646 106402 NOP
4080 017650 000240 NOP
4081 017652 000240 NOP
4082 017654 000240
4083 017656 104035 3$: ERROR +35 ;PIRQ'S DON'T TAKE OVER BIRQ S
4084 017660 005037 172100 clr mer ;CLEAN UP STACK
4085 017664 005726 TST (SP)+
4086 017666 005726 TST (SP)+
4087 017670 000402 BR 5$ ;BRANCH AROUND PIRQ INTERRUPT
4088 017672 005726 4$: TST (SP)+ ;CLEAN UP STACK
4089 017674 005726 TST (SP)+
4090 017676 005037 177772 5$: CLR PIRQ ;CLEAR ANY REQUESTS
4091 017702 005077 162430 CLR @CSR2 ;CLEAR JUST IN CASE
4092 017706 000401 BR TST30 ;;GO TO THE NEXT TEST
4093
4094 017710 010000 PIRQT: .WORD 10000 ;PIRQ'S 4

```

J8

TEST ARBITRATION BETWEEN PIRQ'S AND INTERRUPTS

```

4096          ;*****
          TST30: SCOPE
4097 017712 000004          CLR @#SRO          ;DISABLE MMU (DO NOT REMOVE !!!)
4098 017714 005037 177572          JMP $EOP          ;EXIT
4099          ;
4100 017724 123727 001220 000001 VIREOP: CMPB $ENV,#1          ; if not APT, don't worry about
4101 017732 001005          BNE 1$          ;
4102 017734 005737 003034          TST CCHPAS          ; maintain cache routine pascnt
4103 017740 001402          BEQ 1$
4104 017742 005337 003034          DEC CCHPAS
4105          ;
4106          ; This VIREOP ROUTINE to provide common End of Pass exit point
4107 017746 000205          1$: rts          r5
4108

```

GLOBAL ERROR MESSAGES

```

4110
4111 017750
4112 017750 015 012 113
      017753 104 112 061
      017756 061 055 104
      017761 101 040 103
      017764 120 125 040
      017767 104 111 101
      017772 107 116 117
      017775 123 124 111
      020000 103 040 055
      020003 040 103 117
      020006 113 104 104
      020011 101 060 015
      020014 012 012
4113 020016 123 127 111 .ASCII /SWITCH REGISTER SELECTION:/<12><12><15>
      020021 124 103 110
      020024 040 122 105
      020027 107 111 123
      020032 124 105 122
      020035 040 123 105
      020040 114 105 103
      020043 124 111 117
      020046 116 072 012
      020051 012 015
4114 020053 102 111 124 .ASCII /BIT NUMBER USE/<12><15>
      020056 040 116 125
      020061 115 102 105
      020064 122 011 011
      020067 011 011 125
      020072 123 105 012
      020075 015
4115 020076 055 055 055 .ASCII / ----- /<12><15>
      020101 055 055 055
      020104 055 055 055
      020107 055 011 011
      020112 011 055 055
      020115 055 055 055
      020120 055 055 055
      020123 055 055 055
      020126 055 055 055
      020131 055 055 055
      020134 055 055 012
      020137 015
4116 020140 011 061 065 .ASCII / 15 HALT ON ERROR/<12><15>
      020143 011 011 011
      020146 110 101 114
      020151 124 040 117
      020154 116 040 105
      020157 122 122 117
      020162 122 012 015
4117 020165 011 061 064 .ASCII / 14 LOOP ON PRESENT TEST/<12><15>
      020170 011 011 011
      020173 114 117 117
      020176 120 040 117
      020201 116 040 120
      020204 122 105 123

```

.SBTTL GLOBAL ERROR MESSAGES

SWTSEL:

.ASCII <15><12>/KDJ11 DA CPU DIAGNOSTIC - COKDDA0/<15><12><12>

.ASCII /SWITCH REGISTER SELECTION:/<12><12><15>

.ASCII /BIT NUMBER USE/<12><15>

.ASCII / ----- /<12><15>

.ASCII / 15 HALT ON ERROR/<12><15>

.ASCII / 14 LOOP ON PRESENT TEST/<12><15>

GLOBAL ERROR MESSAGES

	020207	105	116	124		
	020212	040	124	105		
	020215	123	124	012		
	020220	015				
4118	020221	011	061	063	.ASCII /	13
	020224	011	011	011		
	020227	111	116	110		
	020232	111	102	111		
	020235	124	040	105		
	020240	122	122	117		
	020243	122	040	124		
	020246	131	120	105		
	020251	117	125	124		
	020254	123	012	015		
4119	020257	011	061	062	.ASCII /	12
	020262	011	011	011		
	020265	105	116	101		
	020270	102	114	105		
	020273	040	124	105		
	020276	123	124	040		
	020301	124	122	101		
	020304	103	111	116		
	020307	107	012	015		
4120	020312	011	061	061	.ASCII /	11
	020315	011	011	011		
	020320	111	116	110		
	020323	111	102	111		
	020326	124	040	111		
	020331	124	105	122		
	020334	101	124	111		
	020337	117	116	123		
	020342	012	015			
4121	020344	011	061	060	.ASCII /	10
	020347	011	011	011		
	020352	102	105	114		
	020355	114	040	117		
	020360	116	040	105		
	020363	122	122	117		
	020366	122	012	015		
4122	020371	011	040	071	.ASCII /	9
	020374	011	011	011		
	020377	114	117	117		
	020402	120	040	117		
	020405	116	040	105		
	020410	122	122	117		
	020413	122	012	015		
4123	020416	011	040	070	.ASCII /	8
	020421	011	011	011		
	020424	114	117	117		
	020427	120	040	117		
	020432	116	040	124		
	020435	105	123	124		
	020440	040	111	116		
	020443	040	123	127		
	020446	122	074	065		
	020451	055	060	076		
	020454	012	015			

GLOBAL ERROR MESSAGES

4124	020456	011	040	067	.ASCII /	7	INHIBIT THE CHECK PARITY TEST/<12><15>
	020461	011	011	011			
	020464	111	116	110			
	020467	111	102	111			
	020472	124	040	124			
	020475	110	105	040			
	020500	103	110	105			
	020503	103	113	040			
	020506	120	101	122			
	020511	111	124	131			
	020514	040	124	105			
	020517	123	124	012			
	020522	015					
4125	020523	011	040	066	.ASCII /	6	Not used/<12><15>
	020526	011	011	011			
	020531	116	157	164			
	020534	040	165	163			
	020537	145	144	012			
	020542	015					
4126	020543	011	065	055	.ASCIZ /	5-0	Subtest number to loop on (BIT 8)/<12><12><1
5>	020546	060	011	011			
	020551	011	123	165			
	020554	142	164	45			
	020557	163	164	040			
	020562	156	165	155			
	020565	142	145	162			
	020570	040	164	157			
	020573	040	154	157			
	020576	157	160	040			
	020601	157	156	040			
	020604	050	102	111			
	020607	124	040	070			
	020612	051	012	012			
	020615	015	000				
4127							
4128	020617	102	101	123	EM1: .ASCIZ /BASIC INSTRUCTION SET ERROR/		
	020622	111	103	040			
	020625	111	116	123			
	020630	124	122	125			
	020633	103	124	111			
	020636	117	116	040			
	020641	123	105	124			
	020644	040	105	122			
	020647	122	117	122			
	020652	000					
4129	020653	115	115	125	EM2: .ASCIZ /MMU ERROR/		
	020656	040	105	122			
	020661	122	117	122			
	020664	000					
4130	020665	106	120	120	EM3: .ASCIZ /FPP ERROR/		
	020670	040	105	122			
	020673	122	117	122			
	020676	000					
4131	020677				EM51:		
4132	020677	103	110	105	EM54: .ASCIZ /CHECKSUM ERROR IN 16 BIT ROM /		
	020702	103	113	123			
	020705	125	115	040			

GLOBAL ERROR MESSAGES

	020710	105	122	122	
	020713	117	122	040	
	020716	111	116	040	
	020721	061	066	055	
	020724	102	111	124	
	020727	040	122	117	
	020732	115	040	000	
4133	020735	124	111	115	EM56: .ASCIZ /TIMEOUT READING LKS/
	020740	105	117	125	
	020743	124	040	122	
	020746	105	101	104	
	020751	111	116	107	
	020754	040	114	113	
	020757	123	000		
4134	020761	114	113	123	EM57: .ASCIZ /LKS<07> DOES NOT BECOME 1/
	020764	074	060	067	
	020767	076	040	104	
	020772	117	105	123	
	020775	040	116	117	
	021000	124	040	102	
	021003	105	103	117	
	021006	115	105	040	
	021011	061	000		
4135	021013	111	114	114	EM61: .ASCIZ /ILLEGAL LKS INTERRUPTS/
	021016	105	107	101	
	021021	114	040	114	
	021024	113	123	040	
	021027	111	116	124	
	021032	105	122	122	
	021035	125	120	124	
	021040	123	000		
4136	021042	114	113	123	EM63: .ASCIZ /LKS READY DOESN'T GO LOW/
	021045	040	122	105	
	021050	101	104	131	
	021053	040	104	117	
	021056	105	123	116	
	021061	047	124	040	
	021064	107	117	040	
	021067	114	117	127	
	021072	000			
4137	021073	127	122	117	EM64: .ASCIZ /WRONG NUMBER OF LKS INTERRUPTS/
	021076	116	107	040	
	021101	116	125	115	
	021104	102	105	122	
	021107	040	117	106	
	021112	040	114	113	
	021115	123	040	111	
	021120	116	124	105	
	021123	122	122	125	
	021126	120	124	123	
	021131	000			
4138	021132	114	113	123	EM65: .ASCIZ /LKS INTERRUPTS HAPPEN AT WRONG PRIORITY/
	021135	040	111	116	
	021140	124	105	122	
	021143	122	125	120	
	021146	124	123	040	
	021151	110	101	120	

B.

GLOBAL ERROR MESSAGES

	021154	120	105	116		
	021157	040	101	124		
	021162	040	127	122		
	021165	117	116	107		
	021170	040	120	122		
	021173	111	117	122		
	021176	111	124	131		
	021201	000				
4139	021202	122	105	123	EM71:	.ASCIZ /RESET DOESN T CLEAR LKS<06>/
	021205	105	124	040		
	021210	104	117	105		
	021213	123	116	047		
	021216	124	040	103		
	021221	114	105	101		
	021224	122	040	114		
	021227	113	123	074		
	021232	060	066	076		
	021235	000				
4140	021236	124	111	115	EM72:	.ASCIZ /TIMEOUT READING SLU REGISTERS/
	021241	105	117	125		
	021244	124	040	122		
	021247	105	101	104		
	021252	111	116	107		
	021255	040	123	114		
	021260	125	040	122		
	021263	105	107	111		
	021266	123	124	105		
	021271	122	123	000		
4141	021274	105	122	122	EM73:	.ASCIZ /ERROR IN XMIT READY/
	021277	117	122	040		
	021302	111	116	040		
	021305	130	115	111		
	021310	124	040	122		
	021313	105	101	104		
	021316	131	000			
4142	021320	122	103	123	EM74:	.ASCIZ /RCSR<7> DOESN'T BECOME 1/
	021323	122	074	067		
	021326	076	040	104		
	021331	117	105	123		
	021334	116	047	124		
	021337	040	102	105		
	021342	103	117	115		
	021345	105	040	061		
	021350	000				
4143	021351	127	122	117	EM75:	.ASCIZ /WRONG CHARACTER RECEIVED/
	021354	116	107	040		
	021357	103	110	101		
	021362	122	101	103		
	021365	124	105	122		
	021370	040	122	105		
	021373	103	105	111		
	021376	126	105	104		
	021401	000				
4144	021402	122	103	123	EM76:	.ASCIZ /RCSR<07> NOT CLEARED AFTER READING RBUF/
	021405	122	074	060		
	021410	067	076	040		
	021413	116	117	124		

GLOBAL ERROR MESSAGES

	021416	040	103	114	
	021421	105	101	122	
	021424	105	104	040	
	021427	101	106	124	
	021432	105	122	040	
	021435	122	105	101	
	021440	104	111	116	
	021443	107	040	122	
	021446	102	125	106	
	021451	000			
4145	021452	130	103	123	EM77: .ASCIZ /XCSR<07> NOT SET ON RESET/
	021455	122	074	060	
	021460	067	076	040	
	021463	116	117	124	
	021466	040	123	105	
	021471	124	040	117	
	021474	116	040	122	
	021477	105	123	105	
	021502	124	000		
4146	021504	122	103	1.	EM100: .ASCIZ /RCSR<07> NOT CLEARED ON RESET/
	021507	122	074	060	
	021512	067	076	040	
	021515	116	117	124	
	021520	040	103	114	
	021523	105	101	122	
	021526	105	104	040	
	021531	117	116	040	
	021534	122	105	123	
	021537	105	124	000	
4147	021542	123	114	125	EM101: .ASCIZ /SLU INTERRUPTS HAPPEN AT 4/
	021545	040	111	116	
	021550	124	105	122	
	021553	122	125	120	
	021556	124	123	040	
	021561	110	101	120	
	021564	120	105	116	
	021567	040	101	124	
	021572	040	064	000	
4148	021575	122	105	123	EM102: .ASCIZ /RESET DOES NOT CLEAR PROPER BITS IN SLU REGISTERS/
	021600	105	124	040	
	021603	104	117	105	
	021606	123	040	116	
	021611	117	124	040	
	021614	103	114	105	
	021617	101	122	040	
	021622	120	122	117	
	021625	120	105	122	
	021630	040	102	111	
	021633	124	123	040	
	021636	111	116	040	
	021641	123	114	125	
	021644	040	122	105	
	021647	107	111	123	
	021652	124	105	122	
	021655	123	000		
4149	021657	124	122	101	EM103: .ASCIZ /TRANSMIT INTERRUPT DOES NOT CLEAR XCSR<07>/
	021662	116	123	115	

D9

GLOBAL ERROR MESSAGES

	021665	111	124	040	
	021670	111	116	124	
	021673	105	122	122	
	021676	125	120	124	
	021701	040	104	117	
	021704	105	123	040	
	021707	116	117	124	
	021712	040	103	114	
	021715	105	101	122	
	021720	040	130	103	
	021723	123	122	074	
	021726	060	067	076	
	021731	000			
4150	021732	122	105	103	EM104: .ASCIZ /RECEIVE INTERRUPTS DO NOT CLEAR RC P/ >/
	021735	105	111	126	
	021740	105	040	111	
	021743	116	124	105	
	021746	122	122	125	
	021751	120	124	123	
	021754	040	104	117	
	021757	116	047	124	
	021762	040	103	114	
	021765	105	101	122	
	021770	040	122	103	
	021773	123	122	074	
	021776	060	067	076	
	022001	000			
4151	022002	102	122	105	EM105: .ASCIZ /BREAK CONDITION DOES NOT SET RBUF PROPERLY/
	022005	101	113	040	
	022010	103	117	116	
	022013	104	111	124	
	022016	111	117	116	
	022021	040	104	117	
	022024	105	123	070	
	022027	116	117	124	
	022032	040	123	105	
	022035	124	040	122	
	022040	102	125	106	
	022043	040	120	122	
	022046	117	120	105	
	022051	122	114	131	
	022054	000			
4152	022055	122	102	125	EM106: .ASCIZ /RBUF <15 11> WASN'T CLEARED ON NEXT CHARACTER/
	022060	106	040	074	
	022063	061	065	055	
	022066	061	061	076	
	022071	040	127	101	
	022074	123	116	047	
	022077	124	040	103	
	022102	114	105	101	
	022105	122	105	104	
	022110	040	117	116	
	022113	040	116	105	
	022116	130	124	040	
	022121	103	110	101	
	022124	122	101	103	
	022127	124	105	122	

E9

GLOBAL ERROR MESSAGES

4153	022132	000				
	022133	123	114	125	EM107:	.ASCIZ /SLU INTERRUPTS DON'T HAPPEN/
	022136	040	111	116		
	022141	124	105	122		
	022144	122	125	120		
	022147	124	123	040		
	022152	104	117	116		
	022155	047	124	040		
	022160	110	101	120		
	022163	120	105	116		
	022166	000				
4154	022167	105	122	122	EM110:	.ASCIZ /ERROR IN WRITING TO SLU REGISTERS/
	022172	117	122	040		
	022175	111	116	040		
	022200	127	122	111		
	022203	124	111	116		
	022206	107	040	124		
	022211	117	040	123		
	022214	114	125	040		
	022217	122	105	107		
	022222	111	123	124		
	022225	105	122	123		
	022230	000				
4155	022231	106	111	122	EM111:	.ASCIZ /FIRST CHARACTER WAS NOT OVERRUN BY THE SECOND/
	022234	123	124	040		
	022237	103	110	101		
	022242	122	101	103		
	022245	124	105	122		
	022250	040	127	101		
	022253	123	040	116		
	022256	117	124	040		
	022261	117	126	105		
	022264	122	122	125		
	022267	116	040	102		
	022272	131	040	124		
	022275	110	105	040		
	022300	123	105	103		
	022303	117	116	104		
	022306	000				
4156	022307	117	126	105	EM112:	.ASCIZ /OVERRUN CONDITION DOES NOT SET PROPER BITS IN RBUF/
	022312	122	122	125		
	022315	116	040	103		
	022320	117	116	104		
	022323	111	124	111		
	022326	117	116	040		
	022331	104	117	105		
	022334	123	040	116		
	022337	117	124	040		
	022342	123	105	124		
	022345	040	120	122		
	022350	117	120	105		
	022353	122	040	102		
	022356	111	124	123		
	022361	040	111	116		
	022364	040	122	102		
	022367	125	106	000		
4157	022372	117	126	105	EM113:	.ASCIZ /OVERRUN BITS WERE NOT CLEARED ON THE NEXT CHARACTER/

FG

GLOBAL ERROR MESSAGES

	022375	122	122	125	
	022400	116	040	102	
	022403	111	124	123	
	022406	040	127	105	
	022411	122	105	040	
	022414	116	117	124	
	022417	040	103	114	
	022422	105	101	122	
	022425	105	104	040	
	022430	117	116	040	
	022433	124	110	105	
	022436	040	116	105	
	022441	130	124	040	
	022444	103	110	101	
	022447	122	101	103	
	022452	124	105	122	
4158	022455	000			
	022456	105	122	122	EM114: .ASCIZ \ERROR ON XCSR<2>\
	022461	117	122	040	
	022464	117	116	040	
	022467	130	103	123	
	022472	122	074	062	
	022475	076	000		
4159	022477	105	122	122	EM123: .ASCIZ /ERROR IN Q22BE DMA CYCLES/
	022502	117	122	040	
	022505	111	116	040	
	022510	121	062	062	
	022513	102	105	040	
	022516	104	115	101	
	022521	040	103	131	
	022524	103	114	105	
	022527	123	000		
4160	022531	120	111	122	EM124: .ASCIZ /PIRQ INTERRUPTS DON'T TAKE PRIORITY OVER Q BUS INTERRUPTS/
	022534	121	040	111	
	022537	116	124	105	
	022542	122	122	125	
	022545	120	124	123	
	022550	040	104	117	
	022553	116	047	124	
	022556	040	124	101	
	022561	113	105	040	
	022564	120	122	111	
	022567	117	122	111	
	022572	124	131	040	
	022575	117	126	105	
	022600	122	040	121	
	022603	040	102	125	
	022606	123	040	111	
	022611	116	124	105	
	022614	122	122	125	
	022617	120	124	123	
	022622	000			
4161	022623	116	117	040	EM125: .ASCIZ /NO POWER DOWN TRAP TO 24 OCCUR/
	022626	120	117	127	
	022631	105	122	040	
	022634	104	117	127	
	022637	116	040	124	

G9

GLOBAL ERROR MESSAGES

	022642	122	101	120	
	022645	040	124	117	
	022650	040	062	064	
	022653	040	117	103	
	022656	103	125	122	
	022661	000			
4162	022662	105	122	122	EM126: .ASCIZ /ERROR DOING Q22BE INTERRUPTS/
	022665	117	122	040	
	022670	104	117	111	
	022673	116	107	040	
	022676	121	062	062	
	022701	102	105	040	
	022704	111	116	124	
	022707	105	122	122	
	022712	125	120	124	
	022715	123	000		
4163	022717	105	122	122	EM127: .ASCIZ /ERROR IN OPERATION OF PMG COUNTER/
	022722	117	122	040	
	022725	111	116	040	
	022730	117	120	105	
	022733	122	101	124	
	022736	111	117	116	
	022741	040	117	106	
	022744	040	120	115	
	022747	107	040	103	
	022752	117	125	116	
	022755	124	105	122	
	022760	000			
4164	022761	125	116	105	EM130: .ASCIZ /UNEXPECTED TRAP TO 4/
	022764	130	120	105	
	022767	103	124	105	
	022772	104	040	124	
	022775	122	101	120	
	023000	040	124	117	
	023003	040	064	000	
4165	023006	105	122	122	EM131: .ASCIZ /ERROR WRITING TO LKS<6>/
	023011	117	122	040	
	023014	127	122	111	
	023017	124	111	116	
	023022	107	040	124	
	023025	117	040	114	
	023030	113	123	074	
	023033	066	076	000	
4166	023036	105	122	122	EM132: .ASCIZ /ERROR IN MAINTENANCE REGISTER/
	023041	117	122	040	
	023044	111	116	040	
	023047	115	101	111	
	023052	116	124	105	
	023055	116	101	116	
	023060	103	105	040	
	023063	122	105	107	
	023066	111	123	124	
	023071	105	122	000	
4167	023074	105	122	122	EM135: .ASCIZ /ERROP IN THE MEMORY DATA PATH/
	023077	117	122	040	
	023102	111	116	040	
	023105	124	110	105	

GLOBAL ERROR MESSAGES

	023110	040	115	105	
	023113	115	117	122	
	023116	131	040	104	
	023121	101	124	101	
	023124	040	120	101	
	023127	124	110	000	
4168	023132	124	111	115	EM136: .ASCIZ /TIMED OUT IN ACCESSING LOCATION 0/
	023135	105	104	040	
	023140	117	125	124	
	023143	040	111	116	
	023146	040	101	103	
	023151	103	105	123	
	023154	123	111	116	
	023157	107	040	114	
	023162	117	103	101	
	023165	124	111	117	
	023170	116	040	060	
	023173	000			
4169	023174	124	111	115	EM137: .ASCIZ /TIMED OUT IN TRYING TO ACCESS MEMORY/
	023177	105	104	040	
	023202	117	125	124	
	023205	040	111	116	
	023210	040	124	122	
	023213	131	111	116	
	023216	107	040	124	
	023221	117	040	101	
	023224	103	103	105	
	023227	123	123	040	
	023232	115	105	115	
	023235	117	122	131	
	023240	000			
4170	023241	105	122	122	EM140: .ASCIZ /ERROR IN FUNCTIONAL REVISION BITS ON THE NATIVE REGISTER/
	023244	117	122	040	
	023247	111	116	040	
	023252	106	125	116	
	023255	103	124	111	
	023260	117	116	101	
	023263	114	040	122	
	023266	105	126	111	
	023271	123	111	117	
	023274	116	040	102	
	023277	111	124	123	
	023302	040	117	116	
	023305	040	124	110	
	023310	105	040	116	
	023313	101	124	111	
	023316	126	105	040	
	023321	122	105	107	
	023324	111	123	124	
	023327	105	122	000	
4171	023332	105	122	122	EM141: .ASCIZ /ERROR IN THE INDICATOR BITS ON THE NATIVE REGISTER/
	023335	117	122	040	
	023340	111	116	040	
	023343	124	110	105	
	023346	040	111	116	
	023351	104	111	103	
	023354	101	124	117	

GLOBAL ERROR MESSAGES

	023357	122	040	102	
	023362	111	124	123	
	023365	040	117	116	
	023370	040	124	110	
	023373	105	040	116	
	023376	101	124	111	
	023401	126	105	040	
	023404	122	105	107	
	023407	111	123	124	
	023412	105	122	000	
4172	023415	105	122	122	EM142: .ASCIZ /ERROR IN THE BOOT SELECT SWITCHES ON THE NATIVE REGISTER/
	023420	117	122	040	
	023423	111	116	040	
	023426	124	110	105	
	023431	040	102	117	
	023434	117	124	040	
	023437	123	105	114	
	023442	105	103	124	
	023445	040	123	127	
	023450	111	124	103	
	023453	110	105	123	
	023456	040	117	116	
	023461	040	124	110	
	023464	105	040	116	
	023467	101	124	111	
	023472	126	105	040	
	023475	122	105	107	
	023500	111	123	124	
	023503	105	122	000	
4173	023506	124	111	115	EM143: .ASCIZ /TIMED OUT IN ACCESS TO THE NATIVE REGISTER/
	023511	105	104	040	
	023514	117	125	124	
	023517	040	111	116	
	023522	040	101	103	
	023525	103	105	123	
	023530	123	040	124	
	023533	117	040	124	
	023536	110	105	040	
	023541	116	101	124	
	023544	111	126	105	
	023547	040	122	105	
	023552	107	111	123	
	023555	124	105	122	
	023560	000			
4174	023561	102	111	124	EM144: .ASCIZ /BIT 7 IN LTC IS NOT TOGGLING BETWEEN 0 AND 1/
	023564	040	067	040	
	023567	111	116	040	
	023572	114	124	103	
	023575	040	111	123	
	023600	040	116	117	
	023603	124	040	124	
	023606	117	107	107	
	023611	114	111	116	
	023614	107	040	102	
	023617	105	124	127	
	023622	105	105	116	
	023625	040	060	040	

J9

GLOBAL ERROR MESSAGES

	023630	101	116	104	
	023633	040	061	000	
4175	023636	124	111	115	EM145: .ASCIZ /TIMEOUT IN ACCESSING THE LKS REGISTER/
	023641	105	117	125	
	023644	124	040	111	
	023647	116	040	101	
	023652	103	103	105	
	023655	123	123	111	
	023660	116	107	040	
	023663	124	110	105	
	023666	040	114	113	
	023671	123	040	122	
	023674	105	107	111	
	023677	123	124	105	
	023702	122	000		
4176	023704	105	122	122	EM146: .ASCIZ /ERROR IN STUCK AT ZERO BITS ON MER/
	023707	117	122	040	
	023712	111	116	040	
	023715	123	124	125	
	023720	103	113	040	
	023723	101	124	040	
	023726	132	105	122	
	023731	117	040	102	
	023734	111	124	123	
	023737	040	117	116	
	023742	040	115	105	
	023745	122	000		
4177	023747	103	117	125	EM147: .ASCIZ \COULD NOT SET ONE OF THE R/W BITS ON THE MER\
	023752	114	104	040	
	023755	116	117	124	
	023760	040	123	105	
	023763	124	040	117	
	023766	116	105	040	
	023771	117	106	040	
	023774	124	110	105	
	023777	040	122	057	
	024002	127	040	102	
	024005	111	124	123	
	024010	040	117	116	
	024013	040	124	110	
	024016	105	040	115	
	024021	105	122	000	
4178	024024	102	111	124	EM150: .ASCIZ /BITS 0,2,14,15 ON THE MER DID NOT CLEAR ON RESET/
	024027	123	040	060	
	024032	054	062	054	
	024035	061	064	054	
	024040	061	065	040	
	024043	117	116	040	
	024046	124	110	105	
	024051	040	115	105	
	024054	122	040	104	
	024057	111	104	040	
	024062	116	117	124	
	024065	040	103	114	
	024070	105	101	122	
	024073	040	117	116	
	024076	040	122	105	

GLOBAL ERROR MESSAGES

	024101	123	105	124	
	024104	000			
4179	024105	124	111	115	EM151: .ASCIZ /TIMEOUT IN ACCESSING THE MER REGISTER/
	024110	105	117	125	
	024113	124	040	111	
	024116	116	040	101	
	024121	103	103	105	
	024124	123	123	111	
	024127	116	107	040	
	024132	124	110	105	
	024135	040	115	105	
	024140	122	040	122	
	024143	105	107	111	
	024146	123	124	105	
	024151	122	000		
4180	024153	105	122	122	EM152: .ASCIZ /ERROR IN THE DATA SHORTS AND STUCK AT MEMORY TEST/
	024156	117	122	040	
	024161	111	116	040	
	024164	124	110	105	
	024167	040	104	101	
	024172	124	101	040	
	024175	123	110	117	
	024200	122	124	123	
	024203	040	101	116	
	024206	104	040	123	
	024211	124	125	103	
	024214	113	040	101	
	024217	124	040	115	
	024222	105	115	117	
	024225	122	131	040	
	024230	124	105	123	
	024233	124	000		
4181	024235	120	101	122	EM153: .ASCIZ /PARITY ABORT OCCURED WITH PARITY ERROR DISABLED/
	024240	111	124	131	
	024243	040	101	102	
	024246	117	122	124	
	024251	040	117	103	
	024254	103	125	122	
	024257	105	104	040	
	024262	127	111	124	
	024265	110	040	120	
	024270	101	122	111	
	024273	124	131	040	
	024276	105	122	122	
	024301	117	122	040	
	024304	104	111	123	
	024307	101	102	114	
	024312	105	104	000	
4182	024315	120	101	122	EM154: .ASCIZ /PARITY ABORT DID NOT OCCUR WITH PARITY ERROR ENABLED/
	024320	111	124	131	
	024323	040	101	102	
	024326	117	122	124	
	024331	040	104	111	
	024334	104	040	116	
	024337	117	124	040	
	024342	117	103	103	
	024345	125	122	040	

GLOBAL ERROR MESSAGES

	024350	127	111	124	
	024353	110	040	120	
	024356	101	122	111	
	024361	124	131	040	
	024364	105	122	122	
	024367	117	122	040	
	024372	105	116	101	
	024375	102	114	105	
	024400	104	000		
4183	024402	115	105	122	EM155: .ASCIZ /MER DIDN'T HAVE CORRECT ADDRESS BITS 11 17/
	024405	040	104	111	
	024410	104	116	047	
	024413	124	040	110	
	024416	101	126	105	
	024421	040	103	117	
	024424	122	122	105	
	024427	103	124	040	
	024432	101	104	104	
	024435	122	105	123	
	024440	123	040	102	
	024443	111	124	123	
	024446	040	061	061	
	024451	055	061	067	
	024454	000			
4184	024455	115	105	122	EM156: .ASCIZ /MER READ EXTENDED ADDRESS BITS HAS FAILED/
	024460	040	122	105	
	024463	101	104	040	
	024466	105	130	124	
	024471	105	116	104	
	024474	105	104	040	
	024477	101	104	104	
	024502	122	105	123	
	024505	123	040	102	
	024510	111	124	123	
	024513	040	110	101	
	024516	123	040	106	
	024521	101	111	114	
	024524	105	104	000	
4185	024527	105	122	122	EM157: .ASCIZ /ERROR IN THE QUICK VERIFY MEMORY TEST/
	024532	117	122	040	
	024535	111	116	040	
	024540	124	110	105	
	024543	040	121	125	
	024546	111	103	113	
	024551	040	126	105	
	024554	122	111	106	
	024557	131	040	115	
	024562	105	115	117	
	024565	122	131	040	
	024570	124	105	123	
	024573	124	000		
4186	024575	105	122	122	EM160: .ASCIZ /ERROR IN RCSR <6>/
	024600	117	122	040	
	024603	111	116	040	
	024606	122	103	123	
	024611	122	040	074	
	024614	066	076	000	

GLOBAL ERROR MESSAGES

4187	024617	116	117	040	EM161: .ASCIZ /NO XMIT INTERRUPTS HAVE OCCURED/
	024622	130	115	111	
	024625	124	040	111	
	024630	116	124	105	
	024633	122	122	125	
	024636	120	124	123	
	024641	040	110	101	
	024644	126	105	040	
	024647	117	103	103	
	024652	125	122	105	
	024655	104	000		
4188	024657	116	117	040	EM162: .ASCIZ /NO RECIEVE INTERRUPTS HAVE OCCURED/
	024662	122	105	103	
	024665	111	105	126	
	024670	105	040	111	
	024673	116	124	105	
	024676	122	122	125	
	024701	120	124	123	
	024704	040	110	101	
	024707	126	105	040	
	024712	117	103	103	
	024715	125	122	105	
	024720	104	000		
4189	024722	125	116	105	EM163: .ASCIZ /UNEXPECTED PARITY ABORT HAS OCCURED/
	024725	130	120	105	
	024730	103	124	105	
	024733	104	040	120	
	024736	101	122	111	
	024741	124	131	040	
	024744	101	102	117	
	024747	122	124	040	
	024752	110	101	123	
	024755	040	117	103	
	024760	103	125	122	
	024763	105	104	000	
4190	024766	115	105	122	EM164: .ASCIZ /MER DIDN'T HAVE CORRECT ADDRESS BITS# 0 AND 15/
	024771	040	104	111	
	024774	104	116	047	
	024777	124	040	110	
	025002	101	126	105	
	025005	040	103	117	
	025010	122	122	105	
	025013	103	124	040	
	025016	101	104	104	
	025021	122	105	123	
	025024	123	040	102	
	025027	111	124	123	
	025032	043	040	060	
	025035	040	101	116	
	025040	104	040	061	
	025043	065	000		
4191	025045	124	117	117	EM165: .ASCIZ /TOO MANY TRANSIVER INTERRUPTS HAPPENED/
	025050	040	115	101	
	025053	116	131	040	
	025056	124	122	101	
	025061	116	123	111	
	025064	126	105	122	

GLOBAL ERROR MESSAGES

	025067	040	111	116	
	025072	124	105	122	
	025075	122	125	120	
	025100	124	123	040	
	025103	110	101	120	
	025106	120	105	116	
	025111	105	104	000	
4192	025114	124	117	117	EM166: .ASCIZ /TOO MANY RECEIVFR INTERRUPTS HAPPENED/
	025117	040	115	101	
	025122	116	131	040	
	025125	122	105	103	
	025130	105	111	126	
	025133	105	122	040	
	025136	111	116	124	
	025141	105	122	122	
	025144	125	120	124	
	025147	123	040	110	
	025152	101	120	120	
	025155	105	116	105	
	025160	104	000		
4193	025162	105	122	122	em167: .asciz /ERROR IN READY BIT OF CSR/
	025165	117	122	040	
	025170	111	116	040	
	025173	122	105	101	
	025176	104	131	040	
	025201	102	111	124	
	025204	040	117	106	
	025207	040	103	123	
	025212	122	000		
4194	025214	116	117	040	em170: .asciz /NO CHARACTER RECEIVED/
	025217	103	110	101	
	025222	122	101	103	
	025225	124	105	122	
	025230	040	122	105	
	025233	103	105	111	
	025236	126	105	104	
	025241	000			
4195	025242	127	122	117	em171: .asciz /WRONG CHARACTER RECEIVED/
	025245	116	107	040	
	025250	103	110	101	
	025253	122	101	103	
	025256	124	105	122	
	025261	040	122	105	
	025264	103	105	111	
	025267	126	105	104	
	025272	000			
4196	025273	122	117	115	em172: .asciz /ROM SIZE IS ZERO/
	025276	040	123	111	
	025301	132	105	040	
	025304	111	123	040	
	025307	132	105	122	
	025312	117	000		
4197					
4198	025314	040	124	105	DH1: .ASCII / TEST ERROR/<15><12>
	025317	123	124	011	
	025322	105	122	122	
	025325	117	122	015	

GLOBAL ERROR MESSAGES

4199	025330	012							
	025331	040	040	043		.ASCIZ	/	*	PC/
	025334	011	040	120					
	025337	103	000						
4200	025341	040	124	105	DH4:	.ASCII	/	TEST	ERROR
	025344	123	124	011				EXPTED	RECEIVED/<15><12>
	025347	105	122	122					
	025352	117	122	011					
	025355	105	130	120					
	025360	103	124	105					
	025363	104	011	122					
	025366	105	103	105					
	025371	111	126	105					
	025374	104	015	012					
4201	025377	040	040	043		.ASCIZ	/	*	PC
	025402	011	040	120				DATA	DATA/
	025405	103	011	040					
	025410	104	101	124					
	025413	101	040	040					
	025416	040	040	040					
	025421	104	101	124					
	025424	101	000						
4202	025426	040	124	105	DH5:	.ASCII	/	TEST	ERROR
	025431	123	124	011				HITMIS	DATA IN DATA IN/<15><12>
	025434	105	122	122					
	025437	117	122	011					
	025442	110	111	124					
	025445	115	111	123					
	025450	011	104	101					
	025453	124	101	040					
	025456	111	116	011					
	025461	104	101	124					
	025464	101	040	111					
	025467	116	015	012					
4203	025472	040	040	043		.ASCIZ	/	*	PC
	025475	011	040	120				REG.	CACHE
	025500	103	011	040				MEMORY/	
	025503	122	105	107					
	025506	056	011	103					
	025511	101	103	110					
	025514	105	011	115					
	025517	105	115	117					
	025522	122	131	000					
4204	025525	040	124	105	DH7:	.ASCII	/	TEST	ERROR
	025530	123	124	011				ADDRESS	MSER/<15><12>
	025533	105	122	122					
	025536	117	122	011					
	025541	101	104	104					
	025544	122	105	123					
	025547	123	011	115					
	025552	123	105	122					
	025555	015	012						
4205	025557	040	040	043		.ASCIZ	/	*	PC
	025562	011	040	120				ACCESSED/	
	025565	103	011	101					
	025570	103	103	105					
	025573	123	123	105					

GLOBAL ERROR MESSAGES

4206	025576	104	000						
	025600	040	124	105	DH24:	.ASCII	/	TEST	ERROR
	025603	123	124	011					NUMBER/<15><12>
	025606	105	122	122					
	025611	117	122	011					
	025614	116	125	115					
	025617	102	105	122					
4207	025622	015	012						
	025624	040	040	043		.ASCIZ	/	#	PC
	025627	011	040	120					OF HITS/
	025632	103	011	117					
	025635	106	040	110					
	025640	111	124	123					
	025643	000							
4208	025644	105	122	122	DH27:	.ASCII	\	ERROR	ERROR
	025647	117	122	011					MSER
	025652	105	122	122					HIT/MISS\<15><12>
	025655	117	122	011					
	025660	115	123	105					
	025663	122	011	110					
	025666	111	124	057					
	025671	115	111	123					
	025674	123	015	012					
4209	025677	040	040	043		.ASCIZ	/	#	PC/
	025702	011	040	120					
	025705	103	000						
4210	025707	040	124	105	DH41:	.ASCII	/	TEST	ERROR
	025712	123	124	011					INSTRUCTION/<15><12>
	025715	105	122	122					
	025720	117	122	011					
	025723	111	116	123					
	025726	124	122	125					
	025731	103	124	111					
	025734	117	116	015					
	025737	012							
4211	025740	040	040	043		.ASCIZ	/	#	PC
	025743	011	040	120					OPCODE/
	025746	103	011	040					
	025751	117	120	103					
	025754	117	104	105					
	025757	000							
4212	025760	040	124	105	DH43:	.ASCII	/	TEST	ERROR
	025763	123	124	011					EXPECTD RECEIVD CACHE/<15><12>
	025766	105	122	122					
	025771	117	122	011					
	025774	105	130	120					
	025777	105	103	124					
	026002	104	011	122					
	026005	105	103	105					
	026010	111	126	104					
	026013	011	103	101					
	026016	103	110	105					
	026021	015	012						
4213	026023	040	040	043		.ASCIZ	/	#	PC
	026026	011	040	120					DATA
	026031	103	011	040					DATA
	026034	104	101	124					LOCATION/

D10

GLOBAL ERROR MESSAGES

	026037	101	011	040					
	026042	104	101	124					
	026045	101	011	114					
	026050	117	103	101					
	026053	124	111	117					
	026056	116	000						
4214	026060	040	124	105	DH47:	.ASCII	/ TEST ERROR	ADDRESS	ADDRESS/<15><12>
	026063	123	124	011					
	026066	105	122	122					
	026071	117	122	011					
	026074	101	104	104					
	026077	122	105	123					
	026102	123	011	101					
	026105	104	104	122					
	026110	105	123	123					
	026113	015	012						
4215	026115	040	040	043		.ASCIZ	/ # PC	<21-16>	<15-0>/
	026120	011	040	120					
	026123	103	011	074					
	026126	062	061	055					
	026131	061	066	076					
	026134	011	040	074					
	026137	061	065	055					
	026142	060	076	000					
4216	026145	040	124	105	DH65:	.ASCII	/ TEST ERROR	PRIORITY	/<15><12>
	026150	123	124	011					
	026153	105	122	122					
	026156	117	122	011					
	026161	120	122	111					
	026164	117	122	111					
	026167	124	131	015					
	026172	012							
4217	026173	040	040	043		.ASCIZ	/ # PC	LEVEL	/
	026176	011	040	120					
	026201	103	011	114					
	026204	105	126	105					
	026207	114	000						
4218	026211	040	124	105	DH72:	.ASCII	/ TEST ERROR	ADDRESS	/<15><12>
	026214	123	124	011					
	026217	105	122	122					
	026222	117	122	011					
	026225	101	104	104					
	026230	122	105	123					
	026233	123	015	012					
4219	026236	040	040	043		.ASCIZ	/ # PC	FAILED	/
	026241	011	040	120					
	026244	103	011	106					
	026247	101	111	114					
	026252	105	104	000					
4220	026255	040	124	105	DH105:	.ASCII	/ TEST ERROR	RBUF	/<15><12>
	026260	123	124	011					
	026263	105	122	122					
	026266	117	122	011					
	026271	122	102	125					
	026274	106	015	012					
4221	026277	040	040	043		.ASCIZ	/ # PC		/
	026302	011	040	120					

GLOBAL ERROR MESSAGES

4222	026305	103	000								
	026307	040	124	105	DH115:	.ASCII	/	TEST	ERROR	MSER	ADDRESS/ <15> <12>
	026312	123	124	011							
	026315	105	122	122							
	026320	117	122	011							
	026323	115	123	105							
	026326	122	011	101							
	026331	104	104	122							
	026334	105	123	123							
	026337	015	012								
4223	026341	040	040	043		.ASCIZ	/	#	PC		ACCESSED/
	026344	011	040	120							
	026347	103	040	011							
	026352	011	101	103							
	026355	103	105	123							
	026360	123	105	104							
	026363	000									
4224	026364	040	124	105	DH134:	.ASCII	/	TEST	ERROR		DATA PAR VIRTUAL/ <15> <12>
	026367	123	124	011							
	026372	105	122	122							
	026375	117	122	011							
	026400	011	040	040							
	026403	040	104	101							
	026406	124	101	040							
	026411	040	120	101							
	026414	122	040	040							
	026417	040	126	111							
	026422	122	124	125							
	026425	101	114	015							
	026430	012									
4225	026431	040	040	043		.ASCIZ	/	#	PC	PATTERN	READ ADDRESS/
	026434	011	040	120							
	026437	103	040	011							
	026442	120	101	124							
	026445	124	105	122							
	026450	116	040	040							
	026453	040	040	122							
	026456	105	101	104							
	026461	040	040	040							
	026464	040	040	040							
	026467	040	040	101							
	026472	104	104	122							
	026475	105	123	123							
	026500	000									
4226						.EVEN					
4227						.WORD					
4228	026502	001162	001116	000000	DT1:	\$TMP1,	\$ERRPC,	0			
4229	026510	001162	001116	000001	DT4:	\$TMP1,	\$ERRPC,	R1,	CCR,	0	
	026516	177746	000000								
4230	026522	001162	001116	000002	DT5:	\$TMP1,	\$ERRPC,	R2,	R1,	\$GDDAT,	0
	026530	000001	001124	000000							
4231	026536	001162	001116	001122	DT7:	\$TMP1,	\$ERRPC,	\$BDADR,	MSER,	0	
	026544	177744	000000								
4232	026550	001162	001116	001124	DT14:	\$TMP1,	\$ERRPC,	\$GDDAT,	TSTLOC,	0	
	026556	003166	000000								
4233	026562	001162	001116	001124	DT17:	\$TMP1,	\$ERRPC,	\$GDDAT,	RECDAT,	0	
	026570	004034	000000								

GLOBAL ERROR MESSAGES

4234	026574	001162	001116	000003	DT24:	.WORD	\$TMP1,\$ERRPC,R3,0
	026602	000000					
4235	026604	001162	001116	177744	DT27:	.WORD	\$TMP1,\$ERRPC,MSER,R3,0
	026612	000003	000000				
4236	026616	001162	001116	001124	DT35:	.WORD	\$TMP1,\$ERRPC,\$GDDAT,MSER,0
	026624	177744	000000				
4237	026630	001162	001116	001126	DT41:	.WORD	\$TMP1,\$ERRPC,\$BDDAT,0
	026636	000000					
4238	026640	001162	001116	000001	DT43:	.WORD	\$TMP1,\$ERRPC,R1,RECDAT,\$BDADR,0
	026646	004034	001122	000000			
4239	026654	001162	001116	172354	DT47:	.WORD	\$TMP1,\$ERRPC,KIPAR6,\$BDADR,0
	026662	001122	000000				
4240	026666	001162	001116	000001	DT50:	.WORD	\$TMP1,\$ERRPC,R1,\$BDADR,0
	026674	001122	000000				
4241	026700	001162	001116	001124	DT51:	.WORD	\$TMP1,\$ERRPC,\$GDDAT,PCR,0
	026706	177522	000000				
4242	026712	001162	001116	001124	DT52:	.WORD	\$TMP1,\$ERRPC,\$GDDAT,BCSR,0
	026720	177520	000000				
4243	026724	001162	001116	001124	DT64:	.WORD	\$TMP1,\$ERRPC,\$GDDAT,LKSFL,0
	026732	002402	000000				
4244	026736	001162	001116	001124	DT65:	.WORD	\$TMP1,\$ERRPC,\$GDDAT,0
	026744	000000					
4245	026746	001162	001116	001124	DT75:	.WORD	\$TMP1,\$ERRPC,\$GDDAT,\$BDDAT,0
	026754	001126	000000				
4246	026760	001162	001116	177562	DT105:	.WORD	\$TMP1,\$ERRPC,RBUF,0
	026766	000000					
4247	026770	001162	001116	001126	DT115:	.WORD	\$TMP1,\$ERRPC,\$BDDAT,KIPAR6,\$BDADR,0
	026776	172354	001122	000000			
4248	027004	001162	001122	000000	DT130:	.WORD	\$TMP1,\$BDADR,0
4249	027012	001162	001116	001124	DT134:	.WORD	\$TMP1,\$ERRPC,\$GDDAT,\$BDDAT,KIPAR2,\$BDADR,0
	027020	001126	172344	001122			
	027026	000000					

MODIFIED ERROR MESSAGE TYPEOUT ROUTINE

```

4251 .SBTTL MODIFIED ERROR MESSAGE TYPEOUT ROUTINE
4252 ;*****
4253 ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
4254 ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
4255 ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
4256 ;*
4257 ;*THE ONLY DIFFERENCE BETWEEN THIS ROUTINE AND THE ORIGINAL "$ERRTYP" FROM
4258 ;*SYSMAC IS THAT YOU CAN PASS INFORMATION IN GENERAL PURPOSE REGISTERS TO THIS
4259 ;*ROUTINE. THE GENERAL PURPOSE REGISTERS USED ARE TO BE SPECIFIED IN DT*
4260 ;*FORMAT. RO SHOULD NOT BE USED.
4261
4262 027030          .ERTYPE:
4263 027030 005037 001162          CLR    $TMP1          ;;JUST CLEAR IT
4264 027034 113737 001102 001162  MOVB   $TSTNM,$TMP1  ;;STORE TEST NUMBER
4265 027042 104401 001175          TYPE   , $CRLF        ;; 'CARRIAGE RETURN' & "LINE FEED"
4266 027046 010046          MOV    RO, (SP)          ;;SAVE RO
4267 027050 005000          CLR    RO              ;;PICKUP THE ITEM INDEX
4268 027052 153700 001114          BISB  @#$ITEMB,RO
4269 027056 001004          BNE    1$              ;;IF ITEM NUMBER IS ZERO, JUST
4270                                ;;TYPE THE PC OF THE ERROR
4271 027060 013746 001116          MOV    $ERRPC, (SP)      ;;SAVE $ERRPC FOR TYPEOUT
4272                                ;;ERROR ADDRESS
4273 027064 104402          TYPCC          ;;GO TYPE- OCTAL ASCII(ALL DIGITS)
4274 027066 000426          BR     6$              ;;GET OUT
4275 027070 005300          1$: DEC    RO              ;;ADJUST THE INDEX SO THAT IT WILL
4276 027072 006300          ASL    RO              ;; WORK FOR THE ERROR TABLE
4277 027074 006300          ASL    RO
4278 027076 006300          ASL    RO
4279 027100 062700 001324          ADD    #$ERRTB,RO    ;;FORM TABLE POINTER
4280 027104 012037 027114          MOV    (RO)+,2$      ;;PICKUP "ERROR MESSAGE" POINTER
4281 027110 001404          BEQ    3$              ;;SKIP TYPEOUT IF NO POINTER
4282 027112 104401          TYPE          ;;
4283 027114 000000          2$: .WORD 0          ;;ERROR MESSAGE POINTER GOES HERE
4284 027116 104401 001175          TYPE   , $CRLF        ;; "CARRIAGE RETURN" & "LINE FEED"
4285 027122 012037 027132          3$: MOV    (RO)+,4$      ;;PICKUP "DATA HEADER" POINTER
4286 027126 001404          BEQ    5$              ;;SKIP TYPEOUT IF 0
4287 027130 104401          TYPE          ;;TYPE THE "DATA HEADER"
4288 027132 000000          4$: .WORD 0          ;; "DATA HEADER" POINTER GOES HERE
4289 027134 104401 001175          TYPE   , $CRLF        ;; "CARRIAGE RETURN" & "LINE FEED"
4290 027140 011000          5$: MOV    (RO),RO          ;;PICKUP "DATA TABLE" POINTER
4291 027142 001004          BNE    7$              ;;GO TYPE THE DATA
4292 027144 012600          6$: MOV    (SP)+,RO        ;;RESTORE RO
4293 027146 104401 001175          TYPE   , $CRLF        ;; "CARRIAGE RETURN" & "LINE FEED"
4294 027152 000207          RTS    PC              ;;RETURN
4295 027154          7$:
4296 027154 021027 000005          CMP    (RO),#5        ;;GENERAL PURPOSE REGISTER?
4297 027160 101021          BHI    9$              ;;IF NOT, GO TYPE DATA
4298 027162 042737 000700 027216  BIC    #BIT8!BIT7!BIT6,8$ ;;CLEAR BITS FOR SOURCE REGISTER
4299 027170 011037 001160          MOV    (RO),$TMP0    ;;SAVE (RO)
4300 027174 000337 001160          SWAB  $TMP0          ;;GET REGISTER NUMBER TO HIGH BYTE
4301 027200 006237 001160          ASR   $TMP0          ;;GET REGISTER NUMBER TO BITS 8 6
4302 027204 006237 001160          ASR   $TMP0
4303 027210 053737 001160 027216  BIS    $TMP0,8$      ;;SET BITS IN MOV INSTRUCTION
4304                                ;;ACCORDING TO REGISTER NUMBER
4305 027216 010046          8$: MOV    RO, (SP)          ;;MOVE CONTEXT OF REGISTER TO STACK
4306 027220 005720          TST   (RO)+          ;;ADVANCE POINTER
4307 027222 000401          BR     10$         ;;GO TYPE

```

MODIFIED ERROR MESSAGE TYPEOUT ROUTINE

```

4308 027224 013046          9$:   MOV    @ (R0) , (SP)    ;; IF NOT GPR, SAVE @ (R0) FOR TYPEOUT
4309 027226 104402          10$:  TYPUC                ;; GO TYPE- OCTAL ASCII (ALL DIGITS)
4310 027230 005710          TST    (R0)                ;; IS THERE ANOTHER NUMBER?
4311 027232 001744          BEQ    6$                    ;; BR IF NO
4312 027234 104401 027242   TYPE   ,11$                 ;; TYPE TWO(2) SPACES
4313 027240 000745          BR     7$                    ;;
4314 027242   040   040   000 11$:  .ASCIZ  / /                    ;; TWO(2) SPACES
4315                                     .EVEN
4316                                     .SBTTL  GLOBAL SUBROUTINES SECTION
4317
4318                                     ;**
4319                                     ; THE GLOBAL SUBROUTINES SECTION CONTAINS THE SUBROUTINES
4320                                     ; THAT ARE USED IN MORE THAN ONE TEST.
4321                                     ; --
4322
4323                                     ;**
4324                                     ; FUNCTIONAL DESCRIPTION:
4325                                     ; SUBROUTINE TO INITIALIZE ALL THE MMU REGISTERS
4326
4327
4328                                     ; INPUTS: NONE
4329
4330                                     ; OUTPUTS: NONE
4331
4332                                     ; SUBORDINATE ROUTINES USED: LOAD PARS
4333                                     ; LOAD PDRS
4334
4335                                     ; FUNCTIONAL SIDE EFFECTS: NONE
4336
4337                                     ; CALLING SEQUENCE:      JSR    PC,INITMM
4338
4339
4340 027246 012701 172240   INITMM: MOV    #172240,R1                ;BASE ADDRESS OF SIPARS
4341 027252 004737 027410   JSR    PC, LDPARS
4342 027256 012701 172260   MOV    #172260,R1                ;BASE ADDRESS OF SPPARS
4343 027262 004737 027410   JSR    PC, LDPARS
4344 027266 012701 172340   MOV    #172340,R1                ;BASE ADDRESS OF KIPARS
4345 027272 004737 027410   JSR    PC, LDPARS
4346 027276 012701 172360   MOV    #172360,R1                ;BASE ADDRESS OF KDPARS
4347 027302 004737 027410   JSR    PC, LDPARS
4348 027306 012701 177640   MOV    #177640,R1                ;BASE ADDRESS OF UIPARS
4349 027312 004737 027410   JSR    PC, LDPARS
4350 027316 012701 177660   MOV    #177660,R1                ;BASE ADDRESS OF UPPARS
4351 027322 004737 027410   JSR    PC, LDPARS
4352 027326 012701 177600   MOV    #177600,R1                ;BASE ADDRESS OF UIPDRS
4353 027332 004737 027440   JSR    PC, LDPDRS
4354 027336 012701 177620   MOV    #177620,R1                ;BASE ADDRESS OF UDPDRS
4355 027342 004737 027440   JSR    PC, LDPDRS
4356 027346 012701 172300   MOV    #172300,R1                ;BASE ADDRESS OF KIPDRS
4357 027352 004737 027440   JSR    PC, LDPDRS
4358 027356 012701 172320   MOV    #172320,R1                ;BASE ADDRESS OF KDPDRS
4359 027362 004737 027440   JSR    PC, LDPDRS
4360 027366 012701 172200   MOV    #172200,R1                ;BASE ADDRESS OF SIPDRS
4361 027372 004737 027440   JSR    PC, LDPDRS
4362 027376 012701 172220   MOV    #172220,R1                ;BASE ADDRESS OF SPPDRS
4363 027402 004737 027440   JSR    PC, LDPDRS
4364 027406 000207          RTS    PC                        ;RETURN

```

GLOBAL SUBROUTINES SECTION

4366
4367
4368
4369
4370
4371
4372
4373
4374
4375
4376
4377
4378
4379
4380
4381
4382
4383
4384
4385
4386 027410 012702 000006
4387 027414 005003
4388 027416 010321
4389 027420 062703 000200
4390 027424 077204
4391 027426 012721 002000
4392 027432 012711 177600
4393 027436 000207

```

; **
; FUNCTIONAL DESCRIPTION:
;   SUBROUTINE TO INITIALIZE ALL THE MMU PAGE ADDRESS REGISTERS (PARS)
;   THIS ROUTINE WILL INITIALIZE 8 PARS STARTING AT A BASE ADDRESS
;   SUPPLIED BY THE CALLING ROUTINE. PARS 0 5 WILL BE MAPPED FROM
;   ADDRESS 0 TO ADDRESS 137777 (0-24K). PAR 6 WILL BE MAPPED FROM
;   ADDRESS 200000 TO 217777 AND PAR 7 WILL BE MAPPED TO THE I/O
;   PAGE.
;
; INPUTS:
;   R1 CONTAINS THE BASE ADDRESS OF THE NEXT 8 PARS TO BE INITIALIZED
;
; OUTPUTS: NONE
;
; SUBORDINATE ROUTINES USED: NONE
;
; FUNCTIONAL SIDE EFFECTS: NONE
;
; CALLING SEQUENCE:      JSR      PC,LDPARS
LDPARS: MOV      #6,      R2          ;LET LOOP COUNTER COUNT FIRST 6 PARS
          CLR      R3          ;INITIALIZE INDEX VALUE
1$:      MOV      R3,      (R1)+    ;LOAD PARS
          ADD      #200,    R3      ;INDEX IN 4K INCREMENTS
          SOB     R2,      1$      ;LOAD FIRST SIX PARS
          MOV      #2000,   (R1)+   ;LET PAR6 MAP TO 200000
          MOV      #177600,(R1)+   ;LET PAR7 MAP TO I/O PAGE
          RTS     PC          ;RETURN

```

GLOBAL SUBROUTINES SECTION

4395
 4396
 4397
 4398
 4399
 4400
 4401
 4402
 4403
 4404
 4405
 4406
 4407
 4408
 4409
 4410
 4411
 4412
 4413
 4414
 4415
 4416 027440 012702 000006
 4417 027444 012721 177406
 4418 027450 077203
 4419 027452 012721 077406
 4420 027456 012711 077406
 4421 027462 000207

```

; **
; FUNCTIONAL DESCRIPTION:
; SUBROUTINE TO INITIALIZE ALL THE MMU PAGE DESCRIPTOR REGISTERS (PDRS).
; THIS ROUTINE WILL INITIALIZE 8 PDRS STARTING AT A BASE ADDRESS
; SUPPLIED BY THE CALLING ROUTINE. PDRS 0 5 WILL BE INITIALIZED TO
; 4K READ/WRITE BYPASS AND PDRS 6 AND 7 WILL BE INITIALIZED TO
; 4K READ/WRITE NO BYPASS.
; NOTE: THERE IS NO NEED TO BYPASS ON I/O PAGE REFERENCES BECAUSE
; THE CACHE DOES NOT ALLOCATE ANY OF THESE REFERENCES.
;
; INPUTS:
; R1 CONTAINS THE BASE ADDRESS OF THE NEXT 8 PDRS TO BE INITIALIZED
;
; OUTPUTS: NONE
;
; SUBORDINATE ROUTINES USED: NONE
;
; FUNCTIONAL SIDE EFFECTS: NONE
;
; CALLING SEQUENCE: JSR PC,LDPARS
;
LDPDRS: MOV #6, R2 ;LET LOOP COUNTER COUNT FIRST 6 PARS
1$: MOV #177406,(R1)+ ;LOAD PDRS WITH 4K READ/WRITE BYPASS
SOB R2, 1$ ;LOAD FIRST SIX PDRS
MOV #77406,(R1)+ ;LET PAR6 BE 4K READ/WRITE NO BYPASS
MOV #77406,(R1) ;LET PAR7 BE 4K READ/WRITE NO BYPASS ALSO
RTS PC ;RETURN
    
```

GLOBAL SUBROUTINES SECTION

```

4423 ;**
4424 ; FUNCTIONAL DESCRIPTION:
4425 ; SUBROUTINE TO HANDLE PARITY ERROR ABORTS FROM THE RAM STORE RAM TESTS.
4426 ;
4427 ; INPUTS:
4428 ; MEMORY SYSTEM ERROR REGISTER CONTAINS BITS INDICATING FAILURE
4429 ;
4430 ; OUTPUTS: NONE
4431 ;
4432 ; SUBORDINATE ROUTINES USED: NONE
4433 ;
4434 ; FUNCTIONAL SIDE EFFECTS: NONE
4435 ;
4436 ; CALLING SEQUENCE: CALLED BY PARITY ABORT
4437 ; MOV @#114, SLOC00 ;SAVE CONTENTS OF PARITY ABORT VECTOR
4438 ; MOV #DSPAR, @#114 ;LET VECTOR POINT TO PARITY ABORT ROUTINE
4439 ;
4440 ; (CACHE PARITY ERROR OCCURS)
4441 ;
4442 027464 011637 001122 RAMPAR: MOV (SP), $BDADR ;STOR ADDESS TRAPPED
4443 027470 032737 000100 177744 BIT #BIT06, MSER ;IF LOW BYTE PARITY ERROR
4444 027476 001401 BEQ 1$ ;THEN
4445 027500 104004 ERROR +4 ;ERROR
4446 027502 1$: BIT #BIT07, MSER ;IF HIGH BYT
4447 E PARITY ERROR ; BEQ 2$ ;THEN
4448 ; ERROR +4 ;ERROR
4449 ;
4450 027502 032737 000040 177744 2$: BIT #BIT05, MSER ;IF TAG PARITY ERROR
4451 027510 001401 BEQ 3$ ;THEN
4452 027512 104004 ERROR +4 ;ERROR
4453 027514 005037 177744 3$: CLR MSER ;INITIALIZE MSER AFTER ERROR
4454 027520 000002 RTI ;RETURN
4455 027522 005237 002402 LKSINT: INC LKSFL ;INCREMENT FLAG
4456 027526 000002 RTI
4457 ;
4458 027530 clkcnt:
4459 027530 022737 000003 002404 100$: cmp #3, lkcnt
4460 027536 001402 beq 101$
4461 027540 005237 002404 inc lkcnt
4462 027544 000002 101$: rti
4463 ;
4464 ;.SBTTL Q22BE SIZE ROUTINE
4465 ;THIS ROUTINE WILL AUTOSIZE FOR UP TO TWO Q22 BUS EXERCISERS. IF NONE
4466 ;FOUND LOCATIONS CSR1 AND CSR12 WILL BE LEFT ZEROES. THIS ROUTINE WILL
4467 ;ONLY RUN IN NOT UFD MODE.
4468 ;
4469 027546 032737 001000 177750 Q22SIZ: BIT #BIT09, MAIREG ;UNIBUS SYSTEM?
4470 027554 001401 BEQ 1$ ;IF NOT, ADVANCE TO ROUTINE
4471 027556 000207 RTS PC ;OTHERWISE, RETURN
4472 ;
4473 ; PREPARE TO DO SIZING
4474 ;
4475 027560 013701 000004 1$: MOV ERRVEC, R1 ;STORE TIMEOUT VECTOR
4476 027564 012737 027740 000004 MOV #7$, ERRVEC ;POINT NEW TO PROGRAM
4477 027572 012737 000340 000006 MOV #340, ERRVEC+2 ;AT PRIORITY 7
4478 027600 005037 001160 CLR $TMP0 ;CLEAR Q22BE COUNTER
4479 027604 012702 170000 MOV #170000, R2 ;FIRST POSSIBLE ADDRESS

```


L10

Q22BE SIZE ROUTINE

```

4480 027610 012703 000510          MOV    #510,R3          ;VECTOR FOR IT
4481 027614 000404          BR     3$              ;TRY THOSE VALUES
4482                               ;
4483                               ; NOW DO ACTUAL SIZING
4484                               ;
4485 027616 062702 000020      2$:   ADD    #20,R2          ;GET CSR FOR NEXT Q22BE
4486 027622 062703 000004      ADD    #4,R3           ;GET VECTOR FOR NEXT ONE
4487 027626 005712          3$:   TST    (R2)         ;TRY TO ACCESS CSR
4488                               ;
4489                               ; IF NO TIMEOUT, STORE EXISTING ADDRESSES TO REGISTERS
4490                               ;
4491 027630 005737 001160      TST    $TMP0          ;FIRST Q22BE FOUND?
4492 027634 001010          BNE    4$              ;IF SECOND, BRANCH
4493 027636 012705 002334      MOV    #CSR1,R5       ;START WITH CSR1 FOR 1ST
4494 027642 010237 002352      MOV    R2,SIMGOA     ;SIMULTANEOUS GO
4495 027646 062737 000016 002352  ADD    #16,SIMGOA    ;ADDRESS
4496 027654 000402          BR     5$              ;BRANCH TO INITIALISE
4497 027656 012705 002364      4$:   MOV    #CSR12,R5  ;START WITH CSR12 FOR 2ND
4498 027662 012704 000004      5$:   MOV    #4,R4       ;INITIALISE 5 REGISTERS
4499 027666 010215          MOV    R2,(R5)        ;INITIALISE CSR1
4500 027670 011565 000002      6$:   MOV    (R5),2(R5)  ;STORE TO NEXT ONE
4501 027674 005725          TST    (R5)+         ;GET NEXT ADDRESS
4502 027676 062715 000002      ADD    #2,(R5)       ;GET ADDRESS, POINT NEXT
4503 027702 077406          SOB    R4,6$         ;DO FOR NEXT 4 REGISTERS
4504 027704 010365 000002      MOV    R3,2(R5)     ;STORE INTERRUPT VECTOR
4505 027710 010365 000004      MOV    R3,4(R5)     ;AND PRIORITY
4506 027714 062765 000002 000004  ADD    #2,4(R5)
4507 027722 005237 001160      INC    $TMP0         ;COUNT Q22BE'S
4508 027726 022737 000002 001160  CMP    #2,$TMP0     ;TWO FOUND?
4509 027734 001406          BEQ    9$             ;IF SO, STOP SIZING
4510 027736 000402          BR     8$             ;OTHERWISE, CONTINUE SIZING
4511                               ;
4512                               ; ON TIMEOUT TRY TO LOOK AT NEXT ADDRESS RANGE
4513                               ;
4514 027740 005726          7$:   TST    (SP)+       ;RESTORE STACK FROM
4515 027742 005726          TST    (SP)+       ;TIMEOUT
4516 027744 022702 170160      8$:   CMP    #170160,R2 ;AT THE LAST POSSIBLE?
4517 027750 001322          BNE    2$             ;IF NOT, BRANCH
4518 027752 005737 002334      9$:   TST    CSR1       ;1 FOUND?
4519 027756 001402          BEQ    10$          ;IF NONE, BRANCH
4520 027760 104401 027772      TYPE    ,ONQ22      ;TYPE FOUND
4521 027764 010137 000004      10$:  MOV    R1,ERRVEC   ;RESTORE TIMEOUT VECTOR
4522 027770 000207          RTS    PC            ;RETURN
4523                               ;
4524 027772          012      015      121  ONOQ22: .ASCIZ <12><15>/Q22BE USED DURING TESTING/
      027775          062      062      102
      030000          105      040      125
      030003          123      105      104
      030006          040      104      125
      030011          122      111      116
      030014          107      040      124
      030017          105      123      124
      030022          111      116      107
      030025          000
4525 .EVEN
4526 .SBTTL Q22BE INTERRUPT INITIALISE ROUTINE
4527 ;THIS ROUTINE WILL INITIALISE Q22BE TO INTERRUPT AT A PRIORITY AT (R0).

```

Q22BE INTERRUPT INITIALISE ROUTINE

```

4528 ;AT THE STARTING ADDRESS IN R3. THE TEST HAVE TO SET ACTUAL DONE BIT
4529 ;BY CLEARING GO.
4530
4531 030026 005013 Q22INT: CLR (R3) ;CLEAR TRANSFER TYPE IN CSR1
4532 030030 052710 000001 BIS #BIT00,(R0) ;ZERO DONE
4533 030034 011063 000002 MOV (R0),2(R3) ;SET PRIORITY IN CSR2
4534 030040 042710 000001 BIC #BIT00,(R0) ;PREPARE TO SET DONE
4535 030044 000207 RTS PC
4536
4537 .SBTTL DMATRN DATO CYCLE THRU Q22BE
4538 ;THIS ROUTINE PERFORMS DATO FROM A LOCATION TEMP THRU THE FIRST
4539 ;FOUND Q22BE STARTING AT LOCATION @CSR1. R0 HAS 0 IF ONLY 1 TRANSFER IS
4540 ;TO BE PERFORMED. OTHERWISE 16 BLOCK MODE TRANSFERS ARE TO BE PERFORMED.
4541 ;IN THE LATTER CASE ADDRESS AND WORD COUNT HAS TO BE LOADED BEFORE.
4542
4543 030046 012777 012525 152270 DMATRN: MOV #12525,@DATA ;DATA USED
4544 030054 005700 TST RO ;DO 1 WORD?
4545 030056 001404 BEQ 1$ ;IF YES, BRANCH
4546 030060 012777 001001 152250 MOV #BIT09!BIT00,@CSR2 ;BLOCK MODE, GO
4547 030066 000414 BR 2$ ;BRANCH TO DO IT
4548 030070 012777 001601 152236 1$: MOV #1601,@CSR1 ;RESET LATENCY COUNT,DATO
4549 030076 012777 002740 152234 MOV #TEMP,@BA ;LOAD DMA ADDRESS
4550 030104 012777 177777 152230 MOV #177777,@WC ;DO 1 WORD
4551 030112 012777 000001 152216 MOV #BIT00,@CSR2 ;DO IT
4552 030120 105777 152212 2$: TSTB @CSR2 ;DMA DONE?
4553 030124 100375 BPL 2$ ;WAIT TILL DONE
4554 030126 000207 3$: RTS PC ;RETURN FROM SUBROUTINE
4555
4556 .SBTTL DMARD DATI THRU Q22BE
4557 ;THIS ROUTINE PERFORMS DATI CYCLE THRU Q22BE IN EITHER BLOCK MODE OR A SINGLE
4558 ;TRANSFER MODE. MEMORY LOCATION USED IS TEMP. R0 IS ZERO FOR SINGLE TRANSFER
4559
4560 030130 012777 002740 152202 DMARD: MOV #TEMP,@BA ;LOAD DMA ADDRESS
4561 030136 005700 TST RO ;DO 1 WORD?
4562 030140 001412 BEQ 1$ ;IF YES, BRANCH
4563 030142 012777 001507 152164 MOV #1507,@CSR1 ;16 DATIB
4564 030150 012777 177770 152164 MOV #177770,@WC ;DO 8 WORD
4565 030156 012777 001001 152152 MOV #BIT09!BIT00,@CSR2 ;BLOCK MODE GO
4566 030164 000411 BR 2$ ;GO CHECK
4567 030166 012777 001407 152140 1$: MOV #1407,@CSR1 ;RESET LATENCY COUNT,DATI
4568 ;LOAD NEW DATA TO DATA R.
4569 030174 012777 177777 152140 MOV #177777,@WC ;DO 1 WORD
4570 030202 012777 000001 152126 MOV #BIT00,@CSR2 ;DO IT
4571 030210 105777 152122 2$: TSTB @CSR2 ;DMA DONE?
4572 030214 100375 BPL 2$ ;WAIT TILL DONE
4573 030216 000207 3$: RTS PC ;RETURN FROM SUBROUTINE
4574
4575
4576
4577 030220 011637 001122 TOUT: MOV (SP), $BDADR ;STORE TRAPPED PC
4578 030224 104041 ERROR +41 ;UNEXPECTED TRAP
4579 030226 000002 RTI
4580
4582 ;MMU GLOBAL SUBROUTINES
4583 ;
4584 ;
4585 ;

```

DMARD DATI THRU Q22BE

```

4586 ;ROUTINE TO INITIALIZE MEMORY MANAGEMENT
4587 ;
4588 030230 010046 MMU: MOV R0, (SP) ;SAVE CONTENTS OF REGISTERS
4589 030232 010146 MOV R1, (SP) ;
4590 030234 010246 MOV R2, (SP) ;
4591 030236 012700 177600 MOV #177600,R0 ;
4592 030242 004737 030330 JSR PC,PDR ;INIT I AND D USER PDR S
4593 030246 004737 030352 JSR PC,PAR ;INIT I USER PAR'S
4594 030252 004737 030352 JSR PC,PAR ;INIT D USER PAR'S
4595 030256 012700 172200 MOV #172200,R0 ;
4596 030262 004737 030330 JSR PC,PDR ;INIT I AND D SUP PDR'S
4597 030266 004737 030352 JSR PC,PAR ;INIT I SUP PAR'S
4598 030272 004737 030352 JSR PC,PAR ;INIT D SUP PAR'S
4599 030276 004737 030330 JSR PC,PDR ;INIT I AND D KER PDP S
4600 030302 004737 030352 JSR PC,PAR ;INIT I KER PAR'S
4601 030306 004737 030352 JSR PC,PAR ;INIT D KER PAR'S
4602 030312 012737 000027 172516 MOV #27,#172516 ;INIT MMR3
4603 030320 012602 MOV (SP)+,R2 ;RESTORE REGISTERS
4604 030322 012601 MOV (SP)+,R1 ;
4605 030324 012600 MOV (SP)+,R0 ;
4606 030326 000207 RTS PC ;RETURN
4607 ;
4608 ;ROUTINE TO INITIALIZE PDR'S
4609 ;
4610 030330 005002 PDR: CLR R2 ;INIT CNTR
4611 030332 012720 077406 PDR1: MOV #77406,(R0)+ ;INIT PDR
4612 030336 062702 000001 ADD #1,R2 ;INCREMENT CNTR
4613 030342 022702 000020 CMP #16.,R2 ;ARE WE DONE?
4614 030346 001371 BNE PDR1 ;BRANCH IF NOT
4615 030350 000207 RTS PC ;RETURN
4616 ;
4617 ;ROUTINE TO INITIALIZE PAR'S
4618 ;
4619 030352 005001 PAR: CLR R1 ;SETUP TO INIT PAR
4620 030354 010120 PAR1: MOV R1,(R0)+ ;INIT PAR
4621 030356 062701 000200 ADD #200,R1 ;GET READY FOR NEXT PAR
4622 030362 022701 001600 CMP #1600,R1 ;REACHED A PAR?
4623 030366 001372 BNE PAR1 ;BRANCH IF NOT
4624 030370 012720 177600 MOV #177600,(R0)+ ;INIT PAR?
4625 030374 000207 RTS PC ;RETURN
4626 ;
4627 ;TIME OUT ROUTINE
4628 ;
4629 030376 005205 ADDTRP: INC R5 ;INCREMENT TIME OUT FLAG
4630 030400 000002 RTI ;RETURN
4631 ;
4632 ;MMU TRAP ROUTINE
4633 ;
4634 030402 023727 003032 000001 MMUTRP: CMP FLAG,#1 ;ARE WE EXPECTING AN ABORT
4635 030410 001401 BEQ 1$ ;YES GO ON
4636 030412 104002 ERROR +2 ;NO GO TO ERROR
4637 030414 010046 1$: MOV R0, (SP) ;SAVE CONTENTS OF REG 0
4638 030416 013700 177776 MOV #177776,R0 ;SAVE A COPY OF PSW
4639 030422 072027 177764 ASH # 14,R0 ;LOOK AT BITS<15:14>
4640 030426 020027 000002 CMP R0,#2 ;WAS PS<15:14>=10
4641 030432 001001 BNE OK ;NO GO ON
4642 030434 000411 BR NOTOK ;YES CHANGE BITS TO 00

```

*6

JMARD JATI THRU Q22BE

4643	030436	013700	177776		OK:	MOV	@#177776,RO	;SAVE A COPY OF PSW
4644	030442	072027	000002			ASH	#2,RO	;LOOK AT BITS<13:12>
4645	030446	072027	177764			ASH	# 14,RO	:
4646	030452	020027	000002			CMP	RO,#2	;WAS PS<13:12>=10
4647	030456	001002				BNE	OK1	;NO GO ON
4648	030460	005066	000004		NOTOK:	CLR	4(SP)	;CLEAR ILLEGAL MODE FROM OLD PSW
4649	030464	013737	177572	003044	OK1:	MOV	@#177572,SAVMRO	;SAVE A COPY OF MMRO
4650	030472	013737	177574	003046		MOV	@#177574,SAVMR1	;SAVE A COPY OF MMR1
4651	030500	013737	177576	003050		MOV	@#177576,SAVMR2	;SAVE A COPY OF MMR2
4652	030506	005037	177572			CLR	@#177572	;CLEAR ABORT BITS AND TURN MM. OFF
4653	030512	005037	003032			CLR	FLAG	;CLEAR MMU ABORT FLAG
4654	030516	012600				MOV	(SP)+,RO	;RESTORE ORIGINAL CONTENTS OF REG 0
4655	030520	000002				RTI		;RETURN

C1.

DMARD DATI THRU Q228E

```

4659          ;PP COMMON SUBROUTINES
4660 030522 012600  ALDTRP: MOV      (SP)+,R0      ;SAVE PC
4661 030524 012605          MOV      (SP)+,R5      ;SAVE STATUS AND RESTORE STACK
4662 030526 104003          ERROR   +3
4663 030530 000110          JMP      (R0)          ;GO BACK INLINE
4664
4665          ;
4666          ;
4667 030532 000000  TRPFLG: .WORD   0
4668 030534 000207  ERRFP:  RTS     R7
4669 030536 000207  ERR:   RTS     R7
4670
4671          ;
4672          ;
4673          ;
4674          ;
4675          ;SUBROUTINE DATA VERFICATION
4676          ;
4677          ; CALLED BY      JSR     R7,DATVER
4678          ;
4679          ;INPUT:        (R4)=EXPECTED DATA
4680          ;              (R1)=RECEIVED DATA
4681          ;
4682          ;THIS ROUTINE VERIFIES THAT THE 4 CONSECUTIVE WORDS STARTING WITH (R4) ARE
4683          ;EQUAL TO THE FOUR WORDS ADDRESSED BY (R1). THE CONTENTS OF R4, AND R1 ARE NOT
4684          ;DISTURBED.
4685          ;LOCATION "COUNT" , IF NOT EQUAL TO 0 SIGNIFIES DATA ERROR
4686          ;IF THE STATUS IS FLOATING MODE, THE LAST TWO BYTES OF RECEIEVED
4687          ;ARE SIMPLY CHECKED FOR ZEROS
4688          ;
4689          ;
4690 030540 010446  DATVFR: MOV      R4, (SP)      ;SAVE R4
4691 030542 010146          MOV      R1, -(SP)      ;SAVE R1
4692 030544 012737 000003 003124  MOV      #3,COUNT    ;SET UP ITERATION COUNT
4693 030552 000137 030570          JMP      DAT1
4694          ;
4695 030556 010446  DATVER: MOV      R4, -(SP)      ;SAVE R4
4696 030560 010146          MOV      R1, (SP)      ;SAVE R1
4697 030562 012737 000005 003124  MOV      #5,COUNT    ;SET UP ITERATION COUNT
4698 030570 005337          DAT1:  DEC      COUNT
4699 030574 001402          BEQ      2$,          ;BRANCH IF DONE
4700 030576 022421          CMP      (R4)+,(R1)+
4701 030600 001773          BEQ      DAT1
4702 030602 012601  2$:   MOV      (SP)+,R1      ;RESTORE R1
4703 030604 012604          MOV      (SP)+,R4      ;RESTORE R4
4704 030606 000207          RTS      R7          ;GO BACK TO CALLING ROUTINE
4705          ;IF DATA ERROR, COUNT NE 0

```

D11

DMARD DATI THRU Q228E

```

470/
4708 ; .....
4709 ; DELAY INTERRUPT TIMER
4710 ; This test 's called by the interrupt receiver and transmitter
4711 ;
4712 ;
4713 ;
4714 DELAY:
4715 030610 013746 000004 MOV @#4,(SP) ;SAVE
4716 030614 013746 000006 MOV @#6,-(SP) ; OLD
4717 030620 010046 MOV R0,(SP) ; STACK
4718 030622 010146 MOV R1,(SP) ; AND REGS
4719
4720 030624 012737 030652 000004 MOV #2,@#4 ;SET TRAP
4721 030632 106737 000006 MFPS @#6 ;PUT PSW ON PS / SAME PRIORITY
4722
4723 030636 012700 000001 MOV #1.,R0 ;SET OUTER LOOP COUNT
4724
4725 030642 012701 011610 3$: MOV #5000.,R1 ;SET INNER LOOP 1/200TH
4726
4727 030646 005737 160000 1$: TST @#160000 ;TAKES ABOUT 10u SEC
4728
4729 030652 022626 2$: CMP (SP)+,(SP)+ ;CLEAN UP STACK
4730 030654 077104 SOB R1,1$ ;DO INNER LOOP
4731 030656 077007 SOB R0,3$ ;DO OUTER LOOP
4732
4733 030660 012601 MOV (SP)+,R1 ;RESTORE
4734 030662 012600 MOV (SP)+,R0 ; WHAT
4735 030664 012637 000006 MOV (SP)+,@#6 ; WE
4736 030670 012637 000004 MOV (SP)+,@#4 ; SAVED
4737
4738 030674 000207 RTS PC
4739

```

E11

DMARD DATI THRU Q22BE

```

4741
4742
4743
4744
4745
4746
4747
4748 030676
4749 030676 012737 000000 172340
4750 030704 012737 000200 172342
4751 030712 012737 000400 172344
4752 030720 012737 000600 172346
4753 030726 012737 001000 172350
4754 030734 012737 001200 172352
4755 030742 012737 001400 172354
4756 030750 012737 177600 172356
4757 030756 012702 000010
4758 030762 012701 172300
4759 030766 012721 077406
4760 030772 077203
4761 030774 012737 000001 177572
4762 031002 012737 000020 172516
4763 031010 000207

```

```

.....
: SETMMU SET UP THE MMU REGISTERS
:
: This test is called by the memory test to set up the PARS so
: all of memory is accessed
:
:-----
SETMMU: SUBR SET UP MMU REGISTERS
: POINT TO PAGE 0 TO ITSELF
MOV #0,@#KIPAR0 : POINT TO PAGE 1 TO ITSELF
MOV #200,@#KIPAR1 : POINT TO PAGE 2 TO ITSELF
MOV #400,@#KIPAR2 : POINT TO PAGE 3 TO ITSELF
MOV #600,@#KIPAR3 : POINT TO PAGE 4 TO ITSELF
MOV #1000,@#KIPAR4 : POINT TO PAGE 5 TO ITSELF
MOV #1200,@#KIPAR5 : POINT TO PAGE 6 TO ITSELF
MOV #1400,@#KIPAR6 : POINT I/O PAGE TO IT'S PLACE IN 22 BIT
MOV #177600,@#KIPAR7: BGNDD
MOV #10,R2 :
MOV #KIPDR0,R1 : SET UP PDR'S TO 4K R/W
10$: MOV #77406,(R1) :
SOB R2,10$ : DOUNTIL WE HAVE INITIALIZED ALL OF THEM
MOV #1,@#SRO : ENABLE MEMORY MANAGEMENT
MOV #BIT4,@#SR3 : ENABLE 22 BIT ADDRESSING
RTS PC : ENDSUBR SET UP MMU REGISTERS

```

F I L

DMARD DATI THRU Q22BE

4765
4766
4767
4768
4769
4770
4771
4772
4773
4774
4775
4776
4777
4778
4779
4780
4781
4782
4783
4784 031012
4785 031012 012737 001600 172344
4786 031020 012701 040000
4787 031024
4788 031024 112711 177777
4789 031030 005201
4790 031032 020127 060000
4791 031036 103413
4792 031040 062737 000200 172344
4793 031046 042701 160000
4794 031052 052701 040000
4795 031056 023727 172344 020000
4796 031064 001443
4797 031066 112711 000000
4798 031072 005201
4799 031074 020127 060000
4800 031100 103415
4801 031102 062737 000200 172344
4802 031110 042701 160000
4803 031114 052701 040000
4804 031120 023727 172344 020000
4805 031126 001422
4806 031130 112711 000000
4807 031134 005201
4808 031136 020127 060000
4809 031142 103730
4810 031144 062737 000200 172344
4811 031152 042701 160000
4812 031156 052701 040000
4813 031162 023727 172344 020000
4814 031170 001401
4815 031172 000714
4816 031174
4817 031174 000207
4818
4819
4820

```
.....
; INIMEM  FILL UP MEMORY
;
; This test is called by the Quick verify memory test. It initializes
; memory to look like the following.
;
;   Address      Data
;   -----
;   0             377
;   1             0
;   2             0
;   3             377
;   4             0
;   5             0
;
;   .
;   .
;   .
```

```
-----
INIMEM:
MOV #1600,@#KIPAR2
MOV #40000,R1
10$:
MOV #1,(R1)
INC R1
CMP R1,#60000
BLO 20$
ADD #200,@#KIPAR2
BIC #160000,R1
BIS #40000,R1
CMP @#KIPAR2,#20000
BEQ 40$
20$:
MOV #0,(R1)
INC R1
CMP R1,#60000
BLO 30$
ADD #200,@#KIPAR2
BIC #160000,R1
BIS #40000,R1
CMP @#KIPAR2,#20000
BEQ 40$
30$:
INC R1
CMP R1,#60000
BLO 10$
ADD #200,@#KIPAR2
BIC #160000,R1
BIS #40000,R1
CMP @#KIPAR2,#20000
BEQ 40$
BR 10$
40$:
50$:
RTS PC
;
; SUBR INITIALIZE MEMORY
; POINT KPAR2 TO LOW ADDRESS
; SET VIRTUAL ADDRESS TO MAP THRU KPAR2
; BGND0
; WRITE A 1 TO 0 OFFSET ADDRESS
;
; IF WE HAVE PASSED THE PAGE BOUNDARY
; THEN
; POINT KPAR2 TO A NEW PAGE
; CLEAR OUT THE PAGE POINTER
; POINT VIRTUAL ADDRESS TO KPAR2
; ENDIF
; DOUNTIL WE HAVE READ ALL ADDRESSES
; WRITE A 0 TO 1 OFFSET ADDRESS
;
; IF WE HAVE PASSED THE PAGE BOUNDARY
; THEN
; POINT KPAR2 TO A NEW PAGE
; CLEAR OUT THE PAGE POINTER
; POINT VIRTUAL ADDRESS TO KPAR2
; ENDIF
; DOUNTIL WE HAVE READ ALL ADDRESSES
; WRITE A 0 TO 2 OFFSET ADDRESS
; BUMP UP ADDRESS BY 3
; IF WE HAVE PASSED THE PAGE BOUNDARY
; THEN
; POINT KPAR2 TO A NEW PAGE
; CLEAR OUT THE PAGE POINTER
; POINT VIRTUAL ADDRESS TO KPAR2
; ENDIF
; DOUNTIL WE HAVE READ ALL ADDRESSES
;
; ENDIF
; ENDSUBR INITIALIZE MEMORY
```


DMARD DATI THRU Q228E

```

4822
4823
4824
4825
4826
4827
4828
4829
4830
4831
4832
4833
4834
4835
4836 031176
4837 031176 012737 001600 172344
4838 031204 012701 040000
4839 031210 012702 000003
4840 031214 160402
4841 031216
4842 031216 060401
4843 031220 020127 060000
4844 031224 103413
4845 031226 062737 000200 172344
4846 031234 042701 160000
4847 031240 052701 040000
4848 031244 023727 172344 020000
4849 031252 001435
4850 031254 005703
4851 031256 001012
4852 031260 120511
4853 031262 001411
4854 031264 010537 001124
4855 031270 111137 001126
4856 031274 010137 001122
4857 031300 104066
4858 031302 000401
4859 031304
4860 031304 110511
4861 031306
4862 031306 060201
4863 031310 020127 060000
4864 031314 103740
4865 031316 062737 000200 172344
4866 031324 042701 160000
4867 031330 052701 040000
4868 031334 023727 172344 020000
4869 031342 001401
4870 031344 000724
4871 031346 000207

```

```

;*****
; RWTMEM READ OR WRITE TO THE MEMORY
;
; This test is called by the Quick verify memory test to Read or
; Write to memory.
;
; R3 = OPERATION TO BE PERFORMED
;
; 0 - READ
; 1 - WRITE
;
; R4 = THE ADDRESS INCREMENT
; R5 = THE EXPECTED PATTERN
;
RWTMEM:
MOV #1600,@#KIPAR2
MOV #40000,R1
MOV #3,R2
SUB R4,R2
10$:
ADD R4,R1
CMP R1,#60000
BLO 15$
ADD #200,@#KIPAR2
BIC #160000,R1
BIS #40000,R1
CMP @#KIPAR2,#20000
BEQ 40$
15$:
TST R3
BNE 20$
CMPB R5,(R1)
BEQ 30$
MOV R5,$GDDAT
MOVB (R1),$BDDAT
MOV R1,$BDADR
ERROR +66
BR 30$
20$:
MOVB R5,(R1)
30$:
ADD R2,R1
CMP R1,#60000
BLO 10$
ADD #200,@#KIPAR2
BIC #160000,R1
BIS #40000,R1
CMP @#KIPAR2,#20000
BEQ 40$
BR 10$
40$:
RTS PC

```

```

; SUBR READ WRITE MEMORY
; POINT TO KPAR2 TO LOW ADDRESS
; POINT TO PAGE 2 VIRTUAL ADDRESS
; WE ARE BUMPING UP BY 3
; SUBTRACT OFF ADDRESS INCREMENT
; BGNDD
; : GET ADDRESS TO LOOK AT
; : IF WE HAVE PASSED THE PAGE BOUNDARY
; : THEN
; : : POINT KPAR2 TO A NEW PAGE
; : : CLEAR OUT THE PAGE BITS
; : : SET THEM BACK TO PAGE 2
; : : ENDDIF
; : IF THE OPERATION IS A READ
; : : THEN
; : : IF CONTENTS <> EXPECTED PATTERN
; : : : THEN
; : : : GET THE EXPECTED PATTERN
; : : : GET THE RECIEVED PATTERN
; : : : GET THE VIRTUAL ADDRESS
; : : : ERROR IN MEMORY
; : : : ENDDIF
; : : ELSE
; : : : WRITE PATTERN TO MEMORY
; : : : ENDDIF
; : : ADD ON DIFFERENCE TO GET ADDRESS+3
; : : IF WE HAVE PASSED THE PAGE BOUNDARY
; : : : THEN
; : : : POINT KPAR2 TO A NEW PAGE
; : : : CLEAR OUT THE PAGE BITS
; : : : SET PAGE BITS BACK TO PAGE 2
; : : : ENDDIF
; : DOUNTIL WE HAVE READ ALL ADDRESSES
; ENDSUBR READ WRITE_MEMORY

```

DMARD DATI THRU Q228E

```

4873
4874
4875
4876
4877
4878
4879
4880
4881 031350
4882 031350 012737 001600 172344
4883 031356 012701 040000
4884 031362
4885 031362 111137 001126
4886 031366 122737 177777 001126
4887 031374 001406
4888 031376 012737 177777 001124
4889 031404 010137 001122
4890 031410 104066
4891 031412 005201
4892 031414 020127 060000
4893 031420 103413
4894 031422 062737 000200 172344
4895 031430 042701 160000
4896 031434 052701 040000
4897 031440 023727 172344 020000
4898 031446 001435
4899 031450 111137 001126
4900 031454 122737 177777 001126
4901 031462 001406
4902 031464 112737 177777 001124
4903 031472 011137 001122
4904 031476 104066
4905 031500 062701 000002
4906 031504 020127 060000
4907 031510 103724
4908 031512 062737 000200 172344
4909 031520 042701 160000
4910 031524 052701 040000
4911 031530 023727 172344 020000
4912 031536 001401
4913 031540 000710
4914 031542 000207
4915
  
```

```

.....
; RRMEM - READ THE MEMORY
;
; This test is called by the memory test write to memory
;
; R4 = THE ADDRESS INCREMENT
; R5 = THE EXPECTED PATTERN
;
RRMEM:
MOV #1600,@KIPAR2
MOV #40000,R1
10$:
MOVB (R1), $BDDAT
CMPB #-1, $BDDAT
BEQ 15$
MOV #-1, $GDDAT
MOV R1, $BDADR
ERROR +66
15$:
INC R1
CMP R1, #60000
BLO 20$
ADD #200, @KIPAR2
BIC #160000, R1
BIS #40000, R1
CMP @KIPAR2, #20000
BEQ 40$
20$:
MOVB (R1), $BDDAT
CMPB #-1, $BDDAT
BEQ 30$
MOVB #-1, $GDDAT
MOV (R1), $BDADR
ERROR +66
30$:
ADD #2, R1
CMP R1, #60000
BLO 10$
ADD #200, @KIPAR2
BIC #160000, R1
BIS #40000, R1
CMP @KIPAR2, #20000
BEQ 40$
BR 10$
40$:
RTS PC
;
; SUBR READ LOCATIONS 0,1
; LET PAR POINT TO LOW ADDRESS
; POINT TO KPAR2 VIRTUAL ADDRESS
; BGND0
; GET A COPY OF RECIEVED PATTERN
; IF 0 OFFSET ADDRESS <> -1 THEN
; ERROR IN MEMORY
; GET THE EXPECTED PATTERN
; GET THE VIRTUAL ADDRESS
;
; POINT TO 1 OFFSET ADDRESS
; IF WE HAVE PASSED THE PAGE BOUNDARY
; THEN
; POINT KPAR2 TO A NEW PAGE
; CLEAR THE PAGE BITS
; POINT BACK TO PAGE 2
; ENDF
;
; GET THE RECIEVED PATTERN
; IF 1 OFFSET ADDRESS <> 1 THEN
; ERROR IN MEMORY
; GET THE EXPECTED PATTERN
; GET THE VIRTUAL ADDRESS
;
; ADD 2 TO ADDRESS TO GET ADDRESS + 3
; IF WE HAVE PASSED THE PAGE BOUNDARY
; THEN
; POINT KPAR2 TO A NEW PAGE
; CLEAR THE PAGE BITS
; POINT BACK TO PAGE 2
; ENDF
; DUNTIL WE HAVE READ ALL ADDRESSES
;
; ENDSUBR READ LOCATIONS 0,1
  
```

111

DMARD DATI THRU Q22BE

```

4917
4918
4919
4920
4921
4922
4923
4924
4925
4926
4927
4928
4929
4930
4931
4932
4933
4934
4935
4936
4937
4938
4939
4940 031544
4941 031544 104401 031552
4942 031550 000207
4943
4944 031552 105 122 122
      031555 117 122 040
      031560 104 105 124
      031563 105 103 124
      031566 105 104 040
      031571 111 116 040
      031574 112 061 061
      031577 040 106 114
      031602 117 101 124
      031605 111 116 107
      031610 040 120 117
      031613 111 116 124
      031616 040 120 122
      031621 117 103 105
      031624 123 123 117
      031627 122 056 000
4945
4946

```

```

: $$$
: SUBROUTINE DETERMINE FLOATING POINT ACCELERATOR (DETFPA)
: THIS SUBROUTINE IS CALLED IF AN ERROR IS DETECTED DURING EXECUTION OF THE
: FLOATING POINT TESTS.
: IT DETERMINES WHEATHER OR NOT THE FLOATING POINT ACCELERATOR CHIP OPTION
: IS PRESENT ON THE CPU BOARD AND PRINTS THE APPROPRIATE ERROR MESSAGE.
: THIS DETERMINATION IS MADE BASED ON THE "FPA AVAILABLE" FLAG, BIT 8
: OF THE MAINTENANCE REGISTER AT LOCATION 17777750. IF THE FPA BIT IS SET
: THEN THE FLOATING POINT ACCELERATOR CHIP IS INSTALLED ON THE CPU BOARD AND
: AN ERROR MESSAGE IS PRINTED WHICH STATES THAT THE FLOATING POINT ERROR IS
: DUE TO THIS CHIP. OTHERWISE, THE J11 IS BLAMED FOR THE FLOATING POINT ERROR.
: $$$
: CALLED BY: CALL @#DETFPA ; $$$
: INPUTS: NONE ; $$$
: OUTPUTS: ERROR MESSAGES ; $$$
:
DETFPA:
      TYPE ,J11FLT ; $$$
      RTS PC ; $$$
:
J11FLT: .ASCIZ /ERROR DETECTED IN J11 FLOATING POINT PROCESSOR./ ; $$$
:
      .EVEN ; $$$
:

```

DMARD DATI THRU Q228E

4949
 4950
 4951

```

.SBTTL END OF PASS ROUTINE
;*****
;*INCREMENT THE PASS NUMBER ($PASS)
;*INDICATE END-OF PROGRAM AFTER 1 PASSES THRU THE PROGRAM
;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
;*IF THERES A MONITOR GO TO IT
;*IF THERE ISN'T JUMP TO LOOP
$EOP:
031632 032737 0' 100 000052 BIT #BIT06,@#52
031632 001030 BNE $GET42
031640 004537 C17724 jsr r5,v'reop
031642 005037 J01102 CLR $TSTNM ;;ZERO THE TEST NUMBER
031646 005037 J01102 CLR $TIMES ;;ZERO THE NUMBER OF ITERATIONS
031652 005037 J01164 INC $PASS ;;INCREMENT THE PASS NUMBER
031656 005237 001206 BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
031662 042737 100000 001206 DEC (PC)+ ;;LOOP?
031670 005327 $EOPCT: .WORD 1
031672 000001 BGT $DOAGN ;;YES
031674 003022 MOV (PC)+,@(PC)+ ;;RESTORE COUNTER
031676 012737 $ENDCT: .WORD 1
031700 000001 $EOPCT
031702 031672 TYPE , $ENDMG ;;TYPE "END PASS #"
031704 104401 031751 MOV $PASS, (SP) ;;SAVE $PASS FOR TYPEOUT
031710 013746 001206 TYPDS ;;GO TYPE DECIMAL ASCII WITH SIGN
031712 104405 TYPE , $ENULL ;;TYPE A NULL CHARACTER
031716 104401 031746 $GET42: MOV @#42,R0 ;;GET MONITOR ADDRESS
031722 013700 000042 BEQ $DOAGN ;;BRANCH IF NO MONITOR
031726 001405 RESET ;;CLEAR THE WORLD
031730 000005 $ENDAD: JSR PC,(R0) ;;GO TO MONITOR
031732 004710 NOP ;;SAVE ROOM
031734 000240 NOP ;;FOR
031736 000240 NOP ;;ACT11
031740 000240 $DOAGN: JMP @(PC)+ ;;RETURN
031742 000137 $RTNAD: .WORD LOOP
031744 005064 $ENULL: .BYTE 1,-1,0 ;;NULL CHARACTER STRING
031746 377 377 000 $ENDMG: .ASCIZ <15><12>/END PASS #/
031751 015 012 105
031754 116 104 040
031757 120 101 123
031762 123 040 043
031765 000

```

4952

```

.SBTTL SCOPE HANDLER ROUTINE
;*****
;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
;*SW14=1 LOOP ON TEST
;*SW11=1 INHIBIT ITERATIONS
;*SW09=1 LOOP ON ERROR
;*SW08=1 LOOP ON TEST IN SWR<5:0>
;*CALL
;* SCOPE ;;SCOPE=IOT
$SCOPE: CKSWR ;;TEST FOR CHANGE IN SOFT SWR

```

031766
 031766 104407

SCOPE HANDLER ROUTINE

```

031770 052737 001000 177520      BIS #1000,BCSR ;ENABLE
031776 032777 040000 147134 1$: BIT #BIT14,@SWR      ;;LOOP ON PRESENT TEST?
032004 001117      BNE $OVER          ;;YES IF SW14=1
                                ;#####START OF CODE FOR THE XOR TESTER#####
032006 000416      $XTSTR: BR 6$      ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
032010 013746 000004      MOV @#ERRVEC, (SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
032014 012737 032034 000004      MOV #5,@#ERRVEC    ;;SET FOR TIMEOUT
032022 005737 177060      TST @#177060       ;;TIME OUT ON XOR?
032026 012637 000004      MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
032032 000466      BR $SVLAD          ;;GO TO THE NEXT TEST
032034 022626      5$: CMP (SP)+,(SP)+ ;;CLEAR THE STACK AFTER A TIME OUT
032036 012637 000004      MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
032042 000426      BR 7$             ;;LOOP ON THE PRESENT TEST
032044      6$:;#####END OF CODE FOR THE XOR TESTER#####
032044 032777 000400 147066      BIT #BIT08,@SWR    ;;LOOP ON SPEC. TEST?
032052 001407      BEQ 2$             ;;BR IF NO
032054 017746 147060      MOV @SWR, (SP)     ;;SET DESIRED TEST NUM. FROM SWR
032060 042716 000300      BIC #SWRMK,(SP)   ;;STRIP AWAY UNDESIED BITS
032064 122637 001102      CMPB (SP)+,$TSTNM ;;ON THE RIGHT TEST?
032070 001465      BEQ $OVER         ;;BR IF YES
032072 105737 001103      2$: TSTB $ERFLG   ;;HAS AN ERROR OCCURRED?
032076 001421      BEQ 3$             ;;BR IF NO
032100 123737 001115 001103      CMPB $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
032106 101015      BHI 3$             ;;BR IF NO
032110 032777 001000 147022      BIT #BIT09,@SWR   ;;LOOP ON ERROR?
032116 001404      BEQ 4$             ;;BR IF NO
032120 013737 001110 001106      7$: MOV $LPERR,$LPADR ;;SET LOOP ADDRESS TO LAST SCOPE
032126 000446      BR $OVER
032130 105037 001103      4$: CLRB $ERFLG   ;;ZERO THE ERROR FLAG
032134 005037 001164      CLR $TIMES        ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
032140 000415      BR 1$             ;;ESCAPE TO THE NEXT TEST
032142 032777 004000 146770      3$: BIT #BIT11,@SWR  ;;INHIBIT ITERATIONS?
032150 001011      BNE 1$             ;;BR IF YES
032152 005737 001206      TST $PASS         ;;IF FIRST PASS OF PROGRAM
032156 001406      BEQ 1$             ;;INHIBIT ITERATIONS
032160 005237 001104      INC $ICNT         ;;INCREMENT ITERATION COUNT
032164 023737 001164 001104      CMP $TIMES,$ICNT  ;;CHECK THE NUMBER OF ITERATIONS MADE
032172 002024      BGE $OVER         ;;BR IF MORE ITERATION REQUIRED
032174 012737 000001 001104      1$: MOV #1,$ICNT   ;;REINITIALIZE THE ITERATION COUNTER
032202 013 37 032266 001164      MOV $MXCNT,$TIMES ;;SET NUMBER OF ITERATIONS TO DO
032210 105237 001102      $SVLAD: INCB $TSTNM ;;COUNT TEST NUMBERS
032214 113737 001102 001204      MOVB $TSTNM,$TESTN ;;SET TEST NUMBER IN APT MAILBOX
032222 011637 001106      MOV (SP),$LPADR   ;;SAVE SCOPE LOOP ADDRESS
032226 011637 001110      MOV (SP),$LPERR  ;;SAVE ERROR LOOP ADDRESS
032232 005037 001166      CLR $ESCAPE      ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
032236 112737 000001 001115      MOVB #1,$ERMAX   ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
032244 013777 001102 146670      $OVER: MOV $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
032252 013716 001106      MOV $LPADR,(SP)  ;;FUJGE RETURN ADDRESS
032256 042737 001000 177520      BIC #1000,BCSR ;DISABLE
032264 000002      RTI
032266 000001      $MXCNT: 1      ;;MAX. NUMBER OF ITERATIONS

```

4953

```

.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 ERTYPE ON ERROR

```

ERROR HANDLER ROUTINE

```

; *THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
; *SW15=1      HALT ON ERROR
; *SW13=1      INHIBIT ERROR TYPEOUTS
; *SW10=1      BELL ON ERROR
; *SW09=1      LOOP ON ERROR
; *CALL
; *          ERROR  +N      ;;ERROR=EMT AND N ERROR ITEM NUMBER
$ERROR:
032270          TST      UQUIET      ;;TEST FOR USER-QUIET MODE
032270 005737 004154      BEQ      9$      ;;BRANCH IF FIELD SERVICE MODE
032274 001403          CLR      R0      ;;IN CASE R0 HAS A #3 IN IT (+C)
032276 005000          JSR      PC,ABORT  ;;TEST FOR ABORT CONDITION
032300 004737 032512      9$:
032304          CKSWR          ;;TEST FOR CHANGE IN SOFT SWR
032304 104407          BIS      #1000,BCSR  ;ENABLE HALT ON BREAK
032306 052737 001000 177520      7$:      $ERFLG      ;;SET THE ERROR FLAG
032314 105237 001103          BEQ      7$      ;;DON'T LET THE FLAG GO TO ZERO
032320 001775          MOV      $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
032322 013777 001102 146612      BIT      #BIT10,@SWR  ;;BELL ON ERROR?
032330 032777 002000 146602      BEQ      1$      ;;NO SKIP
032336 001402          TYPE      , $BELL      ;;RING BELL
032340 104401 001170          INC      $ERTTL      ;;COUNT THE NUMBER OF ERRORS
032344 00237 001112          MOV      (SP), $ERRPC      ;;GET ADDRESS OF ERROR INSTRUCTION
032350 011637 001116          SUB      #2, $ERRPC
032354 162737 000002 001116      MOV      @ $ERRPC, $ITEMB      ;;STRIP AND SAVE THE ERROR ITEM CODE
032362 117737 146530 001114      BIT      #BIT13,@SWR      ;;SKIP TYPEOUT IF SET
032370 032777 020000 146542      BNE      20$      ;;SKIP TYPEOUTS
032376 001004          JSR      PC,ERT/PE      ;;GO TO USER ERROR ROUTINE
032400 004737 027030          TYPE      , $CRLF
032404 104401 001175      20$:
032410          CMP      #APTENV, $ENV      ;;RUNNING IN APT MODE
032410 122737 000001 001220      BNE      2$      ;;NO, SKIP APT ERROR REPORT
032416 001007          MOV      $ITEMB, 21$      ;;SET ITEM NUMBER AS ERROR NUMBER
032420 113737 001114 032432      JSR      PC, $ATY4      ;;REPORT FATAL ERROR TO APT
032426 004737 032670      21$:
032432          .BYTE      0
032433          .BYTE      0
032434 000777          BR      22$      ;;APT ERROR LOOP
032436 005777 146476      22$:
032442 100002          TST      @SWR      ;;HALT ON ERROR
032444 000000          BPL      3$      ;;SKIP IF CONTINUE
032446 104407          HALT          ;;HALT ON ERROR!
032450 032777 001000 146462      3$:      CKSWR          ;;TEST FOR CHANGE IN SOFT-SWR
032456 001402          BIT      #BIT09,@SWR      ;;LOOP ON ERROR SWITCH SET?
032460 013716 001110          BEQ      4$      ;;BR IF NO
032464 005737 001166          MOV      $LPERR,(SP)      ;;FUDGE RETURN FOR LOOPING
032470 001402          TST      $ESCAPE      ;;CHECK FOR AN ESCAPE ADDRESS
032472 013716 001166          BEQ      5$      ;;BR IF NONE
032476          MOV      $ESCAPE,(SP)      ;;FUDGE RETURN ADDRESS FOR ESCAPE
032476 022737 031732 000042      5$:      CMP      # $ENDAD,@#42      ;;ACT 11 AUTO ACCEPT?
032504 001001          BNE      6$      ;;BRANCH IF NO
032506 000000          HALT          ;;YES
032510          6$:
032510 000002          RTI          ;;RETURN
;SBTTL ABORT ROUTINE FOR LCP/ORION UFD MODE
032512 005737 004152      ABORT:      TST      UFDPLG      ;;TEST FOR USER FRIENDLY MODE
032516 001454          BEQ      NOABRT      ;;IF NOT UFD THEN CONTINUE NORMAL OPERATION
032520 020027 000032          CMP      R0,#32      ;;IS IT A +Z ?
  
```

ABORT ROUTINE FOR LCP/ORION UFD MODE

```

032524 001443      BEQ      ABORTZ      ;JUST GO BACK TO CHAIN IF IT IS (NO ERROR)
032526 020027 000003  CMP      RO,#3      ;IS IS A +C ?
032532 001404      BFQ      ABORTC      ;BR TO LOAD +C ON XXDP+ STACK (NO ERROR)
032534 005737 004154  TST      UQUIET      ;TEST FOR USER-QUIET MODE
032540 001443      BEQ      NOABRT      ;IF FIELD-SERVICE MODE, CONTINUE NORMAL OPERATION
                                ; BECAUSE FIELD-SERVICE MODE DOES NOT QUIT ON ERROR
                                ;SET DRSERR THEN LEAVE
032542 000422      BR       ABORTE      ;RESTORE EMT LOCATION (30)
032544 013737 004146 000030 ABORTC: MOV      SAV30,30 ;RESTORE EMT PRIORITY LOCATION (32)
032552 013737 004150 000032  MOV      SAV32,32
032560 104043      EMT      +43        ;GET XXDP STACK LOC. INTO RO FROM MONITOR
032562 005720      1$:    TST      (RO)+  ;FIND END OF STACK
032564 001376      BNE      1$
032566 112760 000057 177777  MOVB     #'/, 1(RO)   ;LOAD SLASH OVER ZERO
032574 112720 000136  MOVB     #'+(RO)+    ;LOAD JPARROW
032600 112720 000103  MOVB     #'C,(RO)+   ;LOAD C
032604 105010      CLRB     (RO)        ;MAKE NEW END TO STACK
032606 000412      BR       ABORTZ      ;NOW LEAVE
032610 013737 004146 000030 ABORTE: MOV      SAV30,30 ;RESTORE EMT LOCATION (30)
032616 013737 004150 000032  MOV      SAV32,32   ;RESTORE EMT PRIORITY LOCATION (32)
032624 104042      EMT      +42        ;GET DCA LOCATION INTO RO FROM MONITOR
032626 012760 177777 000042  MOV      #-1,42(RO) ;SET A -1 INTO LOCATION DRSERR IN MONITOR
032634 013700 000042  ABORTZ: MOV      @#42,RO ;AND PUT THE MONITOR RETURN ADDRESS IN RO
032640 005037 000042  CLR      @#42        ;CLEAR MONITOR RETURN FLAG
032644 000137 031732  JMP      $ENDAD      ;RETURN TO MONITOR DO NOT PUSH STACK HERE
032650 000207      NOABRT: RTS      PC   ;IF NOTUFD RETURN TO MAINLINE

4954 .SBTTL APT COMMUNICATIONS ROUTINE
;*****
032652 112737 000001 033116 $ATY1: MOVB     #1,$FFLG ;:TO REPORT FATAL ERROR
032660 112737 000001 033114 $ATY3: MOVB     #1,$MFLG ;:TO TYPE A MESSAGE
032666 000403      BR       $ATYC
032670 112737 000001 033116 $ATY4: MOVB     #1,$FFLG ;:TO ONLY REPORT FATAL ERROR
032676      $ATYC:
032676 010046      MOV      RO,-(SP)    ;:PUSH RO ON STACK
032700 010146      MOV      R1,-(SP)    ;:PUSH R1 ON STACK
032702 105737 033114  TSTB     $MFLG      ;:SHOULD TYPE A MESSAGE?
032706 001450      BEQ      5$          ;:IF NOT: BR
032710 122737 000001 001220  CMPB     #APTENV,$ENV ;:OPERATING UNDER APT?
032716 001031      BNE      3$          ;:IF NOT: BR
032720 132737 000100 001221  BITB     #APTPOOL,$ENVM ;:SHOULD SPOOL MESSAGES?
032726 001425      BEQ      3$          ;:IF NOT: BR
032730 017600 000004  MOV      @4(SP),RO   ;:GET MESSAGE ADDR.
032734 062766 000002 000004  ADD      #2,4(SP)    ;:BUMP RETURN ADDR.
032742 005737 001200  1$:    TST      $MSGTYPE ;:SEE IF DONE W/ LAST XMISSION?
032746 001375      BNE      1$          ;:IF NOT: WAIT
032750 010037 001214  MOV      RO,$MSGAD   ;:PUT ADDR IN MAILBOX
032754 105720      2$:    TSTB     (RO)+  ;:FIND END OF MESSAGE
032756 001376      BNE      2$
032760 163700 001214  SUB      $MSGAD,RO   ;:SUB START OF MESSAGE
032764 006200      ASR      RO          ;:GET MESSAGE LNTH IN WORDS
032766 010037 001216  MOV      RO,$MSGGLT  ;:PUT LENGTH IN MAILBOX
032772 012737 000004 001200  MOV      #4,$MSGTYPE ;:TELL APT TO TAKE MSG.
033000 000413      BR       5$
033002 017637 000004 033026 3$:    MOV      @4(SP),4$   ;:PUT MSG ADDR IN JSR LINKAGE
033010 062766 000002 000004  ADD      #2,4(SP)    ;:BUMP RETURN ADDRESS
033016 013746 177776  MOV      177776,(SP) ;:PUSH 177776 ON STACK
033022 004737 033120  JSR      PC,$TYPE   ;:CALL TYPE MACRO
033026 000000      4$:    .WORD     0

```

APT COMMUNICATIONS ROUTINE

```

033030      5$:
033030 105737 033116 10$: TSTB $FFLG      ;; SHOULD REPORT FATAL ERROR?
033034 001416      BEQ 12$      ;; IF NOT: BR
033036 005737 001220 TST $ENV      ;; RUNNING UNDER APT?
033042 001413      BEQ 12$      ;; IF NOT: BR
033044 005737 001200 11$: TST $MSGTYPE  ;; FINISHED LAST MESSAGE?
033050 001375      BNE 11$      ;; IF NOT: WAIT
033052 017637 000004 001202 MOV @4(SP), $FATAL ;; GET ERROR #
033060 062766 000002 000004 ADD @2,4(SP)      ;; BUMP RETURN ADDR.
033066 005237 001200 INC $MSGTYPE      ;; TELL APT TO TAKE ERROR
033072 105037 033116 12$: CLRB $FFLG      ;; CLEAR FATAL FLAG
033076 105037 033115 CLRB $LFLG      ;; CLEAR LOG FLAG
033102 105037 033114 CLRB $MFLG      ;; CLEAR MESSAGE FLAG
033106 012601      MOV (SP)+,R1      ;; POP STACK INTO R1
033110 012600      MOV (SP)+,R0      ;; POP STACK INTO R0
033112 000207      RTS PC      ;; RETURN
033114 000      $MFLG: .BYTE 0      ;; MESSG. FLAG
033115 000      $LFLG: .BYTE 0      ;; LOG FLAG
033116 000      $FFLG: .BYTE 0      ;; FATAL FLAG
          .EVEN

```

```

000200 APTSIZE=200
000001 APTENV=001
000100 APTSPool=100
000040 APTCSUP=040

```

4955

.SBTTL TYPE ROUTINE

```

*****
*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: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
*NOTE3: $FILLC 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

```

```

033120 105737 001157 $TYPE: TSTB $TPFLG      ;; IS THERE A TERMINAL?
033124 100002      BPL 1$      ;; BR IF YES
033126 000000      HALT      ;; HALT HERE IF NO TERMINAL
033130 000430      BR 3$      ;; LEAVE
033132 010046      MOV RO,-(SP)  ;; SAVE RO
033134 017600 000002 1$: MOV @2(SP),RO  ;; GET ADDRESS OF ASCIZ STRING
033140 122737 000001 001220 CMPB #APTENV,$ENV  ;; RUNNING IN APT MODE
033146 001011      BNE 62$      ;; NO,GO CHECK FOR APT CONSOLE
033150 132737 000100 001221 BITB #APTSPool,$ENVM ;; SPOOL MESSAGE TO APT
033156 001405      BEQ 62$      ;; NO,GO CHECK FOR CONSOLE
033160 010037 033170      MOV RO,61$  ;; SETUP MESSAGE ADDRESS FOR APT
033164 004737 032660      JSR PC,$ATY3 ;; SPOOL MESSAGE TO APT
033170 000000      .WORD 0      ;; MESSAGE ADDRESS
033172 132737 000040 001221 61$: BITB #APTCSUP,$ENVM ;; APT CONSOLE SUPPRESSED
033200 001003      BNE 60$      ;; YES,SKIP TYPE OUT
033202 112046      MOV (RO)+,-(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
033204 001005      BNE 4$      ;; BR IF IT ISN'T THE TERMINATOR
033206 005726      TST (SP)+  ;; IF TERMINATOR POP IT OFF THE STACK

```


PE ROUTINE

```

033210 012600          60$: MOV    (SP),R0      ;;RESTORE R0
033212 062716 000002  3$:  ADD    #2,(SP)    ;;ADJUST RETURN PC
033216 000002          RTI                    ;;RETURN
033220 122716 000011  4$:  CMPB   #HT,(SP)    ;;BRANCH IF <HT>
033224 001430          BEQ    8$                    ;;
033226 122716 000200  CMPB   #CRLF,(SP)    ;;BRANCH IF NOT <CRLF>
033232 001006          BNE    5$                    ;;
033234 005726          TST    (SP)+        ;;POP <CR><LF> EQUIV
033236 104401          TYPE   $CRLF          ;;TYPE A CR AND LF
033240 001175          $CRLF
033242 105037 033450  CLRB   $CHARCNT     ;;CLEAR CHARACTER COUNT
033246 006755          BR     2$                    ;;GET NEXT CHARACTER
033250 004737 033332  5$:  JSR    PC,$TYPEC    ;;GO TYPE THIS CHARACTER
033254 123726 001156  6$:  CMPB   #FILLC,(SP)+  ;;IS IT TIME FOR FILLER CHARS.?
033260 001350          BNE    2$                    ;;IF NO GO GET NEXT CHAR.
033262 013746 001154  MOV    $NULL,(SP)   ;;GET # OF FILLER CHARS. NEEDED
                                ;;AND THE NULL CHAR.
033266 105366 000001  7$:  DECB   1(SP)        ;;DOES A NULL NEED TO BE TYPED?
033272 002770          BLT    6$                    ;;BR IF NO- GO POP THE NULL OFF OF STACK
033274 004737 033332  JSR    PC,$TYPEC    ;;GO TYPE A NULL
033300 105337 033450  DECB   $CHARCNT     ;;DO NOT COUNT AS A COUNT
033304 000770          BR     7$                    ;;LOOP
                                ;HORIZONTAL TAB PROCESSOR
033306 112716 000040  8$:  MOVB   #' ,(SP)    ;;REPLACE TAB WITH SPACE
033312 004737 033332  9$:  JSR    PC,$TYPEC    ;;TYPE A SPACE
033316 132737 000007 033450  BITB   #7,$CHARCNT   ;;BRANCH IF NOT AT
033324 001372          BNE    9$                    ;;TAB STOP
033326 005726          TST    (SP)+        ;;POP SPACE OFF STACK
033330 000724          BR     2$                    ;;GET NEXT CHARACTER
                                $TYPEC:
033332 105777 145606  TSTB   @TKS          ;;CHAR IN KYBD BUFFER?
033336 100022          BPL    10$          ;;BR IF NOT
033340 017746 145602  MOV    @TKB,(SP)    ;;GET CHAR
033344 042716 177600  BIC    #177600,(SP) ;;STRIP EXTRANEIOUS BITS
033350 122716 000023  CMPB   #XOFF,(SP)  ;;WAS CHAR XOFF
033354 001012          BNE    102$         ;;BR IF NOT
033356 105777 145562 101$: TSTB   @TKS          ;;WAIT FOR CHAR
033362 100375          BPL    101$         ;;
033364 117716 145556  MCVB   @TKB,(SP)    ;;GET CHAR
033370 042716 177600  BIC    #177600,(SP) ;;STRIP IT
033374 122716 000021  CMPB   #XON,(SP)   ;;WAS IT XON?
033400 001366          BNE    101$         ;;BR IF NOT
033402 105777 145540 102$: TST    (SP)+        ;;FIX STACK
033404 005726          TST    (SP)+        ;;
033410 100375          BPL    10$          ;;WAIT UNTIL PRINTER IS READY
033412 116677 000002 145532 10$:  MOVB   2(SP),@TPB    ;;LOAD CHAR TO BE TYPED INTO DATA REG.
033420 122766 000015 000002  CMPB   #CR,2(SP)    ;;IS CHARACTER A CARRIAGE RETURN?
033426 001003          BNE    1$                    ;;BRANCH IF NO
033430 105037 033450  CLRB   $CHARCNT     ;;YES -CLEAR CHARACTER COUNT
033434 000406          BR     $TYPEX        ;;EXIT
033436 122766 000012 000002 1$:  CMPB   #LF,2(SP)    ;;IS CHARACTER A LINE FEED?
033444 001402          BEQ    $TYPEX        ;;BRANCH IF YES
033446 105227          INCB   (PC)+        ;;COUNT THE CHARACTER
033450 000000          $CHARCNT: .WORD 0 ;;CHARACTER COUNT STORAGE

```

TYPE ROUTINE

033452 000207
4956

033454 017646 000000
033467 116637 000001 033677
033466 112637 033701
033472 062716 000002
033476 000406
033500 112737 000001 033677
033506 112737 000006 033701
033514 112737 000005 033676
033522 010346
033524 010446
033526 010546
033530 113704 033701
033534 005404
033536 062704 000006
033542 110437 033700
033546 113704 033677
033552 016605 000012
033556 005003
033560 006105
033562 000404
033564 006105
033566 006105
033570 006105
033572 010503
033574 006103
033576 105337 033700
033602 100016
033604 042703 177770
033610 001002
033612 005704
033614 001403
033616 005204
033620 052703 000060

```

$TYPEX: RTS      PC
.SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
;*****
;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6 DIGIT
;*OCTAL (ASCII) NUMBER AND TYPE IT.
;*$TYPOS -ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE
;*$CALL:
;*      MOV      NUM, (SP)      ;;NUMBER TO BE TYPED
;*      TYPOS    ;;CALL FOR TYPEOUT
;*      .BYTE   N              ;;N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
;*      .BYTE   M              ;;M=1 OR 0
;*                               ;;1=TYPE LEADING ZEROS
;*                               ;;0-SUPPRESS LEADING ZEROS
;*$TYPON -- ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST
;*$TYPOS OR $TYPOC
;*$CALL:
;*      MOV      NUM, (SP)      ;;NUMBER TO BE TYPED
;*      TYPON    ;;CALL FOR TYPEOUT
;*$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER
;*$CALL:
;*      MOV      NUM, -(SP)     ;;NUMBER TO BE TYPED
;*      TYPOC    ;;CALL FOR TYPEOUT
$TYPOS: MOV      @ (SP), -(SP)  ;;PICKUP THE MODE
        MOVB     1(SP), $OFILL  ;;LOAD ZERO FILL SWITCH
        MOVB     (SP)+, $OMODE+1 ;;NUMBER OF DIGITS TO TYPE
        ADD      #2, (SP)      ;;ADJUST RETURN ADDRESS
        BR      $TYPON
$TYPOC: MOVB     #1, $OFILL     ;;SET THE ZERO FILL SWITCH
        MOVB     #6, $OMODE+1  ;;SET FOR SIX(6) DIGITS
$TYPON: MOVB     #5, $OCNT     ;;SET THE ITERATION COUNT
        MOV      R3, -(SP)     ;;SAVE R3
        MOV      R4, -(SP)     ;;SAVE R4
        MOV      R5, -(SP)     ;;SAVE R5
        MOVB     $OMODE+1, R4  ;;GET THE NUMBER OF DIGITS TO TYPE
        NEG      R4
        ADD      #6, R4        ;;SUBTRACT IT FOR MAX. ALLOWED
        MOVB     R4, $OMODE    ;;SAVE IT FOR USE
        MOVB     $OFILL, R4    ;;GET THE ZERO FILL SWITCH
        MOV      12(SP), R5   ;;PICKUP THE INPUT NUMBER
        CLR      R3           ;;CLEAR THE OUTPUT WORD
1$:     ROL      R5           ;;ROTATE MSB INTO 'C'
        BR      3$           ;;GO DO MSB
2$:     ROL      R5           ;;FORM THIS DIGIT
        ROL      R5
        ROL      R5
        MOV      R5, R3
3$:     ROL      R3           ;;GET LSB OF THIS DIGIT
        C.LCB   $OMODE       ;;TYPE THIS DIGIT?
        BPL     7$           ;;BR IF NO
        BIC     #177770, R3  ;;GET RID OF JUNK
        BNE     4$           ;;TEST FOR 0
        TST     R4           ;;SUPPRESS THIS 0?
        BEQ     5$           ;;BR IF YES
4$:     INC      R4           ;;DON'T SUPPRESS ANYMORE 0'S
        BIS     #'0, R3      ;;MAKE THIS DIGIT ASCII
    
```

BINARY TO OCTAL (ASCII) AND TYPE

```

033624 052703 000040      5$:  BIS      #' ,R3      ;;MAKE ASCII IF NOT ALREADY
033630 110337 033674      MOVVB   R3,8$      ;;SAVE FOR TYPING
033634 104401 033674      TYPE   .8$       ;;GO TYPE THIS DIGIT
033640 105337 033676      7$:  DECB   $OCNT   ;;COUNT BY 1
033644 003347      BGT    2$        ;;BR IF MORE TO DO
033646 002402      BLT    6$        ;;BR IF DONE
033650 005204      INC    R4        ;;INSURE LAST DIGIT ISN T A BLANK
033652 000744      BR     2$        ;;GO DO THE LAST DIGIT
033654 012605      6$:  MOV    (SP)+,R5  ;;RESTORE R5
033656 012604      MOV    (SP)+,R4  ;;RESTORE R4
033660 012603      MOV    (SP)+,R3  ;;RESTORE R3
033662 016666 000002 000004  MOV    2(SP),4(SP) ;;SET THE STACK FOR RETURNING
033670 012616      MOV    (SP)+,(SP)
033672 000002      RTI                    ;;RETURN
033674      000      8$:  .BYTE  0      ;;STORAGE FOR ASCII DIGIT
033675      000      .BYTE  0      ;;TERMINATOR FOR TYPE ROUTINE
033676      000      $OCNT: .BYTE  0      ;;OCTAL DIGIT COUNTER
033677      000      $OFILL: .BYTE  0      ;;ZERO FILL SWITCH
033700 000000      $OMODE: .WORD  0      ;;NUMBER OF DIGITS TO TYPE
4957      .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
;*****
;*THIS ROUTINE IS USED TO CHANGE A 16 BIT BINARY NUMBER TO A 5 DIGIT
;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
;*REPLACED WITH SPACES.
;*CALL:
;*      MOV    NUM,-(SP)      ;;PUT THE BINARY NUMBER ON THE STACK
;*      TYPDS      ;;GO TO THE ROUTINE
$TYPDS:
MOV    R0,-(SP)      ;;PUSH R0 ON STACK
MOV    R1,-(SP)      ;;PUSH R1 ON STACK
MOV    R2,-(SP)      ;;PUSH R2 ON STACK
MOV    R3,-(SP)      ;;PUSH R3 ON STACK
MOV    R5,-(SP)      ;;PUSH R5 ON STACK
MOV    #20200,-(SP)  ;;SET BLANK SWITCH AND SIGN
MOV    20(SP),R5     ;;GET THE INPUT NUMBER
BPL    1$           ;;BR IF INPUT IS POS.
NEG    R5           ;;MAKE THE BINARY NUMBER POS.
MOVVB  #'-,1(SP)    ;;MAKE THE ASCII NUMBER NEG.
1$:  CLR    R0           ;;ZERO THE CONSTANTS INDEX
MOV    #DBLK,R3     ;;SETUP THE OUTPUT POINTER
MOVVB  #' ,(R3)+    ;;SET THE FIRST CHARACTER TO A BLANK
2$:  CLR    R2           ;;CLEAR THE BCD NUMBER
MOV    $DTBL(R0),R1 ;;GET THE CONSTANT
3$:  SUB    R1,R5       ;;FORM THIS BCD DIGIT
BLT    4$           ;;BR IF DONE
INC    R2           ;;INCREASE THE BCD DIGIT BY 1
BR     3$
4$:  ADD    R1,R5       ;;ADD BACK THE CONSTANT
TST    R2           ;;CHECK IF BCD DIGIT=0
BNE    5$           ;;FALL THROUGH IF 0
TSTB  (SP)         ;;STILL DOING LEADING 0 S?
BMI    7$           ;;BR IF YES
5$:  ASLB  (SP)         ;;MSD?
BCC    6$           ;;BR IF NO
MOVVB  1(SP),-1(R3) ;;YES--SET THE SIGN

```

ELP

CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

034012 052702 000060      6$:   BIS      #'0,R2      ;;MAKE THE BCD DIGIT ASCII
034016 052702 000040      7$:   BIS      #' ,R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
034022 110223              MOVB     R2,(R3)+     ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
034024 005720              TST      (R0)+         ;;JUST INCREMENTING
034026 020027 000010      CMP      R0,#10      ;;CHECK THE TABLE INDEX
034032 002746              BLT      2$          ;;GO DO THE NEXT DIGIT
034034 003002              BGT      8$          ;;GO TO EXIT
034036 010502              MOV      R5,R2      ;;GET THE LSD
034040 000764              BR       6$          ;;GO CHANGE TO ASCII
034042 105726              8$:   TSTB     (SP)+     ;;WAS THE LSD THE FIRST NON ZERO?
034044 100003              9$:   BPL      9$          ;;BR IF NO
034046 116663 177777 177776 MOVB     1(SP), 2(R3) ;;YES SET THE SIGN FOR TYPING
034054 105013              9$:   CLRB     (R3)      ;;SET THE TERMINATOR
034056 012605              MOV      (SP)+,R5    ;;POP STACK INTO R5
034060 012603              MOV      (SP)+,R3    ;;POP STACK INTO R3
034062 012602              MOV      (SP)+,R2    ;;POP STACK INTO R2
034064 012601              MOV      (SP)+,R1    ;;POP STACK INTO R1
034066 012600              MOV      (SP)+,R0    ;;POP STACK INTO R0
034070 104401 034116      TYPE     ,#DBLK      ;;NOW TYPE THE NUMBER
034074 016666 000002 000004 MOV      2(SP),4(SP)  ;;ADJUST THE STACK
034102 012616              MOV      (SP)+,(SP)
034104 000002              RTI                    ;;RETURN TO USER
034106 023420      $DTBL: 10000.
034110 001750              1000.
034112 000144              100.
034114 000012              10.
034116      $DBLK: .BLKW 4

```

4958

```

.SBTTL  TTY INPUT ROUTINE
;*****
.ENABL  LSB
;*****
;SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
;ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
;SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
;WHEN OPERATING IN TTY FLAG MODE.
034126 022737 000176 001140 $CKSWR: CMP      #SWREG,SWR      ;;IS THE SOFT SWR SELECTED?
034134 001074              BNE      15$          ;;BRANCH IF NO
034136 105777 145002              TSTB     @TKS         ;;CHAR THERE?
034142 100071              BPL      15$          ;;IF NO, DON'T WAIT AROUND
034144 117746 144776              MOVB     @TKB,-(SP)   ;;SAVE THE CHAR
034150 042716 177600              BIC      #C177,(SP)  ;;STRIP-OFF THE ASCII
034154 022726 000007              CMP      #7,(SP)+    ;;IS IT A CONTROL G?
034160 001062              BNE      15$          ;;NO, RETURN TO USER
034162 123727 001134 000001 CMPB     $AUTOB,#1    ;;ARE WE RUNNING IN AUTO MODE?
034170 001456              BEQ      15$          ;;BRANCH IF YES
034172 104401 034663              TYPE     ,#CNTLG     ;;ECHO THE CONTROL G (+G)
034176 104401 034670      $GTSWR: TYPE     ,#MSWR     ;;TYPE CURRENT CONTENTS
034202 013746 000176              MOV      SWREG,(SP)  ;;SAVE SWREG FOR TYPEOUT
034206 104402              TYPOC    ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
034210 104401 034701              TYPE     ,#MNEW      ;;PROMPT FOR NEW SWR
034214 005046              19$:   CLR      -(SP)    ;;CLEAR COUNTER
034216 005046              CLR      -(SP)      ;;THE NEW SWR
034220 105777 144720              7$:   TSTB     @TKS         ;;CHAR THERE?
034224 100375              BPL      7$          ;;IF NOT TRY AGAIN
034226 117746 144714              MOVB     @TKB,(SP)   ;;PICK UP CHAR
034232 042716 177600              BIC      #C177,(SP)  ;;MAKE IT 7-BIT ASCII
034236 021627 000025              9$:   CMP      (SP),#25  ;;IS IT A CONTROL U?

```

FL

TTY INPUT ROUTINE

```

034242 001005          BNE      10$          ;;BRANCH IF NOT
034244 104401 034656   TYPE     , $CNTLU    ;;YES, ECHO CONTROL U (^U)
034250 062706 000006   20$:    ADD     @6,SP    ;;IGNORE PREVIOUS INPUT
034254 000757          BR       19$          ;;LET'S TRY IT AGAIN
034256 021627 000015   10$:    CMP     (SP),#15    ;;IS IT A <CR>?
034262 001022          BNE     16$          ;;BRANCH IF NO
034264 005766 000004   TST     4(SP)        ;;YES, IS IT THE FIRST CHAR?
034270 001403          BEQ     11$          ;;BRANCH IF YES
034272 016677 000002 144640   MOV     2(SP),@SWR    ;;SAVE NEW SWR
034300 062706 000006   11$:    ADD     @6,SP    ;;CLEAR UP STACK
034304 104401 001175   14$:    TYPE     , $CRLF    ;;ECHO <CR> AND <LF>
034310 123727 001135 000001   CMPB   $INTAG,#1    ;;RE ENABLE TTY KBD INTERRUPTS?
034316 001003          BNE     15$          ;;BRANCH IF NOT
034320 012777 000100 144616   MOV     @100,@$TKS   ;;RE ENABLE TTY KBD INTERRUPTS
034326 000002          RTI                    ;;RETURN
034330 004737 033332   15$:    JSR     PC,$TYPEC   ;;ECHO CHAR
034334 021627 000060   16$:    CMP     (SP),#60   ;;CHAR < 0?
034340 002420          BLT     18$          ;;BRANCH IF YES
034342 021627 000067   CMP     (SP),#67    ;;CHAR > 7?
034346 003015          BGT     18$          ;;BRANCH IF YES
034350 042726 000060   BIC     @60,(SP)+   ;;STRIP-OFF ASCII
034354 005766 000002   TST     2(SP)        ;;IS THIS THE FIRST CHAR
034360 001403          BEQ     17$          ;;BRANCH IF YES
034362 006316          ASL     (SP)         ;;NO, SHIFT PRESENT
034364 006316          ASL     (SP)         ;; CHAR OVER TO MAKE
034366 006316          ASL     (SP)         ;; ROOM FOR NEW ONE.
034370 005266 000002   17$:    INC     2(SP)        ;;KEEP COUNT OF CHAR
034374 056616 177776   BIS     2(SP),(SP)   ;;SET IN NEW CHAR
034400 000707          BR       7$          ;;GET THE NEXT ONE
034402 104401 001174   18$:    TYPE     , $QUES   ;;TYPE ?<CR><LF>
034406 000720          BR       20$          ;;SIMULATE CONTROL U
.DSABL  LSB
;*****
;THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
;*CALL:
;* RDCHR          ;;INPUT A SINGLE CHARACTER FROM THE TTY
;* RETURN HERE   ;;CHARACTER IS ON THE STACK
;*              ;;WITH PARITY BIT STRIPPED OFF
;
$RDCHR: MOV     (SP),-(SP)  ;;PUSH DOWN THE PC
034412 016666 000004 000002   MOV     4(SP),2(SP)  ;;SAVE THE PS
034420 105777 144520   1$:    TSTB   @TKS        ;;WAIT FOR
034424 100375          BPL     1$          ;;A CHARACTER
034426 117766 144514 000004   MOVB   @TKB,4(SP)    ;;READ THE TTY
034434 042766 177600 000004   BIC     @C<177>,4(SP) ;;GET RID OF JUNK IF ANY
034442 026627 000004 000023   CMP     4(SP),#23    ;;IS IT A CONTROL-S?
034450 001013          BNE     3$          ;;BRANCH IF NO
034452 105777 144466   2$:    TSTB   @TKS        ;;WAIT FOR A CHARACTER
034456 100375          BPL     2$          ;;LOOP UNTIL ITS THERE
034460 117746 144462   MOVB   @TKB,-(SP)    ;;GET CHARACTER
034464 042716 177600   BIC     @C177,(SP)   ;;MAKE IT 7-BIT ASCII
034470 022627 000021   CMP     (SP)+,#21    ;;IS IT A CONTROL Q?
034474 001366          BNE     2$          ;;IF NOT DISCARD IT
034476 000750          BR       1$          ;;YES, RESUME
034500 026627 000004 000021  3$:    CMP     4(SP),#$XON  ;;IS IT A RANDOM XON?
034506 001744          BEQ     1$          ;;BRANCH IF YES
034510 026627 000004 000140   CMP     4(SP),#140   ;;IS IT UPPER CASE?
;RAN001
;RAN001

```

TTY INPUT ROUTINE

```

034516 002407          BLT      4$          ;;BRANCH IF YES
034520 026627 000004 000175    CMP      4(SP),#175      ;;IS IT A SPECIAL CHAR?
034526 003003          BGT      4$          ;;BRANCH IF YES
034530 042766 000040 000004    BIC      #40,4(SP)      ;;MAKE IT UPPER CASE
034536 000002          RTI          ;;GO BACK TO USER
4$:
;*****
;THIS ROUTINE WILL INPUT A STRING FROM THE TTY
;CALL:
;*
;*   RDLIN          ;;INPUT A STRING FROM THE TTY
;*   RETURN HERE   ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
;*                ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
$RDLIN: MOV      R3, (SP)      ;;SAVE R3
1$:    MOV      #177,R3      ;;GET ADDRESS
2$:    CMP      #177,R3      ;;BUFFER FULL?
      BLOS     4$          ;;BR IF YES
      RDCHR    (SP)+,(R3)    ;;GO READ ONE CHARACTER FROM THE TTY
10$:   CMPB    #177,(R3)    ;;GET CHARACTER
      BNE     3$          ;;IS IT A RUBOUT
      TYPE    , $QUES      ;;SKIP IF NOT
      BR      1$          ;;TYPE A '?'
3$:    MOVB    (R3),9$      ;;CLEAR THE BUFFER AND LOOP
      TYPE    ,9$          ;;ECHO THE CHARACTER
      CMPB    #15,(R3)+    ;;CHECK FOR RETURN
      BNE     2$          ;;LOOP IF NOT RETURN
      CLRB    1(R3)        ;;CLEAR RETURN (THE 15)
      TYPE    , $LF        ;;TYPE A LINE FEED
      MOV     (SP)+,R3      ;;RESTORE R3
      MOV     (SP)-,(SP)    ;;ADJUST THE STACK AND PUT ADDRESS OF THE
      MOV     4(SP),2(SP)  ;; FIRST ASCII CHARACTER ON IT
      MOV     #177,4(SP)
      RTI
9$:    .BYTE   0          ;;RETURN
      .BYTE   0          ;;STORAGE FOR ASCII CHAR. TO TYPE
      .BLKB   8          ;;TERMINATOR
      .ASCIZ  /+U/<15><12> ;;RESERVE 8 BYTES FOR TTY INPUT
      .ASCIZ  /+G/<15><12> ;;CONTROL "U"
$TTYIN: .BLKB   8
$CNTLU: .ASCIZ  /+G/<15><12> ;;CONTROL "G"
$CNTLG: .ASCIZ  /+G/<15><12>
$MSWR:  .ASCIZ  <15><12>/SWR = /
$MNEW:  .ASCIZ  / NEW = /
.SBTTL  READ AN OCTAL NUMBER FROM THE TTY
;*****
;THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
;CHANGE IT TO BINARY.
;CALL:
;*
;*   RDOCT          ;;READ AN OCTAL NUMBER
;*   RETURN HERE   ;;LOW ORDER BITS ARE ON TOP OF THE STACK
;*                ;;HIGH ORDER BITS ARE IN $HIOCT
$RDOCT: MOV     (SP), (SP)  ;;PROVIDE SPACE FOR THE
      MOV     4(SP),2(SP)  ;;INPUT NUMBER
      MOV     R0,-(SP)     ;;PUSH R0 ON STACK
      MOV     R1,-(SP)     ;;PUSH R1 ON STACK

```

4959

READ AN OCTAL NUMBER FROM THE TTY

```

034726 010246      MOV      R2, (SP)      ;; PUSH R2 ON STACK
034730 104411      1$: RDLIN      ;; READ AN ASCII LINE
034732 012600      MOV      (SP)+,R0     ;; GET ADDRESS OF 1ST CHARACTER
034734 005001      CLR      R1          ;; CLEAR DATA WORD
034736 005002      CLR      R2
034740 112046      2$: MOVB     (R0)+,-(SP) ;; PICKUP THIS CHARACTER
034742 001412      BEQ      3$         ;; IF ZERO GET OUT
034744 006301      ASL      R1          ;; *2
034746 006102      ROL      R2
034750 006301      ASL      R1          ;; *4
034752 006102      ROL      R2
034754 006301      ASL      R1          ;; *8
034756 006102      ROL      R2
034760 042716      BIC      #C7,(SP)    ;; STRIP THE ASCII JUNK
034764 062601      ADD      (SP)+,R1    ;; ADD IN THIS DIGIT
034766 000764      BR       2$         ;; LOOP
034770 005726      3$: TST      (SP)+    ;; CLEAN TERMINATOR FROM STACK
034772 010166      MOV      R1,12(SP)   ;; SAVE THE RESULT
034776 010237      MOV      R2,$HIOCT
035002 012602      MOV      (SP)+,R2    ;; POP STACK INTO R2
035004 012601      MOV      (SP)+,R1    ;; POP STACK INTO R1
035006 012600      MOV      (SP)+,R0    ;; POP STACK INTO R0
035010 000002      RTI
035012 000000      $HIOCT: .WORD 0     ;; HIGH ORDER BITS GO HERE

```

4960

```

.SBTTL TRAP DECODER
;*****
;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
;*GO TO THAT ROUTINE.

```

```

035014 010046      $TRAP: MOV      RO,-(SP)      ;; SAVE RO
035016 016600      MOV      2(SP),RO      ;; GET TRAP ADDRESS
035022 005740      TST      (RO)         ;; BACKUP BY 2
035024 111000      MOVB     (RO),RO      ;; GET RIGHT BYTE OF TRAP
035026 006300      ASL      RO          ;; POSITION FOR INDEXING
035030 016000      MOV      $TRPAD(RO),RO ;; INDEX TO TABLE
035034 000200      RTS      RO          ;; GO TO ROUTINE

```

```

;; THIS IS USE TO HANDLE THE "GETPRI" MACRO
035036 011646      $TRAP2: MOV      (SP), (SP) ;; MOVE THE PC DOWN
035040 016666      MOV      4(SP),2(SP)   ;; MOVE THE PSW DOWN
035046 000002      RTI                  ;; RESTORE THE PSW

```

```

.SBTTL TRAP TABLE
;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
;*BY THE "TRAP" INSTRUCTION.
: ROUTINE
:
:

```

```

035050 035036      $TRPAD: .WORD  $TRAP2
035052 033120      $TYPE   ;; CALL=TYPE   TRAP+1(104401) TTY TYPEOUT ROUTINE
035054 033500      $TYPOC  ;; CALL=TYPOC  TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
035056 033454      $TYPOS  ;; CALL=TYPOS  TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
035060 033514      $TYPON  ;; CALL=TYPON  TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
035062 033702      $TYPDS  ;; CALL=TYPDS  TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
035064 034176      $GTSWR  ;; CALL=GTSWR  TRAP+6(104406) GET SOFT SWR SETTING
035066 034126      $CKSWR  ;; CALL=CKSWR  TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
035070 034410      $RDCHR  ;; CALL=RDCHR  TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
035072 034540      $RDLIN  ;; CALL=RDLIN  TRAP+11(104411) TTY TYPEIN STRING ROUTINE
035074 034712      $RDOCT  ;; CALL=RDOCT  TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY

```

POWER DOWN AND UP ROUTINES

4961

```

.SBTTL POWER DOWN AND UP ROUTINES
;*****
;POWER DOWN ROUTINE
035076 012737 035236 000024 $PWRDN: MOV    $$ILLUP,@#PWRVEC ;;SET FOR FAST UP
035104 012737 000340 000026      MOV    #340,@#PWRVEC+2 ;;PRIO:7
035112 010046      MOV    R0,-(SP)      ;;PUSH R0 ON STACK
035114 010146      MOV    R1,(SP)      ;;PUSH R1 ON STACK
035116 010246      MOV    R2,-(SP)      ;;PUSH R2 ON STACK
035120 010346      MOV    R3,(SP)      ;;PUSH R3 ON STACK
035122 010446      MOV    R4,-(SP)      ;;PUSH R4 ON STACK
035124 010546      MOV    R5,-(SP)      ;;PUSH R5 ON STACK
035126 017746      MOV    @SWR,-(SP)    ;;PUSH @SWR ON STACK
035132 010637 035242      MOV    SP,$SAVR6    ;;SAVE SP
035136 012737 035150 000024      MOV    $$PWRUP,@#PWRVEC ;;SET UP VECTOR
035144 000000      HALT
035146 000776      BR     -2          ;;HANG UP
;*****
;POWER UP ROUTINE
035150 012737 035236 000024 $PWRUP: MOV    $$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
035156 013706 035242      MOV    $SAVR6,SP    ;;GET SP
035162 005037 035242      CLR    $SAVR6      ;;WAIT LOOP FOR THE TTY
035166 005237 035242      1$:  INC    $SAVR6    ;;WAIT FOR THE INC
035172 001375      BNE    1$          ;;OF WORD
035174 012677 143740      MOV    (SP)+,@SWR   ;;POP STACK INTO @SWR
035200 012605      MOV    (SP)+,R5    ;;POP STACK INTO R5
035202 012604      MOV    (SP)+,R4    ;;POP STACK INTO R4
035204 012603      MOV    (SP)+,R3    ;;POP STACK INTO R3
035206 012602      MOV    (SP)+,R2    ;;POP STACK INTO R2
035210 012601      MOV    (SP)+,R1    ;;POP STACK INTO R1
035212 012600      MOV    (SP)+,R0    ;;POP STACK INTO R0
035214 012737 035076 000024      MOV    $$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
035222 012737 000340 000026      MOV    #340,@#PWRVEC+2 ;;PRIO:7
035230 104401      TYPE    $POWER     ;;REPORT THE POWER FAILURE
035232 035244      $PWRMG: .WORD    $POWER ;;POWER FAIL MESSAGE POINTER
035234 000002      RTI
035236 000000      $ILLUP: HALT      ;;THE POWER UP SEQUENCE WAS STARTED
035240 000776      BR     -2          ;; BEFORE THE POWER DOWN WAS COMPLETE
035242 000000      $SAVR6: 0          ;;PUT THE SP HERE
035244      015      012      120
035247      117      127      105
035252      122      000

.EVEN
.END
4963      000001

```


Symbol table

ABASE = 000000	APRIOR= 000000	CSR22 002366	EM100 021504	EM61 021013
ABORT 032512	APTCSU= 000040	CURADD 004054	EM101 021542	EM63 021042
ABORTC 032544	APTENV- 000001	CURDAT 004046	EM102 021575	EM64 021073
ABORTE 032610	APTSIZ= 000200	DATA 002344	EM103 021657	EM65 021132
ABORTZ 032634	APTSP0= 000100	DATA2 002374	EM104 021732	EM71 021202
ACDW1 = 000000	ASWREG= 000000	DATVER 030556	EM105 022002	EM72 021236
ACDW2 = 000000	AATESTN= 000000	DATVFR 030540	EM106 022055	EM73 021274
ACPUOP= 000000	AUNIT = 000000	DAT1 030570	EM107 022133	EM74 021320
ACTCHS 002410	AUSWR = 000000	DCOUNT 004030	EM110 022167	EM75 021351
ACO = 0000000	AVECT1= 000000	DDISP = 177570	EM111 022231	EM76 021402
AC1 = 0000001	AVECT2= 000000	DELAY 030610	EM112 022307	EM77 021452
AC2 = 0000002	BA 002340	DETFPA 031544	EM113 022372	ERR 030536
AC3 = 0000003	BA2 002370	DH1 025314	EM114 022456	ERRCNT 003024
AC4 = 0000004	BCR = 177524	DH105 026255	EM123 022477	ERRFP 030534
AC5 = 0000005	BCSR = 177520	DH115 026307	EM124 022531	ERROR = 104000
AC6 = 0000006	BDR = 177524	DH134 026364	EM125 022623	ERRVEC= 000004
AC7 = 0000007	BEGROM= 100000	DH24 025600	EM126 022662	ERTYPE 027030
ADDTRP 030376	BIT0 = 000001	DH27 025644	EM127 022717	EXPDAT 004032
ADDW0 = 000000	BIT00 = 000001	DH4 025341	EM130 022761	FLAG 003032
ADDW1 = 000000	BIT01 = 000002	DH41 025707	EM131 023006	FLO 003066
ADDW10= 000000	BIT02 = 000004	DH43 025760	EM132 023036	FLOAT 003056
ADDW11= 000000	BIT03 = 000010	DH47 026060	EM135 023074	FPVEC = 000244
ADDW12= 000000	BIT04 = 000020	DH5 025426	EM136 023132	FSTADD 004050
ADDW13= 000000	BIT05 = 000040	DH65 026145	EM137 023174	GOODAD 003022
ADDW14= 000000	BIT06 = 000100	DH7 025525	EM140 023241	GTSWR = 104406
ADDW15= 000000	BIT07 = 000200	DH72 026211	EM141 023332	HITMIS= 17752
ADDW2 = 000000	BIT08 = 000400	DISPLA 001142	EM142 023415	HT = 000011
ADDW3 = 000000	BIT09 = 001000	DISPRE 000174	EM143 023506	INIMEM 031012
ADDW4 = 000000	BIT1 = 000002	DMARD 030130	EM144 023561	INITMM 027246
ADDW5 = 000000	BIT10 = 002000	DMATRN 030046	EM145 023636	INQ22 017352
ADDW6 = 000000	BIT11 = 004000	DSWR = 177570	EM146 023704	IOTVEC= 000020
ADDW7 = 000000	BIT12 = 010000	DT1 026502	EM147 023747	J11FLT 031552
ADDW8 = 000000	BIT13 = 020000	DT105 026760	EM150 024024	KDPAR0= 172360
ADDW9 = 000000	BIT14 = 040000	DT115 026770	EM151 024105	KDPAR1= 172362
ADEVCT= 000000	BIT15 = 100000	DT130 027004	EM152 024153	KDPAR2= 172364
ADEVH = 000000	BIT2 = 000004	DT134 027012	EM153 024235	KDPAR3= 172366
ADLSB 004040	BIT3 = 000010	DT14 026550	EM154 024315	KDPAR4= 172370
AENV = 000000	BIT4 = 000020	DT17 026562	EM155 024402	KDPAR5= 172372
AENVM = 000000	BIT5 = 000040	DT24 026574	EM156 024455	KDPAR6= 172374
AFATAL= 000000	BIT6 = 000100	DT27 026604	EM157 024527	KDPAR7= 172376
ALLCTR 003156	BIT7 = 000200	DT35 026616	EM160 024575	KDPDR0= 172320
AMADR1= 000000	BIT8 = 000400	DT4 026510	EM161 024617	KDPDR1= 172322
AMADR2= 000000	BIT9 = 001000	DT41 026630	EM162 024657	KDPDR2= 172324
AMADR3= 000000	BPTVEC= 000014	DT43 026640	EM163 024722	KDPDR3= 172326
AMADR4= 000000	BTEXP 003104	DT47 026654	EM164 024766	KDPDR4= 172330
AMAMS1= 000000	BTRES 003114	DT5 026522	EM165 025045	KDPDR5= 172332
AMAMS2= 000000	CCHPAS 003034	DT50 026666	EM166 025114	KDPDR6= 172334
AMAMS3= 000000	CCR = 177746	DT51 026700	EM167 025162	KDPDR7= 172336
AMAMS4= 000000	CKSWR = 104407	DT52 026712	EM170 025214	KIPAR0= 172340
AMSGAD= 000000	CLKCNT 027530	DT64 026724	EM171 025242	KIPAR1= 172342
AMSGLG= 000000	COUNT 003124	DT65 026736	EM172 025273	KIPAR2= 172344
AMSGTY= 000000	CPEREG= 177766	DT7 026536	EM2 020653	KIPAR3= 172346
AMTYP1= 000000	CR = 000015	DT75 026746	EM3 020665	KIPAR4= 172350
AMTYP2= 000000	CRLF = 000200	EEPAS 003036	EM51 020677	KIPAR5= 172352
AMTYP3= 000000	CSR1 002334	EMTSAV 004060	EM54 020677	KIPAR6= 172354
AMTYP4= 000000	CSR12 002364	EMTVEC= 000030	EM56 020735	KIPAR7= 172356
APASS = 000000	CSR2 002336	EM1 020617	EM57 020761	KIPDR0= 172300

Symbol table

KIPDR1=	172302	PIRQT	017710	SDPAR6=	172274	SW10	= 002000	TAB6A	003330
KIPDR2=	172304	PIRQVE=	000240	SDPAR7=	172276	SW11	= 004000	TAB7	003340
KIPDR3=	172306	POLY	= 120001	SDPDR0=	172220	SW12	= 010000	TAB8	003350
KIPDR4=	172310	PRO	= 000000	SDPDR1=	172222	SW13	= 020000	TAB9	003360
KIPDR5=	172312	PR1	= 000040	SDPDR2=	172224	SW14	= 040000	TBITVE=	000014
KIPDR6=	172314	PR2	= 000100	SDPDR3=	172226	SW15	= 100000	TCHAR	003004
KIPDR7=	172316	PR3	= 000140	SDPDR4=	172230	SW2	= 000004	TCOUNT	002360
KMCR	= 177734	PR4	= 000200	SDPDR5=	172232	SW3	= 000010	TEMP	002740
LDPARS	027410	PR5	= 000240	SDPDR6=	172234	SW4	= 000020	TIMERR	006050
LDPDRS	027440	PR6	= 000300	SDPDR7=	172236	SW5	= 000040	TIMOUT	003000
LEDCNT	002354	PR7	= 000340	SEQ	003076	SW6	= 000100	TKVEC	= 000060
LF	= 000012	PS	= 177776	SETMMU	030676	SW7	= 000200	TOUT	030220
LKCNT	002404	PSW	= 177776	SIMGOA	002352	SW8	= 000400	TPVEC	= 000064
LKS	= 177546	PWDSEQ	004036	SIPAR0=	172240	SW9	= 001000	TRAPVE=	000034
LKSFL	002402	PWRVEC=	000024	SIPAR1=	172242	TAB1	003240	TRPFLG	030532
LKSINT	027522	Q22EN	003002	SIPAR2=	172244	TAB10	003370	TRTVEC=	000014
LOOP	005064	Q22INT	030026	SIPAR3=	172246	TAB11	003400	TSTADD	003026
LOOPIN	003160	Q22SIZ	027546	SIPAR4=	172250	TAB11A	003410	TSTLOC	003166
LOPBAK	002362	RAMPAR	027464	SIPAR5=	172252	TAB12	003420	TST1	005220
LOWADD	003020	RBUF	= 177562	SIPAR6=	172254	TAB13	003430	TST10	010532
LSTADD	004052	RBUF1	= 176502	SIPAR7=	172256	TAB13B	003440	TST11	011552
MAIREG=	177750	RCOUNT	002356	SIPDR0=	172200	TAB14	003450	TST12	011762
MASK	003164	RCSR	= 177560	SIPDR1=	172202	TAB15	003460	TST13	012332
MER	= 172100	RCSR1	= 176500	SIPDR2=	172204	TAB16	003470	TST14	012546
MERTAG	003054	RDCHR	= 104410	SIPDR3=	172206	TAB17	003500	TST15	012706
MMRO	= 177572	RDLIN	= 104411	SIPDR4=	172210	TAB18	003510	TST16	013104
MMR1	= 177574	RDOCT	= 104412	SIPDR5=	172212	TAB2	003250	TST17	013302
MMR2	= 177576	RECDAT	004034	SIPDR6=	172214	TAB21	003520	TST2	005502
MMR3	= 172516	RECDST	003146	SIPDR7=	172216	TAB22	003530	TST20	013700
MMU	030230	RECFEC	003126	SLEND	016714	TAB23	003540	TST21	014056
MMUERR	006016	RECST	003136	SLOC00	003010	TAB24	003550	TST22	015100
MMUTRP	030402	RESVEC=	000010	SLOC01	003012	TAB25	003560	TST23	015654
MMVEC	= 000250	RITEDA	004042	SPS	003100	TAB26	003570	TST24	016266
MSFR	= 177744	RRMEM	031350	SPSJ	003102	TAB27	003600	TST25	016720
NATREG=	177520	RWTMEM	031176	SRO	= 177572	TAB28	003610	TST26	017134
NEWADD	003030	R6	= 000006	SR1	= 177574	TAB29	003620	TST27	017434
NEWDAT	004044	R7	= 000007	SR2	= 177576	TAB29A	003630	TST3	006102
NOABRT	032650	SAVBR	002414	SR3	= 172516	TAB3	003260	TST30	017712
NOSLU	016716	SAVLOC	002406	STACK	= 001100	TAB30	003640	TST4	006342
NOTOK	030460	SAVMRO	003044	START	004060	TAB31	003650	TST5	006570
NULL	= 000000	SAVMR1	003046	STBOT	= 001000	TAB32	003660	TST6	007452
NXTST	011552	SAVMR2	003050	STKLMT=	177774	TAB33	003670	TST7	010200
OK	030436	SAVPCR	002412	SWR	001140	TAB34	003700	TYPDS	= 104405
OK1	030464	SAVPOS	003162	SWREG	000176	TAB4	003270	TYPE	= 104401
ONQ22	027772	SAVSUP	003040	SWTSEL	017750	TAB40	003710	TYPOC	= 104402
PAR	030352	SAVSWR	003052	SW0	= 000001	TAB41	003720	TYPON	= 104404
PARABT	004056	SAVTIM	003016	SW00	= 000001	TAB42	003730	TYPOS	= 104403
PAREND	007452	SAVUSE	003042	SW01	= 000002	TAB43	003740	UDPAR0=	177660
PARINT	007444	SAV30	004146	SW02	= 000004	TAB45	003750	UDPAR1=	177662
PARPAT	003014	SAV32	004150	SW03	= 000010	TAB46	003760	UDPAR2=	177664
PAR1	030354	SCOPE	= 000004	SW04	= 000020	TAB47	003770	UDPAR3=	177666
PATT	006324	SDPAR0=	172260	SW05	= 000040	TAB47A	004000	UDPAR4=	177670
PCR	= 177522	SDPAR1=	172262	SW06	= 000100	TAB48	004010	UDPAR5=	177672
PDR	030330	SDPAR2=	172264	SW07	= 000200	TAB49	004020	UDPAR6=	177674
PDR1	030332	SDPAR3=	172266	SW08	= 000400	TAB5	003300	UDPAR7=	177676
PIR	= 177772	SDPAR4=	172270	SW09	= 001000	TAB5A	003310	UDPDR0=	177620
PIRQ	= 177772	SDPAR5=	172272	SW1	= 000002	TAB6	003320	UDPDR1=	177622

Symbol table

UDPDR2=	177624	\$ATY1	032652	\$ENDAD	031732	\$MAIL	001200	\$SVPC =	000232
UDPDR3=	177626	\$ATY3	032660	\$ENDCT	031700	\$MAMS1	001230	\$SWR =	167400
UDPDR4=	177630	\$ATY4	032670	\$ENDMG	031751	\$MAMS2	001234	\$SWREG	001222
UDPDR5=	177632	\$AUTO8	001134	\$ENULL	031746	\$MAMS3	001240	\$SWRMK =	000300
UDPDR6=	177634	\$BASE	001254	\$ENV	001220	\$MAMS4	001244	\$TESTN	001204
UDPDR7=	177636	\$BDADR	001122	\$ENVM	001221	\$MBADR	000234	\$TIMES	001164
UFDFLG	004152	\$BDDAT	001126	\$EOP	031632	\$MFLG	033114	\$TKB	001146
UFDSET=	000001	\$BELL	001170	\$EOPCT	031672	\$MNEW	034701	\$TKS	001144
UIPAR0=	177640	\$CDW1	001260	\$ERFLG	001103	\$MSGAD	001214	\$TMPO	001160
UIPAR1=	177642	\$CDW2	001262	\$ERMAX	001115	\$MSGLG	001216	\$TMP1	001162
UIPAR2=	177644	\$CHARC	033450	\$ERROR	032270	\$MSGTY	001200	\$TN =	000031
UIPAR3=	177646	\$CKSWR	034126	\$ERRPC	001116	\$MSWR	034670	\$TPB	001152
UIPAR4=	177650	\$CMTAG	001100	\$ERRTB	001324	\$MTYP1	001231	\$TPFLG	001157
UIPAR5=	177652	\$CM3 =	000000	\$ERTTL	001112	\$MTYP2	001235	\$TPS	001150
UIPAR6=	177654	\$CM4 =	000002	\$ESCAP	001166	\$MTY 3	001241	\$TRAP	035014
UIPAR7=	177656	\$CNTLG	034663	\$ETABL	001220	\$MTYP4	001245	\$TRAP2	035036
UIPDR0=	177600	\$CNTLU	034656	\$ETEND	001324	\$MXCNT	032266	\$TRP =	000013
UIPDR1=	177602	\$CPUOP	001226	\$FATAL	001202	\$NULL	001154	\$TRPAD	035050
UIPDR2=	177604	\$CRLF	001175	\$FFLG	033116	\$NWTST=	000000	\$TSTM	000236
UIPDR3=	177606	\$DBLK	034116	\$FILLC	001156	\$OCNT	033676	\$TSTMH	001102
UIPDR4=	177610	\$DDW0	001264	\$FILLS	001155	\$OMODE	033700	\$TTYIN	034646
UIPDR5=	177612	\$DDW1	001266	\$GDADR	001120	\$OVER	032244	\$TYPDS	033702
UIPDR6=	177614	\$DDW10	001310	\$GDDAT	001124	\$PASS	001206	\$TYPE	033120
UIPDR7=	177616	\$DDW11	001312	\$GET42	031722	\$PASTM	000240	\$TYPEC	033332
UQUIET	004154	\$DDW12	001314	\$GTSWR	034176	\$POWER	035244	\$TYPEX	033452
VIREOP	017724	\$DDW13	001316	\$HD =	000001	\$PWRDN	035076	\$TYPOC	033500
VMKOR	004156	\$DDW14	001320	\$HIBTS	000232	\$PWRMG	035232	\$TYPON	033514
VQBE1	002346	\$DDW15	001322	\$HIOCT	035012	\$PWRUP	035150	\$TYPOS	033454
VQBE2	002376	\$DDW2	001270	\$ICNT	001104	\$QUES	001174	\$UNIT	001212
VQPR1	002350	\$DDW3	001272	\$ILLUP	035236	\$RDCHR	034410	\$UNITM	000242
VQPR2	002400	\$DDW4	001274	\$INTAG	001135	\$RDLIN	034540	\$USWR	001224
WC	002342	\$DDW5	001276	\$ITEMB	001114	\$RDOCT	034712	\$VECT1	001250
WC2	002372	\$DDW6	001300	\$LF	001176	\$RDSZ =	000010	\$VECT2	001252
WLDTRP	030522	\$DDW7	001302	\$LFLG	033115	\$RTNAD	031744	\$XOFF =	000023
XBUF =	177566	\$DDW8	001304	\$LPADR	001106	\$SAVR6	035242	\$XON =	000021
XBUF1 =	176506	\$DDW9	001306	\$LPERR	001110	\$SCOPE	031766	\$XTSTR	032006
XCSR =	177564	\$DEVCT	001210	\$MADR1	001232	\$SETUP=	000137	\$\$GET4=	000000
XCSR1 =	176504	\$DEVM	001256	\$MADR2	001236	\$STUP =	177777	\$OFILL	033677
\$APTHD	000232	\$DOAGN	031742	\$MADR3	001242	\$SVLAD	032210	.\$X =	000232
\$ATYC	032676	\$DTBL	034106	\$MADR4	001246				

. ABS. 035254 000 (RW,I,GBL,ABS,OVR)
 000000 001 (RW,I,LCL,REL,CON)

Errors detected: 0

*** Assembler statistics

Work file reads: 400
 Work file writes: 301
 Size of work file: 55224 Words (216 Pages)
 Size of core pool: 19684 Words (75 Pages)
 Operating system: RSX-11M/PLUS (Under VAX/VMS)

Elapsed time: 00:02:59.53
 COKDDB0.COKDDB0/-SP=ORION.MLB/ML,COKDDB0.MAC/DS:GBL