

Micro Fiche Scan

Name of device(s) tested:

KDJ11-DA

Test description:

KDJ11-DA CLUSTER DIAG

MAINDEC Number or Package Identifier (after SEP 1977):

COKDDA0

Fiche Document Part Number:

AH-FG68A-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

U
A B
COKDDAO
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39

.REME

IDENTIFICATION

PRODUCT CODE: AC-FG67A-MC
PRODUCT NAME: COKDDAO KDJ11-DA CLUSTER DIAG.
PRODUCT DATE: JANUARY,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- 1119	OPERATIONAL SWITCH SETTINGS
11- 1123	MEMORY MANAGEMENT DEFINITIONS
11- 1124	BASIC DEFINITIONS
11- 1126	TRAP CATCHER
11- 1134	ACT11 HOOKS
11- 1135	APT PARAMETER BLOCK
12- 1136	COMMON TAGS
12- 1136	APT MAILBOX-ETABLE
13- 1136	ERROR POINTER TABLE
13- 1139	ERROR DEFINITIONS
13- 1448	GLOBAL VARIABLES AND REGISTER NAMES
13- 1516	GLOBAL DATA SECTION
13- 1706	INITIALIZE THE COMMON TAGS
13- 1718	GET VALUE FOR SOFTWARE SWITCH REGISTER
14- 1766	TEST - NATIVE REGISTER
15- 1842	TEST - 16 BIT ROM CHECKSUM TEST
16- 1878	TEST - KDJ11-DA DATA PATHS
17- 1951	TEST - MEMORY ACCESSABILITY
18- 2007	TEST - MEMORY ERROR REGISTER
19- 2179	TEST - DATA SHORTS AND STUCK AT BITS
20- 2325	TEST - QUICK VERIFY DATA AND ADDRESSING TEST
21- 2409	TEST - CHECK PARITY DETECT LOGIC AND RAMS
22- 2611	TEST - LTC BIT 7
23- 2681	TEST - LKS INTERRUPT PRIORITY
24- 2780	TEST - RESETTING LKS
25- 2832	TEST - MAINTENANCE REGISTER TEST
26- 2865	TEST - SERIAL LINE UNIT REGISTERS
27- 2922	TEST - XCSR BIT 7
28- 2974	TEST - RCSR BIT 7 AND XCSR BIT 2
29- 3060	TEST - RESET AND XCSR<2!0>
30- 3098	TEST - RESET AND INTERRUPT ENABLE BITS
31- 3240	TEST - INTERRUPT PRIORITY FOR SLU
32- 3397	TEST - BREAK CONDITION
33- 3489	TEST - OVERRUN CCNDITION
34- 3568	TEST - LED'S ON
35- 3626	TEST - DIFFERENT LEVELS OF INTERRUPTS
36- 3709	TEST - ARBITRATION BETWEEN PIRQ'S AND INTERRUPTS
38- 3775	GLOBAL ERROR MESSAGES
39- 3912	MODIFIED ERROR MESSAGE TYPEOUT ROUTINE
39- 3978	GLOBAL SUBROUTINES SECTION
42- 4118	Q22BE SIZE ROUTINE
42- 4180	Q22BE INTERRUPT INITIALISE ROUTINE
42- 4191	DMATRN DATO CYCLE THRU Q22BE
42- 4210	DMARD DATI THRU Q22BE
50- 4605	END OF PASS ROUTINE
50- 4606	SCOPE HANDLER ROUTINE
50- 4607	ERROR HANDLER ROUTINE
50- 4607	ABORT ROUTINE FOR LCP/ORTON UFD MODE
50- 4608	APT COMMUNICATIONS ROUTINE
50- 4609	TYPE ROUTINE
50- 4610	BINARY TO OCTAL (ASCII) AND TYPE
50- 4611	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
50- 4612	TTY INFUT ROUTINE
50- 4613	READ AN OCTAL NUMBER FROM THE TTY
50- 4614	TRAP DECODER
50- 4614	TRAP TABLE
50- 4615	POWER DOWN AND UP ROUTINES

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
93

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

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
125

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

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
183

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" library as 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:
17777572=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

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
240

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 TYPEOUTS
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 't', 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 DCING Q22BE INTERRUPTS

241
242
243
244
245

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

Note: th's may not correspond to the actual Program Counter.

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
296

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 LEDs)

ENDTST

TIMOUT: Clean stack
Error NATIVE register timed out

298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319

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

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
377

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 TIMOUT  
} 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

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

380
381 BGNTST
382 } Set timeout trap to TIMEOUT
383 } Set timeout priority to 7
384 } For MSB_Address := 200000 to 177777 D0
385 } Contents of address := 0
386 } Go to the next test

387
388 } TIMEOUT:
389 } Error Address should not have timed out
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 }
434 }

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 TIMEOUT
} Setup timeout VECTOR Priority to 7
} Setup Parity abort VECTOR to PARINI
} 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 1777777 to 0

Note: The KDJ11-DA board uses 256k X 1 chips. This allows us to use only 2 patterns per RAM. If one bit 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.

435
 436
 437
 438
 439
 440
 441
 442
 443
 444
 445
 446
 447
 448
 449
 450
 451
 452
 453
 454
 455
 456
 457
 458
 459
 460
 461
 462
 463
 464
 465
 466
 467
 468
 469
 470
 471
 472
 473
 474
 475
 476
 477
 478
 479
 480
 481
 482
 483
 484
 485
 486
 487
 488
 489
 490
 491

```

BGNTST
  FOR MSB_Address := 0 to 1777776 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 := 1777776 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
  
```

TEST - QUICK VERIFY DATA AND ADDRESSING 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

TEST CHECK PARITY DETECT LOGIC AND RAMS

CHECK PARITY DETECT LOGIC

This test will check out the parity detect'ion 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.

```

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
548

```

```

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

```

54
 550
 551
 552
 553
 554
 555
 556
 557
 558
 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
 605

```

} READ FIRST 2 WORDS OF ROM
  GET EXPECTED CHECKSUM

} DO CALCULATE CHECKSUM
  ADD WORD TO REGISTER
  DO UNTIL ALL WORDS ADDED
  END CALCULATION

} NEGATE THE SUM OF WORDS
  IF NEG. SUM < > EXPECTED CHECKSUM
  ERROR
ENDTST
  
```

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 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 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 's not t'cking
ENDTST
  
```

TIMOUT: Clean stack
 Error LTC register timed out

```
606
607
608 TEST - LKS INTERRUPT PRIORITY
609
610 CHECK THAT LKS INTERRUPTS HAPPEN AT PRIORITY 5 CLEARING LKS<07>
611 AND DON'T HAPPEN AT PRIORITY 6.
612 ROUTINE TEST
613 IF UFD AND LKS IS DISABLED THEN
614     EXIT TEST
615 ENDIF
616 .
617 .   SET PRIORITY TO 5
618 .   CLEAR INTERRUPT FLAG
619 .   LET LKS<06>=#1 (ENABLE INTERRUPTS)
620 .   SET COUNTER TO WAIT FOR 3 INT_RRUPTS
621 .   REPEAT
622 .       DECREMENT COUNTER
623 .       UNTIL INTERRUPT_FLAG EQ #3 OR COUNTER EQ #0
624 .       CLEAR LKS<06>
625 .       IF LKS<07> EQ #1 THEN
626 .           ERROR (WAS NOT CLEARED ON INTERRUPT)
627 .       ENDIF
628 .       IF COUNTER LT TIME_REQUIRED_FOR_3_INTERRUPTS_FOR_800HZ
629 .           ERROR (INTERRUPTS NEVER GO LOW)
630 .       ENDIF
631 .       IF INTERRUPT_FLAG LT #3 THEN
632 .           ERROR (INTERRUPTS DON'T HAPPEN)
633 .       ENDIF
634 .       CLEAR INTERRUPT_FLAG
635 .       WAIT FOR LKS<7>=1
636 .       LET LKS<7>=0
637 .       IF LKS<7> NE #0 THEN
638 .           ERROR (LKS<7> NOT CLEARED)
639 .       ENDIF
640 .       SET PRIORITY TO 6
641 .       SET COUNTER TO 1 SLOW CLOCK INTERRUPT
642 .       SET LKS<06>
643 .       REPEAT
644 .           DECREMENT COUNTER
645 .           UNTIL COUNTER EQ #0 OR INTERRUPT_FLAG NE #0
646 .           IF INTERRUPT_FLAG NE #0 THEN
647 .               ERROR (INTERRUPT WAS AT WRONG PRIORITY)
648 .           ENDIF
649 .       RESTORE ORIGINAL PRIORITY
650 ENDROUTINE
651 ROUTINE LINE CLOCK INTERRUPT
652     INCREMENT INTERRUPT_FLAG
653 ENDROUTINE
654
655
656 TEST   RESETTING LKS
657
658 RESETTING LKS(*)
659 THIS TEST WILL PROVE THAT RESET INSTRUCTION CLEARS LKS<06>.
660 ROUTINE TEST
661 IF UFD AND LKS IS DISABLED THEN
662
```



```

663          EXIT TEST
664      ÉNDIF
665      . POINT LKS VECTOR 100 TO ERROR LKS ILLEGAL_INTERRUPT
666      . SYNCHRONIZE LKS BY WAITING FOR 3 PULSES
667      . LET LKS<06>=#1
668      . EXECUTE "RESET"
669      . IF LKS<6> NE #0 THEN
670      .     ERROR
671      . ÉNDIF
672      . IF ILLEGAL_LINE_CLOCK_INTERRUPT NE 0 THEN
673      .     ERROR
674      . ÉNDIF
675  ÉNDROUTINE
676
677  ROUTINE ERROR_LKS_ILLEGAL_INTERRUPT
678      FLAG_ILLEGAL_LINE_CLOCK_INTERRUPT
679  RETURN
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
719

```

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 LIN_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 UFD AND CONSOLE NOT PRESENT
    GO TO TEST_22
ÉNDIF
    IF BCR<07> EQ #0 THEN

```

```

720          .          READ BCR<2-0> TO GET BAUD RATE
721          .          ENDIF
722          .          LET 4=ADDRESS OF TIMEOUT ROUTINE
723          .          DO FOR RCSR,XCSR,RBUF,XBUF
724          .          .          READ XCSR,XCSR,RBUF,XBUF
725          .          .          IF TIMEOUT FLAG NE #0 THEN
726          .          .          .          ERROR
727          .          .          .          ENDIF
728          .          .          ENDDO
729          .          ENDRoutine
730
731          ROUTINE TIMEOUT
732          .          LET TIMEOUT_FLAG=#1
733          .          ENDRoutine
734
735
736
737          TEST      XCSR BIT 7
738
739          CHECK THAT XCSR<07> CAN BE 0 AND 1.
740
741          XCSR      <07>          TRANSMITTER READY
742
743          ROUTINE TEST
744          .          WAIT FOR XCSR<07>=#1 NO MORE THAN 200MSEC
745          .          IF XCSR<07> NE #1 THEN
746          .          .          ERROR
747          .          .          ENDF
748          .          .          LET XBUF=#NULL
749          .          .          WAIT FOR XCSR<07>=#1
750          .          .          LET XBUF=#NULL
751          .          .          IF XCSR<07> NE 0 THEN
752          .          .          .          ERROR (READY DIDN'T GO LOW)
753          .          .          ENDF
754          .          ENDRoutine
755
756
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          .          .          .          ERROR (RCSR<07> DOES NOT BECOME 1 OR XCSR<02> DOES NOT
773          .          .          .          WORK)
774          .          .          ENDF
775          .          .          IF RBUF NE #125 THEN
776          .          .          .          ERROR

```

```

777      .      ENDIF
778      .      IF RCSR<07> NE #0 THEN
779      .          ERROR (RCSR<07>DOES NOT GO LOW)
780      .      ENDIF
781      .      LET XCSR<02>=#0
782      .      ENDROUTINE
783
784
785
786      TEST    RESET AND XCSR<2!0>
787
788      CHECK THAT RESET CLEARS XCSR<0!2>.
789      ROUTINE TEST
790      .      (CHECK RCSR<07> AND XCSR<07> AND RESET)
791      .      LET XCSR<02,00>=#1 (LOOP BACK MODE)
792      .      EXECUTE "RESET"
793      .      IF XCSR<02!00> NE #0 THEN
794      .          ERROR
795      .      ENDIF
796      .      LET XCSR<02>=#0
797      .      ENDROUTINE
798
799
800
801      TEST    - RESET AND INTERRUPT ENABLE BITS
802
803      CHECK THAT INTERRUPTS DON'T HAPPEN AT PRIORITY 4 AND THAT RESET
804      CLEARS XCSR<06> AND RCSR<06>.
805
806      RCSR    <06>    RECEIVER INTERRUPT ENABLE
807      XCSR    <06>    TRANSMITTER INTERRUPT ENABLE
808
809      ROUTINE TEST
810      .      LET 60=#ADDRESS_OF_ILLEGAL_INTERRUPT_XRCSR
811      .      LET 64=#ADDRESS_OF_ILLEGAL_INTERRUPT_XRCSR
812      .      SET PRIORITY TO 4
813      .      LET XCSR<02>=#1 (LOOPBACK MODE)
814      .      LET XCSR<06>=#1 (ENABLE TRANSMIT INTERRUPTS)
815      .      LET RCSR<06>=#1 (ENABLE RECEIVE INTERRUPTS)
816      .      WAIT FOR XCSR<07>=#1 (READY TO TRANSMIT)
817      .      LET XBUF=#NULL (SEND A CHARACTER)
818      .      WAIT FOR ILLEGAL INTERRUPTS (ABOUT 200MSEC)
819      .      EXECUTE "RESET"
820      .      IF XCSR<06> NE #0 OR RCSR<06> NE #0 OR XRCSR NE #0 THEN
821      .          ERROR
822      .      ENDIF
823      .      RESTORE PRIORITY TO NORMAL
824      .      ENDROUTINE
825
826      ROUTINE ILLEGAL_INTERRUPT_XRCSR
827      .      INCREMENT XRCSR
828      .      ENDROUTINE
829
830
831
832      TEST    - INTERRUPT PRIORITY FOR SLU
833

```

```

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

```

891      .      WAIT FOR RCSR<07>=#1
892      .      IF RBUF<15!13!11> NE #0 THEN
893      .          ERROR
894      .      ENDIF
895      .      LET XCSR<00>=#1
896      .      EXECUTE "RESET"
897      .      IF XCSR<00> NE #0 THEN
898      .          ERROR
899      .      ENDIF
900      .      LET XCSR<02>=#0
901      .      ENDROUTINE
902
903
904
905      TEST      OVERRUN CONDITION
906
907      CHECK OVERRUN CONDITION
908
909      RCSR      <14>      OVERRUN ERROR
910
911      SLU #1 IS TESTED SINCE SLU #0 IS BEING USED AS THE CONSOLE
912
913      ROUTINE TEST
914      .      LET XCSR<02>=#1 (LOOPBACK MODE)
915      .      WAIT FOR XCSR<07>=#1
916      .      LET XBUF=#252
917      .      WAIT FOR XCSR<07>=#1
918      .      LET XBUF=#125 (SEND THE 2ND W/O READING THE 1ST CHARACTER)
919      .      WAIT FOR RCSR<07>=#1
920      .      STALL FOR LOWEST BAUD RATE TO GET 2ND CHARACTER
921      .      IF LOW BYTE OF RBUF NE #125 THEN
922      .          ERROR (1ST CHARACTER WASN'T OVERRUN)
923      .      ENDIF
924      .      IF RBUF<15!14> NE #1 THEN
925      .          ERROR (NO OVERRUN BIT SET)
926      .      ENDIF
927      .      WAIT FOR XCSR<07>=#1
928      .      LET XBUF=#NULL
929      .      WAIT FOR RCSR<07>=#1
930      .      IF RBUF<15!14> NE #0 THEN
931      .          ERROR (WASN'T CLEARED ON THE NEXT CHARACTER RECEIVED)
932      .      ENDIF
933      .      LET XCSR<02>=#0
934      .      ENDROUTINE
935
936
937
938
939
940
941
942
943
944
945      3.8 LED'S TEST CODE
946      This test will determine that the LED's situated on the CPU board
947

```

948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969

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

```
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
```

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
 1027

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
.   
```

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
1073

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 accessability
test 35	Memory Error Register
test 36	Data shport 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 Levels of interrupt
test 55	Arbitration between PIRQ's and interrupts

1075
1076
1077
1078
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1098
1099
1100
1101
1102
1103
1104

3.11 PROGRAM UPDATES AND MODIFICATIONS

Version COKDDAO

10-1-85

Michael Charchaflian

4.0 BIBLIOGRAPHY

N/A

5.0 GLOSSARY

N/A

APPENDICES

N/A

E

1116 167400
1117 000300
1118

\$SWR=167400
\$SWRMK=300
.TITLE COKDDAO KDJ11-DA CLUSTER DIAG.
;*COPYRIGHT (C) SEPT 85
;*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.

1119 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

1121
1122
1123

.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 REGISTERS

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

UIPARC= 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      UDPARO= 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      SIPARO= 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
            .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

```

1124

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 TRIVEC= 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
1125 000001 UFDSET= 1 ;FLAG FOR UFD
1126 .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
1127 000174 000000 DISPREG: .WORD 0 ;;SOFTWARE DISPLAY REGISTER
000176 000000 SWREG: .WORD 0 ;;SOFTWARE SWITCH REGISTER
1128 000200 005037 001160 CLR $TMPO
1129 000204 000137 004054 JMP @#START
1130 000220 000220 =220
1131 000220 012737 000777 001160 MOV #777,$TMPO
1132 000226 000137 004054 JMP @#START
1133
1134 .SBTTL ACT11 HOOKS
;*****
;HOOKS REQUIRED BY ACT11
000232 $SVPC=. ;SAVE PC
000046 000046 =46
000046 030114 $ENDAD ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
000052 000052 =52
000052 000000 .WORD 0 ;;2)SET LOC.52 TO ZERO
1135 000232 =$SVPC ;; RESTORE PC
.SBTTL APT PARAMETER BLOCK
;*****
;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
;*****
000024 000232 . $X=. ;;SAVE CURRENT LOCATION
000024 000024 =24 ;;SET POWER FAIL TO POINT TO START OF PROGRAM
200 200 ;;FOR APT START UP
000044 000044 =44 ;;POINT TO APT INDIRECT ADDRESS PNTR.
000044 000232 $APTHDR ;;POINT TO APT HEADER BLOCK
000232 000232 .=.$X ;;RESET LOCATION COUNTER
;*****
;SETUP APT PARAMETER BLOCK AS CTAINED 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	\$MBADR:	.WORD	\$MAIL	::ADDRESS OF APT MAILBOX (BITS 0-15)
000236	000000	\$TSTM:	.WORD		::RUN TIM OF LONGEST TEST
000240	000000	\$PASTM:	.WORD		::RUN TIME IN SECS. OF 1ST PASS ON 1 UNIT (QUICK VERIFY)
000242	000000	\$UNITM:	.WORD		::ADDITIONAL RUN TIME (SECS) OF A PASS FOR EACH ADDITIONAL UNIT
000244	000052		.WORD	\$ETEND-\$MAIL/2	::LENGTH MAILBOX-ETABLE(WORDS)

COMMON TAGS

1136

```

.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 0000CC $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

```


G3

APT MAILBOX-ETABLE

001216	000000	\$MSGLG: .WORD	AMSGLG	:: MESSAGE LENGTH
001220		\$ETABLE:		:: APT ENVIRONMENT TAB_E
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#2 BUS PRIORITY#2
001254	000000	\$BASE: .WORD	ABASE	:: BASE ADDRESS OF EQUIPMENT UNDER TEST
001256	000000	\$DEVN: .WORD	ADEVN	:: 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

```

1137
1138
1139
1140
1141 001324 017151
1142 001326 023514
1143 001330 024702
1144 001332 000000
1145
1146 001334 017205
1147 001336 023514
1148 001340 024702
1149 001342 000000
1150
1151 001344 017217
1152 001346 023514
1153 001350 024702
1154 001352 000000
1155
1156 001354 017231
1157 001356 023514
1158 001360 024702
1159 001362 000000
1160
1161 001364 017267
1162 001366 023514
1163 001370 024702
1164 001372 000000
1165
1166 001374 017313
1167 001376 023514
1168 001400 024702
1169 001402 000000
1170
1171 001404 017425
1172 001406 023514
1173 001410 024702
1174 001412 000000
1175
1176 001414 017464
1177 001416 024345
1178 001420 025136
1179 001422 000000
1180
1181 001424 017534
1182 001426 023514

```

```

.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

```

ERRCR DEFINITIONS

```

1183 001430 024702          DT1          ;$TMP1,$ERRPC
1184 001432 000000          0
1185      ;ITEM 12
1186 001434 017570          EM72          ;TIMEOUT READING SLU REGISTERS
1187 001436 024411          DH72          ;TEST #, PC, ADDRESS FAILED
1188 001440 025030          DT41          ;$TMP1,$ERRPC,$BDDAT
1189 001442 000000          0
1190      ;ITEM 13
1191 001444 017626          EM73          ;XMIT READY DIDN'T GO LOW
1192 001446 023514          DH1           ;TEST #, PC
1193 001450 024702          DT1           ;$TMP1,$ERRPC
1194 001452 000000          0
1195      ;ITEM 14
1196 001454 017652          EM74          ;RCSR DOESN'T BECOME 1
1197 001456 023514          DH1           ;TEST #, PC
1198 001460 024702          DT1           ;$TMP1,$ERRPC
1199 001462 000000          0
1200      ;ITEM 15
1201 001464 017703          EM75          ;WRONG CHARACTER RECEIVED
1202 001466 023541          DH4           ;TEST #, PC, EXPECTED DATA, RECEIVED DATA
1203 001470 025146          DT75          ;$TMP1,$ERRPC,$GDDAT,$BDDAT
1204 001472 000000          0
1205      ;ITEM 16
1206 001474 017734          EM76          ;RCSR<07> NOT CLEARED AFTER READING RBUF
1207 001476 023514          DH1           ;TEST #, PC
1208 001500 024702          DT1           ;$TMP1,$ERRPC
1209 001502 000000          0
1210      ;ITEM 17
1211 001504 020004          EM77          ;XCSR<07> NOT SET ON RESET
1212 001506 023514          DH1           ;TEST #, PC
1213 001510 024702          DT1           ;$TMP1,$ERRPC
1214 001512 000000          0
1215      ;ITEM 20
1216 001514 020036          EM100         ;RCSR<07> NOT CLEARED ON RESET
1217 001516 023514          DH1           ;TEST #, PC
1218 001520 024702          DT1           ;$TMP1,$ERRPC
1219 001522 000000          0
1220      ;ITEM 21
1221 001524 020074          EM101         ;SLU INTERRUPTS HAPPEN AT 4
1222 001526 023514          DH1           ;TEST #, PC
1223 001530 024702          DT1           ;$TMP1,$ERRPC
1224 001532 000000          0
1225      ;ITEM 22
1226 001534 020127          EM102         ;RESET DOES NOT CLEAR XCSR<6> AND RSCR<6>
1227 001536 023514          DH1           ;TEST #, PC
1228 001540 024702          DT1           ;$TMP1,$ERRPC
1229 001542 000000          0
1230      ;ITEM 23
1231 001544 020211          EM103         ;TRANSMIT INTERRUPT DOES NOT CLEAR XCSR<07>
1232 001546 023514          DH1           ;TEST #, PC
1233 001550 024702          DT1           ;$TMP1,$ERRPC
1234 001552 000000          0
1235      ;ITEM 24
1236 001554 020264          EM104         ;RECEIVE INTERRUPTS DON'T CLEAR RCSR<07>
1237 001556 023514          DH1           ;TEST #, PC
1238 001560 024702          DT1           ;$TMP1,$ERRPC
1239 001562 000000          0

```

ERROR DEFINITIONS

1240					
1241	001564	020334	;ITEM 25	EM105	;BREAK CONDITION DOES NOT SET RBUF PROPERLY
1242	001566	024455		DH105	;TEST #, PC, RBUF
1243	001570	025160		DT105	;\$TMP1,\$ERRPC,RBUF
1244	001572	000000		0	
1245			;ITEM 26	EM106	;RBUF WASN'T CLEARED ON NEXT CHAR.
1246	001574	020407		DH105	;TEST #, PC, RBUF
1247	001576	024455		DT105	;\$TMP1,\$ERRPC,RBUF
1248	001600	025160		0	
1249	001602	000000			
1250			;ITEM 27	EM107	;ERROR IN WRITING TO XCSR<0>
1251	001604	020465		DH1	;TEST #, PC
1252	001606	023514		DT1	;\$TMP1,\$ERRPC
1253	001610	024702		0	
1254	001612	000000			
1255			;ITEM 30	EM110	;RESET DOES NOT CLEAR XCSR<00>
1256	001614	020521		DH1	;TEST #, PC
1257	001616	023514		DT1	;\$TMP1,\$ERRPC
1258	001620	024702		0	
1259	001622	000000			
1260			;ITEM 31	EM111	;FIRST CHARACTER WAS NOT OVERRUN BY THE SECOND
1261	001624	020563		DH4	;TEST #, PC, EXPECTED DATA, RECEIVED DATA
1262	001626	023541		DT75	;\$TMP1,\$ERRPC,\$GDDAT,\$BDDAT
1263	001630	025146		0	
1264	001632	000000			
1265			;ITEM 32	EM112	;OVERRUN CONDITION DOES NOT SET PROPER BITS IN RBUF
1266	001634	020641		DH105	;TEST #, PC, RBUF
1267	001636	024455		DT105	;\$TMP1,\$ERRPC,RBUF
1268	001640	025160		0	
1269	001642	000000			
1270			;ITEM 33	EM113	;RBUF WAS NOT CLEARED ON THE NEXT CHARACTER
1271	001644	020724		DH105	;TEST #, PC, RBUF
1272	001646	024455		DT105	;\$TMP1,\$ERRPC,RBUF
1273	001650	025160		0	
1274	001652	000000			
1275			;ITEM 34	EM114	;ERROR IN XCSR<2>
1276	001654	021010		DH1	;TEST #, PC
1277	001656	023514		DT1	;\$TMP1,\$ERRPC
1278	001660	024702		0	
1279	001662	000000			
1280			;ITEM 35	EM124	;PIRQ INTERRUPTS DON'T TAKE PRIORITY OVER Q BUS INTERRUPTS
1281	001664	021063		DH1	;TEST #, PC
1282	001666	023514		DT1	;\$TMP1,\$ERRPC
1283	001670	024702		0	
1284	001672	000000			
1285			;ITEM 36	EM125	;NO POWER DOWN TRAP TO 24 OCCUP
1286	001674	021155		DH1	;TEST #, PC
1287	001676	023514		DT1	;\$TMP1,\$ERRPC
1288	001700	024702		0	
1289	001702	000000			
1290			;ITEM 37	EM126	;ERROR IN INTERRUPTS FROM Q228E
1291	001704	021214		DH1	;TEST #, PC
1292	001706	023514		DT1	;\$TMP1,\$ERRPC
1293	001710	024702		0	
1294	001712	000000			
1295			;ITEM 40	EM127	;ERROR IN PMG COUNTER
1296	001714	021251			

ERROR DEFINITIONS

1297	001716	023514	DH1	:TEST #, PC
1298	001720	024702	DT1	;\$TMP1,\$ERRPC
1299	001722	000000	0	
1300			:ITEM 41	
1301	001724	021313	EM130	:UNEXPECTED TIMEOUT
1302	001726	023514	DH1	:TEST #, PC
1303	001730	025204	DT130	;\$TMP1,\$ERRPC
1304	001732	000000	0	
1305			:ITEM 42	
1306	001734	021340	EM131	:ERROR WRITING TO LKS<6>
1307	001736	023514	DH1	:TEST #, PC
1308	001740	024702	DT1	;\$TMP1,\$ERRPC
1309	001742	000000	0	
1310			:ITEM 43	
1311	001744	021370	EM132	:MAINTENANCE REGISTER ERROR
1312	001746	023514	DH1	:TEST #, PC
1313	001750	024702	DT1	;\$TMP1,\$ERRPC
1314	001752	000000	0	
1315			:ITEM 44	
1316	001754	021426	EM135	: ERROR IN THE DATA PATHS
1317	001756	023541	DH4	:TEST #, PC, EXPECTED DATA, RECEIVED DATA
1318	001760	025146	DT75	;\$TMP1,\$L (RPC,\$GDDAT,\$BDDAT
1319	001762	000000	0	
1320			:ITEM 45	
1321	001764	021464	EM136	:TIMED OUT IN ACCESSING LOCATION 0
1322	001766	023514	DH1	:TEST #, PC
1323	001770	024702	DT1	;\$TMP1,\$ERRPC
1324	001772	000000	0	
1325			:ITEM 46	
1326	001774	021526	EM137	:TIMED OUT IN TRYING TO ACCESS MEMORY
1327	001776	023514	DH1	:TEST #, PC
1328	002000	024702	DT1	;\$TMP1,\$ERRPC
1329	002002	000000	0	
1330			:ITEM 47	
1331	002004	021573	EM140	:ERROR IN FUNCTIONAL REV BITS ON NATIVE REGISTER
1332	002006	023514	DH1	:TEST #, PC
1333	002010	024702	DT1	;\$TMP1,\$ERRPC
1334	002012	000000	0	
1335			:ITEM 50	
1336	002014	021664	EM141	:ERROR IN INDICATOR BITS ON THE NATIVE REGISTER
1337	002016	023514	DH1	:TEST #, PC
1338	002020	024702	DT1	;\$TMP1,\$ERRPC
1339	002022	000000	0	
1340			:ITEM 51	
1341	002024	021747	EM142	:ERROR IN THE BOOT SELECT SWITCHES ON THE NATIVE REGISTER
1342	002026	023514	DH1	:TEST #, PC
1343	002030	024702	DT1	;\$TMP1,\$ERRPC
1344	002032	000000	0	
1345			:ITEM 52	
1346	002034	022040	EM143	:TIMED OUT IN ACCESSING THE NATIVE REGISTER
1347	002036	023514	DH1	:TEST #, PC
1348	002040	024702	DT1	;\$TMP1,\$ERRPC
1349	002042	000000	0	
1350			:ITEM 53	
1351	002044	022113	EM144	:LTC MONITOR IS NOT TOGGLING
1352	002046	023514	DH1	:TEST #, PC
1353	002050	024702	DT1	;\$TMP1,\$ERRPC

ERROR DEFINITIONS

1354	002052	000000		
1355			:ITEM 54	0
1356	002054	022170		EM145
1357	002056	023514		DH1
1358	002060	024702		DT1
1359	002062	000000		0
1360			:ITEM 55	0
1361	002064	022236		EM146
1362	002066	023514		DH1
1363	002070	024702		DT1
1364	002072	000000		0
1365			:ITEM 56	0
1366	002074	022301		EM147
1367	002076	023541		DH4
1368	002100	025146		DT75
1369	002102	000000		0
1370			:ITEM 57	0
1371	002104	022356		EM150
1372	002106	023514		DH1
1373	002110	024702		DT1
1374	002112	000000		0
1375			:ITEM 60	0
1376	002114	022437		EM151
1377	002116	023514		DH1
1378	002120	024702		DT1
1379	002122	000000		0
1380			:ITEM 61	0
1381	002124	022505		EM152
1382	002126	024564		DH134
1383	002130	025212		DT134
1384	002132	000000		0
1385			:ITEM 62	0
1386	002134	022567		EM153
1387	002136	023514		DH1
1388	002140	024702		DT1
1389	002142	000000		0
1390			:ITEM 63	0
1391	002144	022647		EM154
1392	002146	023514		DH1
1393	002150	024702		DT1
1394	002152	000000		0
1395			:ITEM 64	0
1396	002154	022734		EM155
1397	002156	023541		DH4
1398	002160	025146		DT75
1399	002162	000000		0
1400			:ITEM 65	0
1401	002164	023007		EM156
1402	002166	023541		DH4
1403	002170	025146		DT75
1404	002172	000000		0
1405			:ITEM 66	0
1406	002174	023061		EM157
1407	002176	024564		DH134
1408	002200	025212		DT134
1409	002202	000000		0
1410			:ITEM 67	0

ERROR DEFINITIONS

```

1411 002204 023127      EM160      ;ERROR IN XCSR <6>
1412 002206 023514      DH1        ;TEST #, PC
1413 002210 024702      DT1        ;$TMP1,$ERRPC
1414 002212 000000      0
1415      ;ITEM 70
1416 002214 023151      EM161      ;NO XMIT INTERRUPTS HAVE OCCURED
1417 002216 023514      DH1        ;TEST #, PC
1418 002220 024702      DT1        ;$TMP1,$ERRPC
1419 002222 000000      0
1420      ;ITEM 71
1421 002224 023211      EM162      ;NO RECIEVE INTERRUPTS HAVE OCCURED
1422 002226 023514      DH1        ;TEST #, PC
1423 002230 024702      DT1        ;$TMP1,$ERRPC
1424 002232 000000      0
1425      ;ITEM 72
1426 002234 023254      EM163      ;UNEXPECTED PARITY ABORT HAS OCCURED
1427 002236 023514      DH1        ;TEST #, PC
1428 002240 024702      DT1        ;$TMP1,$ERRPC
1429 002242 000000      0
1430      ;ITEM 73
1431 002244 023320      EM164      ;MER DIDN'T HAVE CORRECT ADDRESS BITS 0 AND 15
1432 002246 023541      DH4        ;TEST#, PC, EXTENDED DATA, RECEIVED DATA
1433 002250 025146      DT75       ;$TMP1,$ERRPC,$GDDAT,$BDDAT
1434 002252 000000      0
1435      ;ITEM 74
1436      ;ITEM 74
1437 002254 023377      EM165      ;TOO MANY TRANSIVER INTERRUPTS HAPPENED
1438 002256 023514      DH1        ;TEST #, PC
1439 002260 024702      DT1        ;$TMP1,$ERRPC
1440 002262 000000      0
1441      ;ITEM 75
1442 002264 023446      EM166      ;TOO MANY RECEIVER INTERRUPTS HAPPENED
1443 002266 023514      DH1        ;TEST #, PC
1444 002270 024702      DT1        ;$TMP1,$ERRPC
1445 002272 000000      0
1446
1447
1448      .SBTTL  GLCSAL VARIABLES AND REGISTER NAMES
1449
1450      ;REGISTERS FOR THE FIRST Q22BE
1451 002274 000000      CSR1:      .WORD 0      ;CONTROL REGISTER 1 FOR Q22BE
1452 002276 000000      CSR2:      .WORD 0      ;CONTROL/STATUS REGISTER 2
1453 002300 000000      BA:        .WORD 0      ;DMA ADDRESS FOR Q22BE
1454 002302 000000      WC:        .WORD 0      ;WORD COUNT REGISTER
1455 002304 000000      DATA:     .WORD 0      ;DMA DATA FOR Q22BE
1456 002306 000000      VQBE1:     .WORD 0      ;ADDRESS OF VECTOR FOR Q22BE
1457 002310 000000      VQPR1:     .WORD 0      ;PRIORITY
1458 002312 000000      SIMGOA:    .WORD 0      ;SIMULTANEUOS GO ADDRESS REGISTER
1459
1460 002314 000000      ledcnt:    .word 0      ;location for the led count      ;mc
1461 002316 000000      RCOUNT:   .WORD 0      ;RECEIVER INTERRUPT COUNT      ;MC
1462 002320 000000      TCOUNT:   .WORD 0      ;TRANSMITTER INTERRUPT COUNT    ;MC
1463
1464
1465      ;REGISTERS FOR THE SECOND Q22BE
1466 002322 000000      CSR12:     .WORD 0      ;CONTROL REGISTER 1 FOR Q22BE
1467 002324 000000      CSR22:     .WORD 0      ;CONTROL/STATUS REGISTER 2

```

GLOBAL VARIABLES AND REGISTER NAMES

```

1468 002326 000000      BA2:      .WORD  0      ;DMA ADDRESS FOR Q22BE
1469 002330 000000      WC2:      .WORD  0      ;WORD COUNT REGISTER
1470 002332 000000      DATA2:  .WORD  0      ;DMA DATA FOR Q22BE
1471 002334 000000      VQBE2:   .WORD  0      ;ADDRESS OF VECTOR FOR Q22BE
1472 002336 003000      VQPR2:   .WORD  0      ;PRIORITY
1473
1474 002340 000000      LKS L:   .WORD  0
1475 002342 000000      ACTCHS:  .WORD  0      ;ACTUAL CHECKSUM
1476 002344 000000      SAVPCR:  .WORD  0
1477 002346 000000      SAVBR:   .WORD  0
1478          002740      =2740
1479 002740 000000      TEMP:    .WORD  0
1480 002742          .BLKW 15.      ;RESERVED FOR BLOCK MODE TRANSFER
1481 003000 000000      TIMOUT:  .WORD  0
1482 003002 000002      Q22EN:   .WORD  2      ;PRIORITY 7-4 FOR Q22BE
1483
1484
1485          177524      BCR=      177524      ;BOOT/DIAGNOSTICS CONFIGURATION
1486          177524      BDR=      177524      ;BOOT/DIAGNOSTICS DISPLAY
1487          177520      BCSR=     177520      ;BOOT/DIAGNOSTICS STATUS
1488          177746      CCR=      177746      ;CACHE CONTROL REGISTER
1489          177752      HITMIS=  177752      ;HIT OR MISS REGISTER
1490          177734      KMCR=    177734      ;UNIBUS CONFIGURATION REGISTER
1491          177546      LKS=     177546      ;CLOCK STATUS REGISTER
1492          177750      MAIREG=  177750      ;MAINTENANCE REGISTER
1493          177520      NATREG=  177520      ;NATIVE REGISTER
1494          172100      MER=     172100      ;MEMORY ERROR REGISTER
1495          177744      MSER=    177744      ;MEMORY SYSTEM ERROR
1496          177522      PCR=     177522      ;PAGE CONTROL REGISTER
1497          177772      PIR=     177772      ;PROGRAM INTERRUPT REQUEST
1498          177560      RCSR=    177560      ;RECEIVER STATUS REGISTER
1499          177562      RBUF=    177562      ;RECEIVER DATA BUFFER
1500          177564      XCSR=    177564      ;TRANSMITTER STATUS REGISTER
1501          177566      XBUF=    177566      ;TRANSMITTER DATA BUFFER
1502          176500      RCSR1=   176500      ;RECEIVER STATUS REGISTER
1503          176502      RBUF1=   176502      ;RECEIVER DATA BUFFER
1504          176504      XCSR1=   176504      ;TRANSMITTER STATUS REGISTER
1505          176506      XBUF1=   176506      ;TRANSMITTER DATA BUFFER
1506
1507          177766      CPereg=  177766      ;CPU ERROR REGISTER
1508
1509          177572      MMRO=SR0          ;MEMORY MANAGEMENT REG. 0
1510          177574      MMR1=SR1          ;MEMORY MANAGEMENT REG. 1
1511          177576      MMR2=SR2          ;MEMORY MANAGEMENT REG. 2
1512          172516      MMR3=SR3          ;MEMORY MANAGEMENT REG. 3
1513          120001      POLY= 120001
1514          000000      NULL=            0
1515
1516      .SBTTL GLOBAL DATA SECTION
1517
1518      ;+>
1519      ; THE GLOBAL DATA SECTION CONTAINS DATA THAT ARE USED
1520      ; IN MORE THAN ONE TEST.
1521      ; -
1522
1523      ;THESE LOCATIONS ARE USED IN MORE THAN ONE TEST TO STORE VECTOR DATA
1524      ;WHEN THE TEST NEEDS TO HAVE AN ERROR CONDITION RESPOND DIFFERENTLY

```


GLOBAL DATA SECTION

```

1525      ;FROM THE DEFAULT RESPONCE.
1526 003004 000000 SLOC00: .WORD 0
1527 003006 000000 SLOC01: .WORD 0
1528
1529      ;THESE LOCATIONS ARE USED IN MORE THAN ONE TEST TO STORE WORKING DATA.
1530 003010 000000 PARPAT: .WORD 0 ;LOCATION TO SAVE PATTERN USED IN PARITY TEST
1531 003012 000000 SAVTIM: .WORD 0 ;LOCATION TO THE SAVE THE TIMEOUT TRAP
1532 003014 000000 LOWADD: .WORD 0 ;STORES LOW ADDRESS FOR RAM TESTS
1533 003016 000000 GOODAD: .WORD 0 ;STORES GOOD ADDRESS FOR RAM TESTS
1534 003020 000000 ERRCNT: .WORD 0 ;CONTAINS TOTAL NO. OF EEROM ERRORS
1535 003022 000000 TSTADD: .WORD 0 ;ADDRESS STORE FOR RAM TESTS
1536 003024 000000 NEWADD: .WORD 0 ;ADDRESS STORE FOR RAM TEST
1537 003026 000000 FLAG: .WORD 0 ;USED TO STORE "FLAG" CONDITIONS
1538 003030 000000 CCHPAS: .WORD 0 ; flag-counter for control of Cache subtests
1539 003032 000000 EEPAS: .WORD 0 ; flag-counter for control of EEROM subtest
1540 003034 000000 SAVSUP: .WORD 0 ;USED TO STORE SUPERVISOR STACK VALUE
1541 003036 000000 SAVUSE: .WORD 0 ;USED TO STORE USER STACK VALUE
1542 003040 000000 SAVMR0: .WORD 0 ;USED TO STORE MMU STATUS REGISTER 0 DATA
1543 003042 000000 SAVMR1: .WORD 0 ;USED TO STORE MMU STATUS REGISTER 1 DATA
1544 003044 000000 SAVMR2: .WORD 0 ;USED TO STORE MMU STATUS REGISTER 2 DATA
1545 003046 000000 SAVSWP: .WORD 0 ; SAVE SFTWRE SWTCH REG DURING EEROM TEST
1546 003050 000000 MERTAG: .WORD 0
1547 003052 000000 FLOAT: .BLKW 4 ;USED TO STORE VALUES FOR MMU TESTS
1548 003062 000000 FLO: .BLKW 4 ;USED TO STORE VALUES FOR MMU TESTS
1549 003072 000000 SEQ: .WORD 0 ;STORES SEQUENCE NUMBER FOR JUMP TESTS
1550 003074 000000 SPS: .WORD 0 ;STORES STACK POINTER FOR JUMP TESTS
1551 003076 000000 SPSJ: .WORD 0 ;STORES STACK POINTER FOR JUMP TESTS
1552 003100 000000 BTEXP: .BLKW 4 ;STORES EXPONENT DURING BIT TESTS
1553 003110 000000 BTRES: .BLKW 4 ;STORES RECIEVED DATA FOR BIT TESTS
1554 003120 000000 COUNT: .WORD 0 ;ERROR INDICATOR FOR FLOATING POINT TESTS
1555 003122 000000 RECFEC: .BLKW 4 ;RECIEVED FLOATING POINT EXCEPTION CODE
1556 003132 000000 RECST: .BLKW 4 ;RECIEVED FLOATING POINT STATUS
1557 003142 000000 RECDST: .BLKW 4 ;DESTINATION ADDRESS FOR FLOATING POINT TESTS
1558
1559      ;THESE LOCATIONS ARE USED BY MORE THAN ONE TEST AS LOOP COUNTERS
1560 003152 000000 ALLCTR: .WORD 0
1561 003154 000000 LOOPIN: .WORD 0
1562
1563      ;SOME MORE TEMPORARY STORAGE FOR RAM TESTS
1564 003156 000000 SAVPOS: .WORD 0 ;STORES TEMPORARY BIT POSITIONS FOR RAM TESTS
1565 003160 000000 MASK: .WORD 0 ;STORES BIT MASK FOR ERROR ISOLATION
1566
1567      ;!!!!!!THIS IS IT. THE PROGRAM TEST LOCATION!!!!!!!!!!!!!!!!!!!!!!
1568 003162 000000 †STLOC: .WORD 0
1569 003164 000000 .BLKW 20.
1570      ;FPP REGISTER DEFINITIONS
1571 000000 AC0= *0
1572 000001 AC1= *1
1573 000002 AC2= *2
1574 000003 AC3= *3
1575 000004 AC4= *4
1576 000005 AC5= *5
1577 000006 AC6= *6
1578 000007 AC7= *7
1579
1580      ;FPP INTERRUPT VECTOR
1581

```

GLOBAL DATA SECTION

1582	000244		FPVEC=244
1583			
1584			
1585	001000		DOT= 1000
1586			
1587			
1589	003234	123456	TAB1: .WORD 123456
1590	003236	000000	.WORD 000000
1591	003240	000000	.WORD 0
1592	003242	000001	.WORD 1
1593	003244	055555	TAB2: .WORD 055555
1594	003246	177777	.WORD 177777
1595	003250	145671	.WORD 145671
1596	003252	100000	.WORD 100000
1597	003254	003000	TAB3: .WORD 003000
1598	003256	123456	.WORD 123456
1599	003260	000000	.WORD 0
1600	003262	000000	.WORD 0
1601	003264	055555	TAB4: .WORD 55555
1602	003266	177777	.WORD -1
1603	003270	000000	.WORD 0
1604	003272	000000	.WORD 0
1605	003274	043243	TAB5: .WORD 43243
1606	003276	000000	.WORD 0
1607	003300	000000	.WORD 0
1608	003302	000000	.WORD 0
1609	003304	162400	TAB5A: .WORD 162400
1610	003306	000000	.WORD 0
1611	003310	000000	.WORD 0
1612	003312	000000	.WORD 0
1613	003314	000000	TAB6: .WORD 0
1614	003316	000000	.WORD 0
1615	003320	000000	.WORD 0
1616	003322	000000	.WORD 0
1617	003324	047050	TAB6A: .WORD 47050
1618	003326	010000	.WORD 10000
1619	003330	000000	.WORD 0
1620	003332	000000	.WORD 0
1621	003334	000200	TAB7: .WORD 200
1622	003336	000000	.WORD 0
1623	003340	000000	.WORD 0
1624	003342	000000	.WORD 0
1625	003344	000200	TAB8: .WORD 200
1626	003346	000000	.WORD 0
1627	003350	000000	.WORD 0
1628	003352	000001	.WORD 1
1629	003354	000400 000000 000000	TAB9: .WORD 400,0,0,0
	003362	000000	
1630	003364	030000	TAB10: .WORD 30000
1631	003366	003000	.WORD 3000
1632	003370	000000	.WORD 0
1633	003372	000000	.WORD 0
1634	003374	016400	TAB11: .WORD 16400
1635	003376	000000	.WORD 0
1636	003400	000000	.WORD 0
1637	003402	000000	.WORD 0
1638	003404	030000 003000 000002	TAB11A: .WORD 30000,3000,2,0

GLOBAL DATA SECTION

1639	003412	000000					
	003414	016100	000000	000000	TAB12:	.WORD	16100,0,0,1
	003422	000001					
1640	003424	016200			TAB13:	.WORD	16200
1641	003426	000000				.WORD	0
1642	003430	000000				.WORD	0
1643	003432	000001				.WORD	1
1644	003434	030000	003000	000000	TAB13B:	.WORD	30000,3000,0,140000
	003442	140000					
1645	003444	030000			TAB14:	.WORD	30000
1646	003446	000000				.WORD	0
1647	003450	000000				.WORD	0
1648	003452	000000				.WORD	0
1649	003454	024700			TAB15:	.WORD	24700
1650	003456	000000				.WORD	0
1651	003460	000000				.WORD	0
1652	003462	000000				.WORD	0
1653	003464	025000			TAB16:	.WORD	25000
1654	003466	175363				.WORD	175363
1655	003470	123456				.WORD	123456
1656	003472	123456				.WORD	123456
1657	003474	030000			TAB17:	.WORD	30000
1658	003476	007020				.WORD	7020
1659	003500	000000	000000			.WORD	0,0
1660	003504	023456			TAB18:	.WORD	23456
1661	003506	000000				.WORD	0
1662	003510	000000				.WORD	0
1663	003512	000001				.WORD	1
1664	003514	100200	000000	000000	TAB21:	.WORD	100200,0,0,0
	003522	000000					
1665	003524	100400	000000	000000	TAB22:	.WORD	100400,0,0,0
	003532	000000					
1666	003534	000200	000000	000000	TAB23:	.WORD	200,0,0,1
	003542	000001					
1667	003544	062400	000000	000000	TAB24:	.WORD	62400,0,0,0
	003552	000000					
1668	003554	001100	000000	000000	TAB25:	.WORD	1100,0,0,0
	003562	000000					
1669	003564	100600	000000	000000	TAB26:	.WORD	100600,0,0,0
	003572	000000					
1670	003574	001000	000000	000000	TAB27:	.WORD	1000,0,0,0
	003602	000000					
1671	003604	000600	000000	000000	TAB28:	.WORD	600,0,0,0
	003612	000000					
1672	003614	010100	000000	000000	TAB29:	.WORD	10100,0,0,0
	003622	000000					
1673	003624	010100	000000	002000	TAB29A:	.WORD	10100,0,2000,0
	003632	000000					
1674							
1675	003634	000500	000000	000000	TAB30:	.WORD	500,0,0,0
	003642	000000					
1676	003644	100400	000000	000000	TAB31:	.WORD	100400,0,0,0
	003652	000000					
1677	003654	016000	000000	000000	TAB32:	.WORD	16000,0,0,0
	003662	000000					
1678	003664	011600	000000	000000	TAB33:	.WORD	11600,0,0,0
	003672	000000					

GLOBAL DATA SECTION

```

1679 003674 000640 0000C0 000000 TAB34: .WORD 640,0,0,0
      003702 000000
1680 003704 077600 000000 000000 TAB40: .WORD 77600,0,0,0
      003712 000000
1681 003714 100200 000000 000000 TAB41: .WORD 100200,0,0,1
      003722 000001
1682 003724 000340 000000 000000 TAB42: .WORD 340,0,0,0
      003732 000000
1683 003734 000077 177777 177777 TAB43: .WORD 77,177777,177777,177776
      003742 177776
1684 003744 000577 177777 177777 TAB45: .WORD 577,-1,-1,-1
      003752 177777
1685 003754 000577 177777 000000 TAB46: .WORD 577,-1,0,0
      003762 000000
1686 003764 173737 124242 052525 TAB47: .WORD 173737,124242,052525,12346
      003772 012346
1687 003774 000000 000000 052525 TAB47A: .WORD 0,0,052525,12346
      004002 012346
1688 004004 173737 124242 000000 TAB48: .WORD 173737,124242,0,0
      004012 000000
1689 004014 000600 000000 000000 TAB49: .WORD 600,0,0,0
      004022 000000

```

```

1690
1691 004024 000000 DCOUNT : .WORD 0
1692 004026 000000 EXPDAT : .WORD 0
1693 004030 000000 RECDAT : .WORD 0
1694 004032 000000 PWDSEQ : .WORD 0
1695 004034 000000 ADLSB : .WORD 0
1696 004036 000000 RITEDA : .WORD 0
1697 004040 000000 NEWDAT : .WORD 0
1698 004042 000000 CURDAT : .WORD 0
1699 004044 000000 FSTADD : .WORD 0
1700 004046 000000 LSTADD : .WORD 0
1701 004050 000000 CURADD : .WORD 0
1702 004052 000000 PARABT : .WORD 0 ;PARIY ABORT FLAG
1703
1704
1705 004054

```

```

START:
;; LCP/ORION ROUTINE TO SAVE EMTULATOR 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
        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:

```

GLOBAL DATA SECTION

1706 004152

```

*****
i$:
.SBTTL INITIALIZE THE COMMON TAGS
;;CLEAR THE COMMON TAGS ($CMTAG) AREA
004152 012706 001100 MOV    #CMTAG,R6    ;;FIRST LOCATION TO BE CLEARED
004156 005026          CLR    (R6)+        ;;CLEAR MEMORY LOCATION
004160 022706 001140 CMP    #SWR,R6     ;;DONE?
004164 001374          BNE    -6           ;;LOOP BACK IF NO
004165 012706 001100 MOV    #STACK,SP   ;;SETUP THE STACK POINTER
;;INITIALIZE A FEW VECTORS
004172 012737 030150 000020 MOV    #SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
004200 012737 000340 000022 MOV    #340,@#IOTVEC+2 ;;LEVEL 7
004206 012737 030452 000030 MOV    #ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
004214 012737 000340 000032 MOV    #340,@#EMTVEC+2 ;;LEVEL 7
004222 012737 033176 000034 MOV    #TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
004230 012737 000340 000036 MOV    #340,@#TRAPVEC+2 ;;LEVEL 7
004236 012737 033260 000024 MOV    #PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
004244 012737 000340 000026 MOV    #340,@#PWRVEC+2 ;;LEVEL 7
004252 013737 030062 030054 MOV    $ENDCT,$EOPCT ;;SETUP END-OF-PROGRAM COUNTER
004260 005037 001164          CLR    $TIMES      ;;INITIALIZE NUMBER OF ITERATIONS
004264 005037 001166          CLR    $ESCAPE     ;;CLEAR THE ESCAPE ON ERROR ADDRESS
004270 112737 000001 001115 MOVB   #1,$ERMAX   ;;ALLOW ONE ERROR PER TEST
004276 012737 004276 001106 MOV    #,$LPADR    ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
004304 012737 004304 001110 MOV    #,$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.
004312 013746 000004          MOV    @#ERRVEC,-(SP) ;;SAVE ERROR VECTOR
004316 012737 004352 000004 MOV    #30000$,@#ERRVEC ;;SET UP ERROR VECTOR
004324 012737 177570 001140 MOV    #DSWR,SWR   ;;SETUP FOR A HARDWARE SWICH REGISTER
004332 012737 177570 001142 MOV    #DDISP,DISPLAY ;;AND A HARDWARE DISPLAY REGISTER
004340 022777 177777 174572 CMP    #-1,@SWR   ;;TRY TO REFERENCE HARDWARE SWR
004346 001012          BNE    30002$     ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
;;AND THE HARDWARE SWR IS NOT = -1
004350 000403          BR     30001$     ;;BRANCH IF NO TIMEOUT
004352 012716 004360 30000$: MOV    #30001$, (SP) ;;SET UP FOR TRAP RETURN
004356 000002          RTI
004360 012737 000176 001140 30001$: MOV    #SWREG,SWR ;;POINT TO SOFTWARE SWR
004366 012737 000174 001142 MOV    #DISPREG,DISPLAY ;;
004374 012637 000004 30002$: MOV    (SP)+,@#ERRVEC ;;RESTORE ERROR VECTOR
004400 005037 001206          CLR    $PASS      ;;CLEAR PASS COUNT
004404 132737 000200 001221 BITB   #APTSIZE,$ENVM ;;TEST USER SIZE UNDER APT
004412 001403          BEQ    30003$     ;;YES,USE NON-APT SWITCH
004414 012737 001222 001140 MOV    #SWREG,SWR  ;;NO,USE APT SWITCH REGISTER
004422          30003$:
1707 004422 013737 004146 004150 MOV    UDFDLG,UQUIET ;;ABORT IN UFD ON ERROR
1708 004430 012737 026402 000004 MOV    #TOUT,@#ERRVEC ;;POINT TO TIMEOUT ROUTINE
1709 004436 012737 000340 000006 MOV    #340,@#ERRVEC+2 ;;AT PRIORITY 7
1710 004444 012737 025664 000114 MOV    #RAMPAR,@#114 ;;POINT PARITY ABORT
1711 004452 012737 000340 000116 MOV    #340,@#116   ;;AT PRIORITY7
1712 004460 012737 026564 000250 MOV    #MMUTRP,@#250 ;;POINT MMU TRAP VECTOR
1713 004466 012737 000340 000252 MOV    #340,@#252   ;;
1714 004474 005037 177766          CLR    @#177766    ;;CLEAR CPU ERROR REGISTER
1715 004500 032737 000100 000052 BIT    #BIT06,@#52 ;;IN UFD QUIET MODE ?
1716 004506 001164          BNE    LOOP       ;;IF SO, SKIP PRINTOUT
1717 004510 104401 016302          TYPE   ,SWTSEL   ;;
1718          .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
004514 005737 000042          TST    @#42       ;;ARE WE RUNNING UNDER XXDP/ACT?

```

GET VALUE FOR SOFTWARE SWITCH REGISTER

```

004520 001012          BNE      30004$      ;;BRANCH IF YES
004522 123727 001220 000001  CMPB     $ENV,#1    ;;ARE WE RUNNING UNDER APT?
004530 001406          BEQ      30004$      ;;BRANCH IF YES
004532 023727 001140 000176  CMP      SWR,#SWREG ;;SOFTWARE SWITCH REG SELECTED?
004540 001005          BNE      30005$      ;;BRANCH IF NO
004542 104406          GTSWR                    ;;GET SOFT SWR SETTINGS
004544 000403          BR                          30005$
004546 112737 000001 001134 30004$: MOVB     #1,$AUTOB  ;;SET AUTO-MODE INDICATOR
004554 30005$:

1719 004554 000240          NOP                          ; debug a' d
1720 004556 123727 001220 000001  CMPB     $ENV,#1    ; running under APT?
1721 004564 001020          BNE      20$             ; default no-APT initialization
1722 004566 013700 001224  MOV      $USWR,R0      ; work copy pass calculation
1723 004572 005700          TST      R0             ; f = 0 default value
1724 004574 001420          BEQ      25$             ; setup default value
1725
1726 004576 042700 000017  BIC      #17,R0        ; clear low order nibble
1727 004602 000241          CLC                          ; assure no unknowns
1728 004604 006000          ROR      R0             ; 4 rotates = divide
1729 004606 006000          ROR      R0             ; by 16 (=pass time)
1730 004610 006000          ROR      R0             ; this area subroutined
1731 004612 006000          ROR      R0             ; with general purpose
1732 004614 005700          TST      R0             ; divide, this test to
1733 004616 001413          BEQ      30$             ; determine skip altogether
1734
1735 004620 010037 003030  MOV      R0,CCHPAS     ;residue = no. of desired passes
1736 004624 000413          BR                          ; continue on
1737 004626 012737 177777 003030 20$: MOV      #-1,CCHPAS   ; largest number no apt mode??
1738 004634 000407          BR                          35$
1739 004636 012737 000001 003030 25$: MOV      #1,CCHPAS   ; normal default = 1
1740 004644 000403          BR                          35$
1741 004646 012737 000000 003030 30$: MOV      #0,CCHPAS   ; no cache tests included
1742 004654 000240          35$: nop                    ; debug aid
1743
1744          ; BIT      #BIT07,@#52  ;UFD MODE?
1745          ; BNE      LOOP          ;IF YES, BRANCH
1746
1747
1748          ; ::::::::::::::::::::: HALT ON ERROR JUMPER
1749 004656 032737 000010 177750  BIT      #BIT03,MAIREG
1750 004664 001437          BEQ      99$
1751 004666 104401 004674  TYPE     30007$      ;;TYPE ASCIZ STRING
004672 000433          BR                          30006$      ;;GET OVER THE ASCIZ
;;30007$: .ASCIZ <15><12>/TRAP ON HALT IS ENABLED, JUMPER IS NOT INSTALLED/<15><12>
30006$:
1752
1753 004762 000434          BR                          1000$
1754
1755          99$:
004764 104401 004772  TYPE     30009$      ;;TYPE ASCIZ STRING
004770 000431          BR                          30008$      ;;GET OVER THE ASCIZ
;;30009$: .ASCIZ <15><12>/TRAP ON HALT IS DISABLED, JUMPER IS INSTALLED/<15><12>
30008$:
1756
1757 005054 004737 025730 1000$: JSR      PC,Q22SIZ  ;SIZE FOR Q22BE
1758
1759

```

H4

GET VALUE FOR SOFTWARE SWITCH REGISTER

1760
1761 005060
1762
1763
1764

LOOP:

j

TEST - NATIVE REGISTER

1766
1767
1768
1769
1770
1771
1772
1773
1774
1775
1776
1777
1778
1779
1780
1781
1782
1783
1784
1785
1786
1787
1788

```

.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 TIMEOUT
; 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 LEDs)
; ENDTST
; TIMEOUT: Clean stack
; Error NATIVE register timed out
;-----
;*****
; NATIVE REGISTER TEST
;*****

```

1789 005060 000004
1790 005062 032777 010000 174050
1791 005070 001424
1791 005072 104401 005100
1791 005076 000421
1792 005142
1793 005142 013737 000004 003012
1794 005150 012737 005320 000004
1795 005156 012737 000340 000006
1796 005164 013700 177520
1797
1798
1799
1800
1801 005170 052737 160000 177520
1802 005176 013701 177520
1803 005202 020001
1804 005204 001402
1805 005206 104047
1806 005210 000445
1807
1808
1809
1810 005212 042737 000177 177520
1811 005220 052737 000177 177520
1812 005226 001401
1813 005230 104050
1814 005232 012701 000001
1815 005236 012702 000007
1816 005242
1817 005242 050137 177520
1818 005246 030137 177520

```

;*****
; NATIVE REGISTER TEST
;*****
TST1: SCOPE
; BIT #BIT12, @SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
; BEQ 1000$ ; THEN TYPE TEST TRACE
; TYPE .30011$ ; TYPE ASCIZ STRING
; BR 30010$ ; GET OVER THE ASCIZ
; .30011$: .ASCIZ <15><12>/TEST 1 - NATIVE REGISTER TEST/
; 30010$:
; 1000$:
MOV @#4, SAVTIM ; SAVE UNEXPECTED TIMEOUT TRAP
MOV +100$, @#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
; 20$: MOV #1, R1 ; START PATTERN IN FIRST BIT
; MOV #7, R2 ; SET LOOP COUNT TO 7
; 25$: ; FOR ALL BITS THE ARE INDICATOR DO
; BIS R1, NATREG ; SET THE BIT
; BIT R1, NATREG ; CHECK THAT IT GOT SET

```


TEST NATIVE REGISTER

```

1819 005252 001002      BNE      30$      ; : IF NOT THEN
1820 005254 104050      ERROR    +50      ; :   ERROR IN THE INDICATOR BITS
1821 005256 000422      BR       110$     ; :
1822 005260 006301      30$:  ASL      R1      ; :   SHIFT PATTERN TO NEXT BIT
1823 005262 077211      SOB      R2,+25$  ; :   ENDFOR
1824 005264 042737 000177 177520  BIC      #177,NATREG ; :   CLEAR THE INDICATORS
1825
1826      ; CHECK THE BOOT SWITCH TO BE READ ONLY
1827      ;
1828 005272 013700 177520      MOV      NATREG,R0 ; :   READ THE NATIVE REGISTER
1829 005276 052737 007400 177520  BIS      #7400,NATREG ; :   TRY TO WRITE TO THE BOOT SELECT SWITCH
1830 005304 013701 177520      MOV      NATREG,R1 ; :   READ IT AGAIN
1831 005310 020100      CMP      R1,R0    ; :   IF THE BITS CHANGED THEN
1832 005312 001404      BEQ      110$     ; :   ERROR IN THE NATIVE REGISTER
1833 005314 104051      ERROR    +51      ; :
1834 005316 000402      BR       110$     ; :
1835 005320 104052      100$: ERROR    +52      ; :   ERROR IN THE NATIVE REGISTER
1836 005322 022626      CMP      (SP)+,(SP)+ ; :   ENDTST
1837
1838 005324      110$:
1839 005324 013737 003012 000004  MOV      SAVTIM,#4 ; :   RESTORE UNEXPECTED TIMEOUT      ;MC
1840

```

TEST - 16 BIT ROM CHECKSUM TEST

```

1842 .SBTTL TEST - 16 BIT ROM CHECKSUM TEST
1843 ;ROM'S CHECKSUMS
1844 ;
1845 ;16 BIT ROM TEST
1846
1847 ;:*****
1848 005332 000004 TST2: SCOPE
1849 005334 032777 010000 173576 BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
1850 005342 001426 BEQ 1000$ ; THEN TYPE TEST TRACE
1850 005344 104401 005352 TYPE 30013$ ;;TYPE ASCIZ STRING
005350 000423 BR 30012$ ;;GET OVER THE ASCIZ
;;30013$: .ASCIZ <15><12>/TEST 2 - KDJ11-D ROM CHECKSUM TEST/
30012$:
1000$:
1851 005420 BR TST3 ;;GO TO THE NEXT TEST SKIP THIS TEST UNTIL WE GET FINAL ROMS
1852 005420 000437 JSR PC,SETMMU ; SET UP THE MMU REGISTERS
1853 005422 004737 027060 MOV #17600,@#KIPAR2 ; POINT PAGE 2 TO SELF TEST
1854 005426 012737 017600 172344 MOV #40000,R1 ; POINT R1 TO PAGE 2
1855 005434 012701 040000 BIS #BIT7,@#NATREG ; ENABLE THE SELF TEST ADDRESS
1856 005440 052737 000200 177520 10$: MOV (R1)+,R2
1857 005446 012102 MOV (R1),R3
1858 005450 011103 CLR R0
1859 005452 005000 MOV #40000,R1
1860 005454 012701 040000 20$: ADD (R1)+,R0
1861 005460 062100 DEC R2
1862 005462 005302 BEQ 30$
1863 005464 001411 CMP R1,#60000
1864 005466 020127 060000 BLO 20$
1865 005472 103772 ADD #200,@#KIPAR2
1866 005474 062737 000200 172344 MOV #40000,R1
1867 005502 012701 040000 BR 20$
1868 005506 000764 30$: NEG R0
1869 005510 005400 CMP R0,R3
1870 005512 020003 BEQ 40$
1871 005514 001401 ERROR +4
1872 005516 104004 40$:
1873 005520
1874
1875
1876

```

TEST - KDJ11-DA DATA PATHS

1878
 1879
 1880
 1881
 1882
 1883
 1884
 1885
 1886
 1887
 1888
 1889
 1890
 1891
 1892
 1893
 1894
 1895
 1896
 1897
 1898
 1899
 1900
 1901
 1902
 1903
 1904
 1905
 1906
 1907

```
.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
:
: BGNTST
: Set Timeout trap to TIMOUT
: Set t mecut priority to 7
: Read location 0
: FOR pattern := first to last
:   Write pattern
:   Read pattern
:   IF Pattern read <> Pattern Written THEN
:     ERROR
:   ENDFOR
: ENDTST
```

1908 005520 0C0004
 1909 005522 032777 010000 173410
 1910 005530 001425
 005532 104401 005540
 005536 000422
 1911 005604
 1912 005604
 1913 005604 013737 000004 003012
 1914 005612 012737 005710 000004
 1915 005620 012737 000340 000006
 1916 005626 005737 000000
 1917 005632 010701
 1918 005634 062701 000070
 1919 005640 012702 000007
 1920 005644
 1921 005644 011137 000000
 1922 005650 023721 000000
 1923 005654 001410
 1924 005656 013737 000000 001126
 1925 005664 016137 177776 001124
 1926 005672 104044
 1927 005674 000401
 1928 005676 077216
 1929 005700 013737 003012 000004
 1930 005706 000415

```

:*****
: DATA PATH TESTS
:*****
TST3: SCOPE
: BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
: BEQ 1000$ ; THEN TYPE TEST TRACE
: TYPE 30015$ ;;TYPE ASCIZ STRING
: BR 30014$ ;;GET OVER THE ASCIZ
: 30015$: .ASCIZ <15><12>/TEST 3 - KDJ11-D DATA PATH TEST/
: 30014$:
: 1000$:
: MOV @#4,SAVTIM ; SAVE UNEXPECTED TIMEOUT TRAP ;MC
: MOV #100$,@#4 ; SET UP THE TIMEOUT TRAP
: MOV #340,@#6
: TST @#0 ; READ LOCATION 0
: MOV PC,R1 ; GET A POINTER TO THE PATTERNS
: ADD #PATT-.,R1
: MOV #7,R2 ; SET THE LOOP COUNT TO 7
: 5$: ; FOR ALL OF THE PATTERNS
: MOV (R1),@#0 ; WRITE THE PATTERN TO 0
: CMP @#0,(R1)+ ; IF THE PATTERN IS NOT READ BACK
: BEQ 10$ ; THEN
: MOV @#0,$BDDAT ; GET READ DATA
: MOV -2(R1),$GDDAT ; GET EXPECTED DATA
: ERROR +4 ; ERROR IN THE DATA PATHS
: BR 15$ ; END IF
: 10$: SOB R2,5$ ; ENDFOR
: 15$: MOV SAVTIM,@#4 ; END TEST
: BR TST4 ; RESTORE UNEXPECTED TIMEOUT TRAP
: ;GO TO THE NEXT TEST
```

TEST - KDJ11-DA DATA PATHS

```

1931
1932
1933
1934      ; TIMEOUT ROUTINE
1935      ;
1936
1937 005710 104045      100$:  ERROR  +45
1938 005712 022626      CMP    (SP)+,(SP)+      ; CLEAN STACK
1939 005714 013737 003012 000004  MOV   SAVTIM,@#4      ; RESTORE UNEXPECTED TIMEOUT TRAP
1940 005722 000407      BR    TST4      ;;GO TO THE NEXT TEST
1941
1942 005724 000000      PATT: .WORD 0
1943 005726 177777      .WORD 177777
1944 005730 177400      .WORD 177400
1945 005732 170360      .WORD 170360
1946 005734 007417      .WORD 007417
1947 005736 052525      .WORD 052525
1948 005740 125252      .WORD 125252
1949

```

TEST MEMORY ACCESSABILITY

1951
1952
1953
1954
1955
1956
1957
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1968
1969
1970
1971
1972
1973

```

.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

```

```

1974 005742 000004
1974 005744 032777 010000 173166
1975 005752 001426
1976 005754 104401 005762
      005760 000423
      006030
1977 006030
1978 006030 013737 000004 003012
1979 006036 012737 006134 000004
1980 006044 012737 000340 000006
1981 006052 004737 027060
1982 006056 012701 040000
1983 006062 012737 001600 172344
1984 006070
1985 006070 142721 000377
1986 006074 020127 060000
1987 006100 103773
1988 006102 062737 000200 172344
1989 006110 012701 040000
1990 006114 023727 172344 020000
1991 006122 001362
1992 006124 013737 003012 000004
1993 006132 000411
1994 006134
1995
1996
1997
1998
1999 006134 005301
2000 006136 104046
2001 006140 022626
2002 006142 005037 177572
2003 006146 013737 003012 000004

```

```

:*****
TST4: SCOPE
      BIT      #BIT12,@SWR      ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
      BEQ      1C00$            ; THEN TYPE TEST TRACE
      TYPE     30017$           ;;TYPE ASCIZ STRING
      BR       30016$           ;;GET OVER THE ASCIZ
:30017$:
:30016$:
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$:
      BICB     #377,(R1)+       ; FOR 160000 TO 200000
      CMP      R1,#60000        ; CLEAR OUT A BYTE
      BLO     1$                ; IF WE HAVE PASSED THE PAGE BOUNDARY
      ADD      #200,@#KIPAR2    ; THEN
      MOV      #40000,R1        ; POINT KPAR2 TO A NEW PAGE
      CMP      @#KIPAR2,#20000  ; SET THE VIRTUAL ADDRESS TO PAGE 2
      BNE     1$                ; ENDF
10$:
      MOV      SAVTIM,@#4
      BR       TST5             ;;GO TO THE NEXT TEST
20$:
:
: TIMEOUT ROUTINE
:
100$:
      DEC      R1                ;
      ERROR   +46                ; REPORT ERROR
      CMP      (SP)+,(SP)+      ; CLEAN STACK
      CLR     @#SRO              ;
      MOV      SAVTIM,@#4       ; RESTORE TIMEOUT

```

85

TEST - MEMORY ACCESSABILITY

2004 006154 000400
2005

BR

TST5

::GO TO THE NEXT TEST

TEST - MEMORY ERROR REGISTER

```

2007 .SBTTL TEST - MEMORY ERROR REGISTER
2008 *****
2009 ;MEMORY ERROR REGISTER
2010 ;
2011 ; This test will check the MEMORY ERROR REGISTER on the KDJ11-D
2012 ; Board.
2013 ;
2014 ; BGNTST
2015 ; Setup timeout VECTOR PC to TIMOUT
2016 ; Setup timeout VECTOR Priority to 7
2017 ; Setup Parity abort VECTOR to PARINT
2018 ; Setup Parity abort VECTOR Priority to 7
2019 ; Do a bus reset
2020 ; Read the Memory Error Register (1777?100)
2021 ; Make sure that the register bits are in the right state
2022 ; Check all R/W bits
2023 ; ENDTST
2024 -----
2025 *****
2026 *****
2027 006156 000004 010000 172752 1001$: SCOPE
2028 006160 032777 010000 172752 BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
2029 006166 001427 010000 172752 BEQ 1000$ ; THEN TYPE TEST TRACE
2030 006170 104401 006176 000000 TYPE 30019$ ; TYPE ASCIZ STRING
2031 006174 000424 006176 000000 BR 30018$ ; GET OVER THE ASCIZ
2032 ; 30019$: .ASCIZ <15><12>/TEST 5 - MEMORY ERROR REGISTER TEST/
2033 ; 30018$:
2034 ; 1000$:
2035 clr parabt ;init parity abort.....mc
2036 mov #340,@#177776 ;set the psw to 7.....mc
2037 CMPB #APIENV,$ENV ;ARE WE IN APT MODE?
2038 BNE 1001$ ;IF NOT: DO THIS TEST
2039 TST $PASS ;FIRST PASS?
2040 BEQ 1001$ ; YES THEN PROCEED
2041 JMP PAREND ; NO THEN GO TO THE NEXT TEST
2042 1001$: MOV @#4,SAVTIM ; SAVE UNEXPECTED TIMEOUT TRAP
2043 MOV #100$,@#4 ; SET UP THE TIMEOUT TRAP
2044 MOV #340,@#6 ;
2045 MOV @#114,-(SP) ; SAVE VECTOR
2046 MOV @#116,-(SP) ; SAVE PRIORITY
2047 MOV #PARINT,@#114 ;
2048 MOV #340,@#116 ;
2049 CLR @#MER ; READ THE MER
2050 MOV @#MER,R0 ;
2051 ; CHECK OUT THE MER 'ALWAYS READ AS 0' BITS
2052 ;
2053 BIS #30032,MER ; TRY TO WRITE THE STUCK AT 0 BITS
2054 MOV MER,R1 ; READ THE MER
2055 CMP R0,R1 ; IF THE STUCK AT 0 BITS CHANGED
2056 BEQ 10$ ; : THEN
2057 ERROR +55 ; : ERROR IN THE MER
2058 ; ENDF
2059 ; CHECK THAT BITS 0,2,14,15 ARE CLEARED ON RESET
2060 10$: BIS #140005,MER ; SET ALL THE R/W BITS ON THE BOARD

```

TEST MEMORY ERROR REGISTER

```

2060 006406 013700 172100      MOV      MER,R0      ; READ THE MER
2061 006412 042700 037772      BIC      #37772,R0   ; MASK OUT ALL THE UNWANTED BITS
2062 006416 020027 140005      CMP      R0,#140005 ; IF THE R/W BITS DID NOT GET SET THEN
2063 006422 001410                BEQ      15$         ; ERROR IN THE MER
2064 006424 005037 172100      CLR      MER
2065 006430 010037 001126      MOV      R0,$BDDAT  ; GET RECIEVED DATA
2066 006434 012737 140005 001124  MOV      #140005,$GDDAT ; GET EXPECTED DATA
2067 006442 104056                ERROR    +56         ; REPORT ERROR
2068 006444 000005                RESET
2069 006446 032737 140005 172100 15$: BIT      #140005,MER ; DO RESET
2070 006454 001401                BEQ      20$         ; MAKE SURE THE APPROPRIATE BITS GET CLEARED
2071 006456 104057                ERROR    +57         ; IF NOT THEN ERROR
2072
2073 ; CHECK OUT THE WRITE WRONG PARITY AND PARITY ERROR ENABLE BITS
2074
2075 006460                20$:
2076 006460 004737 027060      JSR      PC,SETMMU  ;
2077 006464 012737 017600 172344  MOV      #17500,@#KIPAR2 ; SET UP KPAR2
2078 006472 012701 057776      MOV      #57776,R1  ; POINT TO LAST VIRTUAL ADDRESS IN PAGE 2
2079 006476 042737 000001 172100  BIC      #BIT0,@#MER ; CLEAR PARITY ERROR ENABLE
2080 006504 052737 000004 172100  BIS      #BIT2,@#MER ; SET WRITE WRONG PARITY
2081
2082 ; WRITE WRONG PARITY TO LOCATION 1777776
2083
2084 006512 005011                CLR      (R1)        ; WRITE WRONG PARITY TO THAT LOCATION
2085 006514 042737 000004 172100  BIC      #BIT2,@#MER ; CLEAR WRITE WRONG PARITY
2086
2087 ; MAKE SURE NO PARITY ABORT OCCURS WHEN PARITY ERRORS ARE DISABLED
2088
2089 006522 005711                TST      (R1)        ; READ THE ADDRESS BACK
2090 006524 005737 003050      TST      MERTAG
2091 006530 000240                NOP
2092
2093 ; SHOULD
2094 ; NOT GET
2095 ; A
2096 ; PARITY ABORT
2097 ;
2098 ;
2099 ;
2100 006532 005737 004052      TST      PARABT     ; IF WE GOT A PARITY ABORT
2101 006536 001405                BEQ      25$         ; THEN
2102 006540 005037 004052      CLR      PARABT     ; CLEAR THE ABORT FLAG
2103 006544 005037 172100      CLR      @#MER      ; CLEAR MEMORY ERROR REGISTER ;MC
2104 006550 104062                ERROR    +62         ; FLAG THE ERROR
2105
2106 ; ENDIF
2107 ; MAKE SURE A PARITY ABORT OCCURS WHEN PARITY ERRORS ARE ENABLED
2108
2109 006552 005037 172100 25$: CLR      @#MER      ; CLEAR OUT THE MER
2110 006556 052737 000001 172100  BIS      #BIT0,@#MER ; NOW ENABLE PARITY ABORTS
2111 006564 005711                TST      (R1)        ; THIS HAS TO BE ONE WORD INTRUCTION
2112
2113 ; READ THE ADDRESS BACK
2114 006566 005737 003050      TST      MERTAG
2115
2116 ; SHOULD
                ; GET

```


TEST - MEMORY ERROR REGISTER

```

2117
2118
2119
2120
2121
2122
2123
2124 006572 042737 000001 172100 BIC #BIT0,@#MER
2125 006600 022737 000001 004052 CMP #1,PARABT
2126 006606 001403 BEQ 30$
2127 006610 005037 004052 CLR PARABT
2128 006614 104063 ERROR +63
2129
2130 ; CHECK OUT THE ADDRESS AND EXTENDED MER READ ENABLED
2131
2132 006616 005037 004052 30$: CLR PARABT
2133 006622 042737 100000 172100 BIC #BIT15,@#MER
2134 006630 013700 172100 MOV @#MER,R0
2135 006634 042700 170037 BIC #170037,R0
2136 006640 020027 007740 CMP R0,#7740
2137 006644 001406 BEQ 35$
2138 006646 010037 001126 MOV R0,$BDDAT
2139 006652 012737 007740 001124 MOV #7740,$GDDAT
2140 006660 104064 ERROR +64
2141 006662 052737 040000 172100 35$: BIS #BIT14,@#MER
2142 006670 013702 172100 MOV @#MER,R2
2143 006674 042702 177037 BIC #177037,R2
2144 006700 020227 000040 CMP R2,#40
2145 006704 001406 BEQ 40$
2146 006706 010237 001126 MOV R2,$BDDAT
2147 006712 012737 000040 001124 MOV #40,$GDDAT
2148 006720 104065 ERROR +65
2149 006722 005011 40$: CLR (R1)
2150 006724 005037 177572 CLR @#SRO
2151 006730 005037 172100 CLR @#MER
2152 006734 000405 BR 101$
2153
2154 ; TIMEOUT ROUTINE
2155
2156
2157 006736 104060 100$: ERROR +60
2158 006740 022626 CMP (SP)+,(SP)+
2159 006742 005011 CLR (R1)
2160 006744 005037 177572 CLR @#SRO
2161
2162 006750 101$: ;MC
2163 006750 012637 000116 MOV (SP)+,@#116
2164 006754 012637 000114 MOV (SP)+,@#114
2165 006760 013737 003012 000004 MOV SAVTIM,@#4
2166 006766 000403 BR TST6 ;:GO TO THE NEXT TEST
2167
2168 ; PARITY ABORT ROUTINE
2169
2170
2171 006770 PARINT:
2172 006770 005237 004052 INC PARABT
2173 006774 000002 RTI

```

```

: A
: PARITY ABORT
: SOON !!!
:
:
:
: DISABLE PARITY ABORTS
: IF WE DIDN'T GET A PARITY ABORT
: THEN
: CLEAR PARITY ABORT
: ERROR DID NOT GET A PARITY ABORT
:
:
:
: CLEAR THE ABORT FLAG
: CLEAR THE PARITY ERROR FLAG
: GET THE MEMORY ERROR REGISTER
: CLEAR THE UNNEEDED BITS
: IF NOT CORRECT BITS SET THEN
: ERROR IN ADDRESS 11-17
: GET RECEIVED DATA
: GET EXPECTED DATA
:
: SET ENABLE READ EXTENDED BIT
: READ THE MER AGAIN
: CLEAR THE UNNEEDED BITS
: IF NOT CORRECT BITS SET THEN
: ERROR IN ADDR 18-21 OR READ EXT BIT
: GET RECEIVED DATA
: GET EXPECTED DATA
:
: ENDF
:
: DISABLE MMU
:
: ENDTST
:
:
: REPORT ERROR
: CLEAN STACK
:
: DISABLE MMU
:
:
:
: RESTORE PRIORITY
: RESTORE VESTOR
: RESTORE UNEXPECTED TIMEOUT
:
:
: FLAG THAT WE CAME HERE
: RETURN

```

F5

TEST - MEMORY ERROR REGISTER

2174 006776

PAREND:

TEST - MEMORY ERROR REGISTER

2176
 2177
 2178
 2179
 2180
 2181
 2182
 2183
 2184
 2185
 2186
 2187
 2188
 2189
 2190
 2191
 2192
 2193
 2194
 2195
 2196
 2197
 2198
 2199
 2200
 2201
 2202
 2203
 2204
 2205
 2206
 2207
 2208
 2209
 2210
 2211
 2212
 2213
 2214
 2215
 2216
 2217
 2218
 2219
 2220
 2221

```
.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 's checked to be set to 0
; 2. IF NOT 0 error
; 3. The locat'on is complemented
; 4. IF contents NOT -1 error
; 5. This is repeated for all addresses
; 6. Steps 1-5 are done from location 1777777 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 1777776 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 := 1777776 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
```

2222 006776 000004
 2223 007000 032777 010000 172132
 2224 007006 001433
 007010 104401 007016
 007014 000430
 2225 007076
 2226 007076 005037 177572
 2227 007102 013737 000004 003012
 2228 007110 012737 007464 000004

```
*****
TST6: SCOPE
      BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
      BEQ 1000$ ; THEN TYPE TEST TRACE
      TYPE 30021$ ;;TYPE ASCIZ STRING
      BR 30020$ ;;GET OVER THE ASCIZ
      .ASCIZ <15><12>/TEST 6 - DATA SHORTS AND STUCK AT BITS TEST/
      30021$:
      30020$:
      1000$:
      CLR @#SRO ;
      MOV @#4,SAVTIM ;STORE UNEXPECTED TIMEOUT
      MOV #100$,@#4 ; SET UP THE TIMEOUT TRAP
```

TEST - DATA SHORTS AND STUCK AT BITS

2229	007116	012737	000340	000006	MOV	#340,@#6	:	
2230	007124	004737	027060		JSR	PC,SETMMU	:	GO SET UP THE MMU REGISTERS
2231	007130	012701	040000		MOV	#40000,R1	:	WE WANT TO MAP THRU PAGE 2
2232	007134	012737	001600	172344	MOV	#1600,@#KIPAR2	:	SET UP KPAR 2
2233	007142	013746	000114		MOV	@#114,-(SP)	:	SAVE VECTOR ;MC
2234	007146	013746	000116		MOV	@#116,-(SP)	:	SAVE PRIORITY ;MC
2235	007152	012737	007452	000114	MOV	#50,@#114	:	set up trap
2236	007160	012737	000340	000116	MOV	#340,@#116	:	" " "
2237	007166	052737	000001	172100	BIS	#BIT0,@#MER	:	ENABLE PARITY ABORTS ;mc
2238							:	
2239	007174						:	FOR ADDRESS 160000 TO 200000 DO
2240	007174	011137	001126		MOV	(R1), \$BDDAT	:	: READ MEMORY
2241	007200	001406			BEQ	15\$:	: IF IT IS NOT ZERO THEN
2242	007202	005037	001124		CLR	\$GDDAT	:	: GET THE EXPECTED DATA
2243	007206	010137	001122		MOV	R1, \$BDADR	:	: GET THE VIRTUAL ADDRESS
2244	007212	104061			ERROR	+61	:	: ERROR IN THE MEMORY
2245	007214	000532			BR	101\$:	: RESTORE AND TO THE NEXT TEST
2246							:	
2247	007216	005111					:	: COMPLEMENT MEMORY
2248	007220	011137	001126		MOV	(R1), \$BDDAT	:	: SAVE THE RECIEVED DATA
2249	007224	022127	177777		CMP	(R1)+, #177777	:	: IF ITS NOT = -1 THEN
2250	007230	001412			BEQ	20\$:	: ERROR IN THE MEMORY
2251	007232	012737	177777	001124	MOV	#-1, \$GDDAT	:	: GET EXPECTED DATA
2252	007240	010137	001122		MOV	R1, \$BDADR	:	: GET THE VIRTUAL ADDRESS
2253	007244	162737	000002	001122	SUB	#2, \$BDADR	:	
2254	007252	104061			ERROR	+61	:	
2255	007254	000512			BR	101\$:	: RESTORE AND TO THE NEXT TEST
2256							:	
2257	007256	020127	060000				:	: IF WE HAVE PASSED THE PAGE BOUNDARY
2258	007262	103744			BLO	10\$:	: THEN
2259	007264	062737	000200	172344	ADD	#200,@#KIPAR2	:	: POINT KPAR2 TO A NEW PAGE
2260	007272	012701	040000		MOV	#40000,R1	:	: SET THE VIRTUAL ADDRESS TO PAGE 2
2261	007276	023727	172344	020000	CMP	@#KIPAR2, #20000	:	: ENDIF
2262	007304	001401			BEQ	25\$:	: ENDFOR
2263	007306	000732			BR	10\$:	
2264	007310						:	
MEMORY	007310	012705	000200		MOV	#200,R5	:	THIS LOOP IS IN HERE TO TEST THE DATA RETENTION OF
2266	007314	012704	177777		MOV	#177777,R4	:	
2267	007320	077401			SQB	R4, 27\$:	
2268	007322	077504			SQB	R5, 26\$:	
2269	007324	012701	060000		MOV	#60000,R1	:	GET A POINTER TO THE TOP OF A PAGE
2270	007330	162737	000200	172344	SUB	#200,@#KIPAR2	:	SET KIPAR TO POINT TO THE TOP PAGE
2271	007336						:	FOR LAST ADDRESS TO FIRST BY 2
2272	007336	014137	001126		MOV	-(R1), \$BDDAT	:	: SAVE THE DATA
2273	007342	023727	001126	177777	CMP	\$BDDAT, #177777	:	: READ BACK MEMORY MAKE SURE IT IS -1
2274	007350	001407			BEQ	35\$:	: IF NOT = -1 THEN
2275	007352	012737	177777	001124	MOV	#177777, \$GDDAT	:	: GET THE EXPECTED DATA
2276	007360	010137	001122		MOV	R1, \$BDADR	:	: GET THE VIRTUAL ADDRESS
2277	007364	104061			ERROR	+61	:	: ERROR IN MEMORY
2278	007366	000445			BR	101\$:	: RESTORE AND TO THE NEXT TEST
2279							:	
2280	007370	005111					:	: COMPLEMENT MEMORY
2281	007372	011137	001126		MOV	(R1), \$BDDAT	:	: IF NOT = 0 THEN
2282	007376	001406			BEQ	40\$:	: ERROR IN MEMORY
2283	007400	005037	001124		CLR	\$GDDAT	:	: GET EXPECTED DATA
2284	007404	010137	001122		MOV	R1, \$BDADR	:	: GET THE VIRTUAL ADDRESS
2285	007410	104061			ERROR	+61	:	

TEST DATA SHORTS AND STUCK AT BJTS

```

2286 007412 000433          BR      101$          ; RESTORE AND TO THE NEXT TEST
2287
2288 007414 020127 040000    40$:  CMP      R1,#40000          ; : IF VIRTUAL ADDRESS ABOUT TO ENTER NEXT PAGE
2289 007420 101346          BHI      30$          ; : : THEN
2290 007422 162737 000200 172344    SUB      @#KIPAR2      ; : : ADJUST THE PAR
2291 007430 012701 060000          MOV      #60000,R1     ; : : RESET THE VIRTUAL ADDRESS TO TOP OF PAGE
2292 007434 023727 172344 001400    CMP      @#KIPAR2,#1400 ; : : ENDFIF
2293 007442 001335          BNE      30$          ; : ENDFOR
2294 007444 005037 172100          CLR      MER          ; : CLEAR THE MER
2295 007450 000414          BR      101$          ; : RESTORE AND TO THE NEXT TEST
2296
2297          ; PARITY ABORT ROUTINE
2298          ;
2299
2300 007452 005037 172100    50$:  CLR      MER          ; INIT THE MER
2301 007456 104072          ERROR   +72          ; REPORT ERROR
2302 007460 022626          CMP      (SP)+,(SP)+  ; CLEAN STACK
2303
2304          ;VECTOR SAVE AND RESTORE
2305          ;UNEXPECTED PARITY ABORT ERROR ROUTINE
2306 007462 000407          BR      101$
2307
2308          ;
2309          ; TIMEOUT ROUTINE
2310          ;
2311
2312 007464 005301    100$:  DEC      R1          ;
2313 007466 104046          ERROR   +46          ; REPORT ERROR
2314 007470 022626          CMP      (SP)+,(SP)+  ; CLEAN STACK
2315 007472 005037 172100          CLR      MER          ;
2316 007476 005037 177572          CLR      @#SRO       ;
2317
2318 007502 012637 000116    101$:  MOV      (SP)+,@#116   ; RESTORE PRIORITY
2319 007506 012637 000114          MOV      (SP)+,@#114   ; RESTORE VECTOR
2320 007512 013737 003012 000004    MOV      SAVTIM,@#4    ; RESTORE UNEXPECTED TIMEOUT
2321 007520 000400          BR      TST7          ;:GO TO THE NEXT TEST
2322
2323

```

TEST QUICK VERIFY DATA AND ADDRESSING TEST

```

2325 .SBTTL TEST - QUICK VERIFY DATA AND ADDRESSING TEST
2326 :*****
2327 :UNIQUE ADDRESSING TEST
2328 :
2329 : This test will check the data and addressing of the memories.
2330 : It uses the KNAIZUK HARTMANN QUICK VERIFY TRAM TEST ALGORITHM.
2331 : This algorithm will test memory for all stuck at faults in the
2332 : data and addressing
2333 :
2334 : Memory is split up into sections of 3.
2335 : I.E. 0,1,2 - 3,4,5 6,7,10 ...
2336 :
2337 : Testing works as follows.
2338 :
2339 : 1. Write a 0 into the 2nd and 3rd address of all groups
2340 : write a -1 into the 1st address of all groups
2341 : 2. make sure that the 2nd address of all groups contain 0
2342 : 3. Write a -1 into the 2nd address of all groups
2343 : 4. Make sure that the 3rd address of all groups contain 0
2344 : 5. Make sure that the 1st and 2nd address of all groups
2345 : contain -1
2346 : 6. Write a 0 into the 1st address of all groups
2347 : 7. Make sure that the 1st address of all groups contain 0
2348 : 8. Write a -1 into the 3rd address of all groups
2349 : 9. Make sure that the 3rd address of all groups contain -1
2350 :
2351 :-----
2352 :*****
2353 :
2354 :

```

```

2355 007522 000004 010000 171406
2356 007524 032777 007542
2357 007532 001434
007534 104401
007540 000431

2358 007624
2359 007624 004737 027060
2360 007624 013737 000004 003012
2361 007630 013737 010020 000004
2362 007636 004737 027174
2363 007644 012704 000001
2364 007650 005005
2365 007654 005003
2366 007656 004737 027360
2367 007660 012704 000001
2368 007664 012705 177777
2369 007670 012703 000001
2370 007674 004737 027360
2371 007700 012704 000002
2372 007704 005005
2373 007710 005003
2374 007712 004737 027360
2375 007714 004737 027532
2376 007720 005004
2377 007724 005005

:*****
:TEST: 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$: .ASCIZ <15><12>/TEST 7 - QUICK VERIFY DATA AND ADDRESSING TEST/
: 30022$:
: 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

```

TEST - QUICK VERIFY DATA AND ADDRESSING TEST

```

2378 007730 012703 000001      MOV    #1,R3      ; SET OPERATION TO WRITE
2379 007734 004737 027360      CALL   RWTMEM    ; GO WRITE THE MEMORY
2380 007740 005004              CLR    R4        ; SET ADDRESS INDEX TO 0
2381 007742 005005              CLR    R5        ; SET EXPECTED PATTERN TO 0
2382 007744 005003              CLR    R3        ; SET OPERATION TO READ
2383 007746 004737 027360      CALL   RWTMEM    ; GO READ AND CHECK MEMORY
2384 007752 012704 000002      MOV    #2,R4     ; SET ADDRESS INDEX TO 2
2385 007756 012705 177777      MOV    #-1,R5    ; SET PATTERN TO 1
2386 007762 012703 000001      MOV    #1,R3     ; SET OPERATION TO WRITE
2387 007766 004737 027360      CALL   RWTMEM    ; GO WRITE THE MEMORY
2388 007772 012704 000002      MOV    #2,R4     ; SET ADDRESS INDEX TO 2
2389 007776 012705 177777      MOV    #-1,R5    ; SET EXPECTED PATTERN TO -1
2390 010002 005003              CLR    R3        ; SET OPERATION TO READ
2391 010004 004737 027360      CALL   RWTMEM    ; GO READ AND CHECK MEMORY
2392 010010 013737 003012 000004  MOV    SAVTIM,@#4 ; RESTORE UNEXPECTED TIMEOUT
2393 010016 000411              BR     TST10     ;;GO TO THE NEXT TEST
2394
2395
2396      ; TIMEOUT ROUTINE
2397      ;
2398
2399 010020 005301      100$: DEC    R1      ;
2400 010022 104046      ERROR  +46        ; REPORT ERROR
2401 010024 022626      CMP    (SP)+,(SP)+ ; CLEAN STACK
2402 010026 005037 177572      CLR    @#SRO     ;
2403 010032 013737 003012 000004  MOV    SAVTIM,@#4 ; RESTORE UNEXPECTED TIMEOUT
2404 010040 001400      BEQ   TST10     ;;GO TO THE NEXT TEST
2405

```

TEST QUICK VERIFY DATA AND ADDRESSING TEST

2407
2408
2409
2410
2411
2412
2413
2414
2415
2416
2417
2418
2419
2420
2421
2422
2423
2424
2425
2426
2427
2428
2429
2430
2431
2432
2433
2434
2435
2436
2437
2438
2439
2440
2441

```

.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

```

```

010042 000004
2442 010044 122737 000001 001220
2443 010052 001005
2444 010054 005737 001206
2445 010060 001402
2446 010062 000137 011036
2447 010066 032777 000200 171044 2000$: BIT #BIT7,@SWR ; IF BIT7 SET IN SWR THEN SKIP THIS TEST
2448 010074 001402
2449 010076 000137 011036
2450 010102 032777 010000 171030 100$: BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
2451 010110 001426
2452 010112 104401 010120
010116 000423
;:30025$:
;:30024$:
1000$:
2453 010166
2454 010166 005037 172100
2455 010172 005037 004052
2456 010176 013746 000114
2457 010202 013746 000116
2458 010206 012737 006770 000114
2459 010214 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
; IF BIT7 SET IN SWR THEN SKIP THIS TEST
;
; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
; THEN TYPE TEST TRACE
;:TYPE ASCIZ STRING
;:GET OVER THE ASCIZ
;:ASCIZ <15><12>/TEST 10 - CHECK PARITY DETECT TEST/
;
; INIT THE MEMORY ERROR REG.
;
; SAVE VECTORE
; SAVE PRIORITY
;
;

```


TEST CHECK PARITY DETECT LOGIC AND RAMS

```

2460
2461      ; CLEAR ALL OF MEMORY WITH WRITE WRONG PARITY SET
2462
2463 010222 004737 027060      JSR    PC,SETMMU      ; GO SET UP THE MMU REGISTERS
2464 010226 105037 003010      CLRB   PARPAT        ; INITIALIZE FIRST PATTERN TO ZERO
2465 010232 012701 040000      MOV    #40000,R1     ; WE WANT TO MAP THRU PAGE 2
2466 010236 012737 001600 172344  MOV    #1600,@#KIPAR2 ; SET UP KPAR 2
2467 010244 012703 000002      MOV    #2,R3        ; LOOP FOR ODD PARITY AND EVEN PARITY
2468
2469
2470 010250      1$:
2471
2472 010250 012737 000004 172100  MOV    #4,@#MER      ; BGND0
2473 010256 113721 003010      MOVB   PARPAT,(R1)+  ; : ENABLE WRITE WRONG PARITY ;MC
2474 010262 005037 172100      CLR    @#MER         ; : CLEAR OUT A BYTE
2475
2476 010266 020127 060000      CMP    R1,#60000    ; : DISABLE WRITE WRONG PARITY ;MC
2477 010272 103766      BLO    1$           ; : FOR 160000 TO 2000000
2478 010274 062737 000200 172344  ADD    #200,@#KIPAR2 ; : IF WE HAVE PASSED THE PAGE BOUNDARY
2479 010302 012701 040000      MOV    #40000,R1   ; : : THEN
2480 010306 023727 172344 020000  CMP    @#KIPAR2,#20000 ; : : POINT KPAR2 TO A NEW PAGE
2481 010314 001355      BNE    1$          ; : : SET THE VIRTUAL ADDRESS TO PAGE 2
2482
2483      ; LET'S NOW MAKE SURE THAT THE ABORTS OCCUR
2484
2485 010316      20$:
2486 010316 012737 001600 172344  MOV    #1600,@#KIPAR2 ; : SET UP PAR 2
2487 010324 012701 040000      MOV    #40000,R1   ; : SET VIRTUAL ADDRESS TO PAGE #2
2488 010330      22$:
2489 010330 042737 000004 172100  BIC    #BIT2,@#MER  ; : BGND0
2490 010336 052737 000001 172100  BIS    #BIT0,@#MER  ; : : DISABLE WRITE WRONG PARITY
2491 010344 105711      TSTB   (R1)         ; : : ENABLE PARITY ERRORS
2492 010346 005737 003050      TST    MERTAG       ; : : READ THE BYTE
2493 010352 000240      NOP
2494
2495
2496
2497
2498
2499
2500
2501
2502 010354 023727 004052 000001  CMP    PARABT,#1    ; : : PARITY
2503 010362 001406      BEQ    25$         ; : : ABORT
2504 010364 005037 004052      CLR    PARABT      ; : : TRAP
2505 010370 005037 172100      CLR    MER         ; : : SHOULD
2506 010374 104063      ERROR  +63        ; : : HAPPEN
2507 010376 000457      BR     40$        ; : : SOON !!!
2508
2509      ; CHECK OUT THE ADDRESS AND EXTENDED MER READ ENABLED
2510
2511 010400      25$:
2512 0. 0400 032737 100001 172100  BIT    #100001,@#MER ; : : ARE BITS 0 AND 15 ARE SET ;MC
2513 010406 001001      BNE    30$        ; : : : YES THEY ARE ;MC
2514 010410 104073      ERROR  +73        ; : : : NO THEY'RE NOT SET ;MC
2515
2516 010412 042737 100000 172100 30$:  BIC    #BIT15,@#MER ; : : : CLEAR THE PARITY ERROR FLAG

```

TEST CHECK PARITY DETECT LOGIC AND RAMS

```

2517 010420 010100      MOV      R1,R0      ; GET THE VIRTUAL ADDRESS
2518 010422 042700 160000 BIC      #160000,R0 ; MASK OUT THE PAGE BITS
2519 010426 072027 177772 ASH      #-6,R0     ; SHIFT OUT BITS 0-5
2520 010432 063700 172344 ADC      @#KIPAR2,R0 ; CREATE BITS 6-21
2521 010436 010004      MOV      R0,R4      ; SAVE A COPY FOR LATER
2522 010440 042704 170037 BIC      #170037,R4  ; CLEAR OUT UNNEEDED BITS
2523 010444 013705 172100 MOV      @#MER,R5    ; READ THE MER
2524 010450 042705 170037 BIC      #170037,R5  ; CLEAR THE UNNEEDED BITS
2525 010454 020405      CMP      R4,R5      ; IF NOT CORRECT BITS SET THEN
2526 010456 001405      BEQ      35$        ; ERROR IN ADDRESS 11-17
2527 010460 010437 001124 MOV      R4,$GDDAT  ; GET EXPECTED DATA
2528 010464 010537 001126 MOV      R5,$BDDAT  ; GET RECEIVED DATA
2529 010470 104064      ERROR    +64        ;
2530 010472 052737 040000 172100 35$: BIS      #BIT14,@#MER ; SET ENABLE READ EXTENDED BIT
2531 010500 013705 172100 MOV      @#MER,R5    ; READ THE MER AGAIN
2532 010504 042705 177037 BIC      #177037,R5  ; CLEAR THE UNNEEDED BITS
2533 010510 072027 177771 ASH      #-7,R0     ; GET BITS 18-21 IN RIGHT PLACE
2534 010514 042700 177037 BIC      #177037,R0  ; MASK OUT UNNEEDED BITS
2535 010520 020005      CMP      R0,R5      ; IF NOT CORRECT BITS SET THEN
2536 010522 001405      BEQ      40$        ; ERROR IN ADDR 18-21 OR READ EXT BIT
2537 010524 010537 001126 MOV      R5,$BDDAT  ; GET RECEIVED DATA
2538 010530 010037 001124 MOV      R0,$GDDAT  ; GET EXPECTED DATA
2539 010534 104065      ERROR    +65        ;
2540 010536 005037 172100 40$: CLR      @#MER      ; RESET MER
2541 010542 005037 004052 CLR      PARABT     ; CLEAR THE PARITY ABORT TRAP
2542 010546 005201      INC      R1         ; BUMP UP THE ADDRESS
2543 010550 020127 060000 CMP      R1,#60000  ; IF WE HAVE PASSED THE PAGE BOUNDARY
2544 010554 103665      BLO      22$        ; THEN
2545 010556 062737 000200 172344 ADD      #200,@#KIPAR2 ; POINT KPAR2 TO A NEW PAGE
2546 010564 012701 040000 MOV      #40000,R1  ; SET THE VIRTUAL ADDRESS TO PAGE 2
2547 010570 023727 172344 020000 CMP      @#KIPAR2,#20000 ;
2548 010576 001254      BNE      22$        ;
2549 010600 112737 000001 003010 MOVB     #1,PARPAT  ; DUNTIL WE HAVE READ ALL ADDRESSES
2550 010606 005303      DEC      R3         ; THIS TIME WE WANT AN ODD # OF 1'S
2551 010610 001412      BEQ      50$        ; DECREMENT LOOP COUNT
2552 010612 012701 040000 MOV      #40000,R1  ; WE WANT TO MAP THRU PAGE 2
2553 010616 012737 001600 172344 MOV      #1600,@#KIPAR2 ; SET UP KPAR 2
2554 010624 052737 000004 172100 BIS      #BIT2,MER  ; ENABLE WRITE WRONG PARITY
OF 2555 010632 000137 010250 JMP      1$         ; DUNTIL WE DONE IT FOR EVEN AND ODD NUMBER
2556 010636      50$:
2557      ;
2558      ; CLEAR ALL OF MEMORY WITH WRITE WRONG PARITY CLEAR
2559      ;
2560 010636 004737 027060 JSR      PC,SETMMU  ; GO SET UP THE MMU REGISTERS
2561 010642 012701 040000 MOV      #40000,R1  ; WE WANT TO MAP THRU PAGE 2
2562 010646 012737 001600 172344 MOV      #1600,@#KIPAR2 ; SET UP KPAR 2
2563 010654      60$: FOR 160000 TO 200000
2564 010654 105021      CLRB     (R1)+      ; CLEAR OUT A BYTE
2565 010656 020127 060000 CMP      R1,#60000  ; IF WE HAVE PASSED THE PAGE BOUNDARY
2566 010662 103774      BLO      60$        ; THEN
2567 010664 062737 000200 172344 ADD      #200,@#KIPAR2 ; POINT KPAR2 TO A NEW PAGE
2568 010672 012701 040000 MOV      #40000,R1  ; SET THE VIRTUAL ADDRESS TO PAGE 2
2569 010676 023727 172344 020000 CMP      @#KIPAR2,#20000 ;
2570 010704 001401      BEQ      70$        ;
2571 010706 000762      BR       60$        ; ENDFOR
2572      ;
2573      ; LET'S NOW MAKE SURE THAT THE ABORTS DON'T OLCUR

```

TEST - CHECK PARITY DETECT LOGIC AND RAMS

2574									
2575	010710				70\$:	MOV	#1600,@#KIPAR2	:	SET UP PAR 2
2576	010710	012737	001600	172344		MOV	#40000,R1	:	SET VIRTUAL ADDRESS TO PAGE #2
2577	010716	012701	040000					:	BGND0
2578	010722				75\$:	BIS	#BIT0,@#MER	:	: ENABLE PARITY ERRORS
2579	010722	052737	C00001	172100		TSTB	(R1)	:	: READ THE BYTE
2580	010730	105711				MOV	#-1,MERTAG	:	
2581	010732	012737	177777	003050				:	
2582								:	: A
2583								:	: PARITY
2584								:	: ABORT
2585								:	: TRAP
2586								:	: SHOULD NOT
2587								:	: HAPPEN !!!
2588								:	
2589								:	
2590								:	
2591	010740	005737	004052			TST	PARABT	:	: IF A PARITY ABORT HAS OCCURED
2592	010744	001405				BEQ	80\$:	: THEN
2593	010746	005037	004052			CLR	PARABT	:	
2594	010752	005037	172100			CLR	MER	:	: INITIALIZE THE MER
2595	010756	104063				ERROR	+63	:	: ERROR A PARITY ABORT HAS OCCURED
2596	010760	005201			80\$:	INC	R1	:	: BUMP UP THE ADDRESS
2597	010762	020127	060000			CMP	R1,#60000	:	: IF WE HAVE PASSED THE PAGE BOUNDARY
2598	010766	103755				BLO	75\$:	: THEN
2599	010770	062737	000200	172344		ADD	#200,@#KIPAR2	:	: POINT KPAR2 TO A NEW PAGE
2600	010776	012701	040000			MOV	#40000,R1	:	: SET THE VIRTUAL ADDRESS TO PAGE 2
2601	011002	023727	172344	020000		CMP	@#KIPAR2,#20000	:	: ENDF
2602	011010	001401				BEQ	90\$:	: DOUNTIL WE HAVE READ ALL ADDRESSES
2603	011012	000743				BR	75\$:	
2604	011014	005037	177572		90\$:	CLR	@#SRO	:	: DISABLE THE MMU
2605	011020	005037	172100			CLR	@#MER	:	
2606	011024	012637	000116			MOV	(SP)+,@#116	:	: RESTORE PRIORITY
2607	011030	012637	000114			MOV	(SP)+,@#114	:	: RESTORE VECTOR
2608	011034	000400				BR	TST11	:	: ;GO TO THE NEXT TEST
2609	011036				NXTST:			:	

TEST - LTC BIT 7

```

2611 .SBTTL TEST LTC BIT 7
2612 :*****
2613 :LTC BIT 7 TEST
2614 :
2615 : This test check for the existance of the LTC register and it
2616 : makes sure that the clock is ticking.
2617 :
2618 :BGNTST
2619 : Set up timeout vector PC to TIMOUT
2620 : Set up timeout vector PSW to 7
2621 : Read the LTC CSR
2622 : IF not timed OUT THEN
2623 : : FOR a set amount of time
2624 : : : Check bit7
2625 : : : IF BIT7 is set THEN
2626 : : : : Increment BIT7 set flag
2627 : : : ELSE
2628 : : : : INCREMENT BIT7 Clear Flag
2629 : : ENDFOR
2630 : : IF either flag has not been set at all THEN
2631 : : : Error Clock is not ticking
2632 : ENDTST
2633 :
2634 : TIMOUT: Clean stack
2635 : : Error LTC register timed out
2636 :
2637 :-----
2638 :
2639 :
2640 :

```

```

:*****;LTC BIT7 TEST*****
2641 011036 000004 010000 170072 ST11: SCOPE
2642 011040 032777 011056 BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
2643 011050 001420 BEQ 1000$ ; THEN TYPE TEST TRACE
011054 000415 TYPE 30027$ ;;TYPE ASCIZ STRING
BR 30026$ ;;GET OVER THE ASCIZ
:30027$: .ASCIZ <15><12>/TEST 11 - LTC BIT7 TEST/
30026$:
1000$:
2644 011110 MOV @#4,SAVTIM ; SAVE UNEXPECTED TIMEOUT
2645 011110 013737 000004 003012 MOV #100$,@#4 ; SET UP THE TIMEOUT TRAP
2646 011116 012737 011216 000004 MOV #340,@#6
2647 011124 012737 000340 000006 TST LKS ; READ THE LKS REGISTER
2648 011132 005737 177546 CLR R0 ; CLEAR THE HIGH COUNTER
2649 011136 005000 CLR R1 ; CLEAR THE LOW COUNTER
2650 011140 005001 MOV #1,R3
2651 011142 012703 000001 MOV #177777,R2 ; SET UP A COUNT
2652 011146 012702 177777 BIT #BIT7,LKS ; FOR AN AMOUNT OF TIME
2653 011152 032737 000200 177546 1$: BEQ 5$ ; CHECK BIT 7 OF THE LKS
2654 011160 001402 INC R0 ; IF IT IS A "1" THEN
2655 011162 005200 BR 10$ ; BUMP UP R0
2656 011164 000401 INC R1 ; ELSE
2657 011166 005201 5$: INC R1 ; BUMP UP R1
2658 011170 077210 10$: SOB R2,1$
2659 011172 005700 TST R0 ; ENDFOR
2660 011174 001402 BEQ 20$ ; IF R0 = 0 OR
2661 011176 005701 TST R1 ; R1 = 0 THEN
2662 011200 001002 BNE 25$ ; ERROR WITH BIT7 OF LTC
2663 011202 077315 20$: SCB R3,1$ ; TOLERATE ONE ERROR ONLY

```

TEST - LTC BIT 7

```
2664 011204 104053          ERROR +53          ; ENDTST
2665
2666 011206 013737 003012 000004 25$:  MOV SAVTIM,@#4      ; RESTORE UNEXPECTED TIMEOUT
2667 011214 000405          BR TST12          ;;GO TO THE NEXT TEST
2668
2669
2670          ; TIMEOUT ROUTINE
2671          ;
2672
2673          100$:
2674 011216 104054          ERROR +54          ; REPORT ERROR
2675 011220 022626          CMP (SP)+,(SP)+  ; CLEAN STACK
2676 011222 013737 003012 000004  MOV SAVTIM,@#4  ; RESTORE UNEXPECTED TIMEOUT
2677
2678
2679
```

TEST - LKS INTERRUPT PRIORITY

```

2681 .SBTTL TEST - LKS INTERRUPT PRIORITY
2682 ;CHECK THAT LKS INTERRUPTS HAPPEN AT PRIORITY 5 CLEARING LKS<07>
2683 ;AND DON'T HAPPEN AT PRIORITY 6.
2684 ;ROUTINE TEST
2685 ;IF UFD AND LKS IS DISABLED THEN
2686 ;EXIT TEST
2687 ;ENDIF
2688 .
2689 .   SET PRIORITY TO 5
2690 .   CLEAR INTERRUPT_FLAG
2691 .   LET LKS<06>=#1 (ENABLE INTERRUPTS)
2692 .   SET COUNTER TO WAIT FOR 3 INTERRUPTS
2693 .   REPEAT
2694 .     DECREMENT COUNTER
2695 .     UNTIL INTERRUPT_FLAG EQ #3 OR COUNTER EQ #0
2696 .     CLEAR LKS<06>
2697 .     IF LKS<07> EQ #1 THEN
2698 .       ERROR (WAS NOT CLEARED ON INTERRUPT)
2699 .     ;ENDIF
2700 .     IF COUNTER LT TIME_REQUIRED_FOR_3_INTERRUPTS_FOR_800HZ
2701 .       ERROR (INTERRUPTS NEVER GO LOW)
2702 .     ;ENDIF
2703 .     IF INTERRUPT_FLAG LT #3 THEN
2704 .       ERROR (INTERRUPTS DON'T HAPPEN)
2705 .     ;ENDIF
2706 .     CLEAR INTERRUPT_FLAG
2707 .     WAIT FOR LKS<7>=1
2708 .     LET LKS<7>=0
2709 .     IF LKS<7> NE #0 THEN
2710 .       ERROR (LKS<7> NOT CLEARED)
2711 .     ;ENDIF
2712 .     SET PRIORITY TO 6
2713 .     SET COUNTER TO 1 SLOW CLOCK INTERRUPT
2714 .     SET LKS<06>
2715 .     REPEAT
2716 .       DECREMENT COUNTER
2717 .       UNTIL COUNTER EQ #0 OR INTERRUPT_FLAG NE #0
2718 .       IF INTERRUPT_FLAG NE #0 THEN
2719 .         ERROR (INTERRUPT WAS AT WRONG PRIORITY)
2720 .       ;ENDIF
2721 .       RESTORE ORIGINAL PRIORITY
2722 ;ENDROUTINE
2723 ;ROUTINE LINE CLOCK INTERRUPT
2724 ;INCREMENT INTERRUPT_FLAG
2725 ;ENDROUTINE
2726
2727 *****
2728 011230 000004 1ST12: SCOPE
2729 011232 032777 010000 167700 BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
2730 011242 104401 011250 BEQ 1000$ ; THEN TYPE TEST TRACE
2731 011314 000422 TYPE 30029$ ;;TYPE ASCIZ STRING
2732 011314 032737 000100 000052 BR 30028$ ;;GET OVER THE ASCIZ
2733 011322 001400 .ASCIZ <15><12>/TEST 12 - LKS INTERRUPT PRIORITY/
2734 011314 000000 30029$:
2735 011314 000000 30028$:
2736 011314 000000 1000$:
2737 011314 000000 BIT #BIT06,@#52 ;UFD MODE?
2738 011314 000000 BEQ 1$ ;IF NOT, GO DO TEST

```

TEST LKS INTERRUPT PRIORITY

```

2734
2735      ; WAIT FOR 3 INTERRUPTS AND CHECK LKS<7> TO BE 0 AFTER INTERRUPT
2736
2737 011324 042737 000100 177546 1$: BIC #BIT06,LKS      ; FROM END OF TEST 42?? PROBLEM
2738 011332 005037 002340          CLR LKSFL          ; CLEAR INTERRUPT FLAG
2739 011336 012737 025722 000100  MOV #LKSINT,100    ; POINT VECTOR TO ROUTINE
2740 011344 012701 077777          MOV #77777,R1     ; COUNTER FOR SLOW CLOCK
2741 011350 052737 000100 177546  BIS #BIT06,LKS    ; SET INTERRUPT ENABLE BIT
2742 011356 032737 000100 177546  BIT #BIT06,LKS    ; BIT SET OK?
2743 011364 001001          BNE 2$            ; IF YES, BRANCH
2744 011366 104042          ERROR +42             ; ERROR WRITING 1 TO LKS<6>
2745 011370 106427 000240          MTPS #240          ; SET PRIORITY TO 5
2746 011374 022737 000003 002340 3$: CMP #3,LKSFL     ; 3 INTERRUPTS HAPPENED?
2747 011402 001401          BEQ 4$            ; IF YES, BRANCH
2748 011404 077105          SOB R1,3$        ; STAY IN A LOOP
2749
2750      ; DISABLE INTERRUPTS AND CHECK THAT PROPER CONDITIONS ARE MET
2751
2752 011406 106427 000340          4$: MTPS #340        ; RAISE PRIORITY
2753 011412 042737 000100 177546  BIC #BIT06,LKS    ; DISABLE INTERRUPTS
2754 011420 032737 000100 177546 6$: BIT #BIT06,LKS ; LKS<6>=0?
2755 011426 001401          BEQ 7$            ; IF YES, BRANCH
2756 011430 104042          ERROR +42             ; ERROR WRITING 0 TO LKS<6>
2757 011432
2758 011432 022737 000003 002340 8$: CMP #3,LKSFL     ; DID 3 INTERRUPTS HAPPEN?
2759 011440 001404          BEQ 9$            ; IF YES, BRANCH
2760 011442 012737 000003 001124  MOV #3,$GDDAT     ; 3 INTERRUPTS EXPECTED
2761 011450 104007          ERROR +7            ; INTERRUPTS DON'T HAPPEN
2762
2763      ; CHECK WHETHER INTERRUPTS HAPPEN AT PRIORITY 6
2764
2765 011452 005037 002340          9$: CLR LKSFL        ; CLEAR INTERRUPT FLAG
2766 011456 106427 000300          MTPS #300        ; RAISE PRIORITY TO 6
2767 011462 012701 077777          MOV #77777,R1     ; COUNTER FOR SLOW CLOCK
2768 011466 052737 000100 177546  BIS #BIT06,LKS    ; SET INTERRUPT ENABLE BIT
2769 011474 005737 002340          10$: TST LKSFL     ; ANY INTERRUPTS?
2770 011500 001001          BNE 11$          ; IF YES, EXIT LOOP
2771 011502 077104          SOB R1,10$       ; CONTINUE WITH COUNT
2772 011504 005737 002340          11$: TST LKSFL     ; ANY INTERRUPTS?
2773 011510 001404          BEQ 12$          ; IF NO, BRANCH
2774 011512 012737 000005 001124  MOV #5,$GDDAT     ; STORE PRIORITY FOR TYPE OUT
2775 011520 104010          ERROR +10        ; INTERRUPTS HAPPEN AT WRONG PRIORITY
2776 011522 106427 000340          12$: MTPS #340        ; RESTORE PRIORITY
2777 011526 042737 000100 177546  BIC #BIT6,LKS    ; DISABLE CLOCK INTERRUPTS
2778

```

TEST - RESETTING LKS

```

2780 .SBTTL TEST - RESETTING LKS
2781 ;RESETTING LKS(*)
2782 ;THIS TEST WILL PROVE THAT RESET INSTRUCTION CLEARS LKS<06>.
2783 ;ROUTINE TEST
2784 ;IF UFD AND LKS IS DISABLED THEN
2785 ;EXIT TEST
2786 ;ENDIF
2787 ;. POINT LKS VECTOR 100 TO ERROR LKS ILLEGAL_INTERRUPT
2788 ;. SYNCHRONIZE LKS BY WAITING FOR 3 PULSES
2789 ;. LET LKS<06>=#1
2790 ;. EXECUTE 'RESET'
2791 ;. IF LKS<6> NE #0 THEN
2792 ;. ERROR
2793 ;. ENDF
2794 ;. IF ILLEGAL_LINE_CLOCK_INTERRUPT NE 0 THEN
2795 ;. ERROR
2796 ;. ENDF
2797 ;ENDROUTINE
2798 ;ROUTINE ERROR LKS ILLEGAL_INTERRUPT
2800 ;FLAG ICLEGAL_LINE_CLOCK_INTERRUPT
2801 ;RETURN
2802
2803 *****
2804 011534 000004 †ST13: SCOPE
2805 011536 032777 010000 167374 BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
2806 011544 001423 BEQ 1000$ ; THEN TYPE TEST TRACE
2806 011546 104401 011554 TYPE 30031$ ;;TYPE ASCIZ STRING
2806 011552 000420 BR 30030$ ;;GET OVER THE ASCIZ
2807 011614 ;:30031$: .ASCIZ <15><12>/TEST 13 - RESETTING LKS TEST/
2807 011614 30030$: 1000$:
2808 011614 005737 001206 TST $PASS ;FIRST PASS?
2809 011620 001040 BNE TST14 ;;IF NOT FIRST PASS, EXIT TEST
2810 011622 032737 000100 000052 BIT #BIT06,@#52 ;UFD MODE?
2811 011630 001400 BEQ 1$ ;IF NOT, BRANCH
2812
2813 ; SYNCHRONISE WITH LINE TIME CLOCK BY WAITING FOR 3 INTERRUPTS
2814
2815 011632 012737 025722 000100 1$: MOV #LKSINT,@#100 ;SET UP INTERRUPT VECTOR
2816 011640 012737 000340 000102 MOV #340,@#102 ;AT PROIRITY 7
2817 011646 052737 000100 177546 BIS #BIT06,LKS ;SET INTERRUPT ENABLE BIT
2818 011654 005037 002340 CLR LKSFL ;CLEAR INTERRUPTS FLAG
2819 011660 012702 077777 MOV #77777,R2 ;COUNTER TO WAIT FOR INTERRUPTS
2820 011664 106427 000240 MTPS #240 ;LOWER PRIORITY TO 5
2821 011670 022737 000003 002340 2$: CMP #3,LKSFL ;3 INTERRUPTS HAPPENED?
2822 011676 001401 BEQ 3$ ;EXIT LOOP, IF SO
2823 011700 077205 SOB R2,2$ ;OTHERWISE, KEEP WAITING
2824 011702 106427 000340 3$: MTPS #340 ;RAISE PRIORITY TO 7
2825 011706 000005 RESET ;EXECUTE RESET
2826 011710 032737 000100 177546 4$: BIT #BIT06,L<S ;INTERRUPT ENABLE BIT CLEARED?
2827 011716 001401 BEQ TST14 ;;IF SO, EXIT TEST
2828 011720 104011 ERROR +11 ;RESET DOESN'T CLEAR LKS
2829
2830

```


TEST - MAINTENANCE REGISTER TEST

```

2832 .SBITL TEST - MAINTENANCE REGISTER TEST
2833 ;MAINTENANCE REGISTER TEST
2834 ;THIS TEST WILL ADDRESS MAINTENANCE REGISTER AND CHECK BITS
2835 ;7-4 TO BE 0010, 2-1 TO BE 10, AND READ BITS 10-08, 03, 00
2836 ;FOR FUTURE USE. THOSE BITS REPRESENT THE FOLLOWING SIGNALS:
2837 ;MULTIPROCESSOR SLAVE, UNIBUS SYSTEM, FPA AVAILABLE, HALT/TRAP
2838 ;OPTION, AND AC POWER OKAY.
2839 ;ROUTINE TEST
2840 ;. IF MAINT. REG. BITS <7 4> NE 0010 OR <2-1> NE 10 THEN
2841 ;. ERROR
2842 ;. ENDF
2843 ;. READ MAINT.REG. BITS <10-08,03,00>
2844 ;ENDROUTINE
2845
2846 ;*****
2847 011722 000004 1ST14: SCOPE
2848 011724 032777 010000 167206 BIT #BIT12,SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
2849 011732 001426 BEQ 1000$ ; THEN TYPE TEST TRACE
2849 011734 104401 011742 TYPE 30033$ ;;TYPE ASCIZ STRING
2849 011740 000423 BR 30032$ ;;GET OVER THE ASCIZ
2849 ;.30033$: .ASCIZ <15><12>/TEST 14 - MAINTENANCE REGISTER TEST/
2849 ;.30032$:
2849 ;.1000$:
2850 012010 BIT #177000,MAIREG ;UNUSED BITS ALL ZEROS? ;MC001
2851 012010 032737 177000 177750 BEQ 1$ ;CHANGED FROM 174000 FOR THE DA
2852 012016 001401 ERROR +43 ;IF OK, BRANCH
2853 012020 104043 BIT #BIT3,MAIREG ;MAINTENANCE REGISTER ERROR
2854 012022 032737 000010 177750 1$: BNE 2$ ;IF JUMPER IS IN
2855 012030 001010 ;. THEN
2856 012032 042737 000010 177750 BIC #BIT3,MAIREG ;. SET HALT TO ODT
2857 012032 042737 000010 177750 CMP #105,MAIREG ;. MAINTENANCE REGISTER SHOULD BE SET UP TO 1
052858 012040 022737 000105 177750 BEQ 2$ ;. CHANGED FROM <5,2> FOR THE DA
2859 012046 001401 ERROR +43 ;IF SO, BRANCH
2860 012050 104043 ;MAINTENANCE REGISTER ERROR
2861 012052
2862
2863

```

TEST - SERIAL LINE UNIT REGISTERS

```

2865 .SBTTL TEST - SERIAL LINE UNIT REGISTERS
2866 ;SERIAL LINE UNIT TEST(*)
2867 ;BCR<2-0> WILL BE READ TO FIND OUT BAUD RATE. SLU WILL BE PROG-
2868 ;RAMMED TO CHECK THE INTERRUPT LEVELS BY SETTING BIT<06> IN
2869 ;RCSR AND XMIT. LOOP BACK CAPABILITIES WILL BE TESTED BY SETTING
2870 ;TO 1 XCSR<02>. THE LINE CLOCK INTERRUPT SUBROUTINE WILL BE
2871 ;USED TO RETURN TO THE EXECUTION OF THE DIAGNOSTICS, IF THE
2872 ;PROGRAM HANGS IN THE LOOP BACK MODE.
2873 ;ROUTINE TEST
2874 ;IF UFD AND CONSOLE NOT PRESENT
2875 ; GO TO TEST_22
2876 ;ENDIF
2877 ; IF BCR<07> EQ #0 THEN
2878 ; READ BCR<2-0> TO GET BAUD RATE
2879 ; ENDF
2880 ; LET 4=ADDRESS OF TIMEOUT ROUTINE
2881 ; DO FOR RCSR,XCSR,RBUF,XBUF
2882 ; READ XRCR,XCSR,RBUF,XBUF
2883 ; IF TIMEOUT FLAG NE #0 THEN
2884 ; ERROR
2885 ; ENDF
2886 ; ENDDO
2887 ;ENDROUTINE
2888
2889 ;ROUTINE TIMEOUT
2890 ; LET TIMEOUT_FLAG=#1
2891 ;ENDROUTINE
2892
2893 ;*****
2894 012052 000004 1ST15: SCOPE
2895 012054 032777 010000 167056 BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
2896 012062 001426 BEQ 1000$ ; THEN TYPE TEST TRACE
2896 012064 104401 012072 TYPE 30035$ ;;TYPE ASCIZ STRING
2896 012070 000423 BR 30034$ ;;GET OVER THE ASCIZ
2897 012140 ;:30035$: .ASCIZ <15><12>/TEST 15 - SLU REGISTER ACCESS TEST/
2897 012140 30034$:
2898 012140 032737 00010 000052 1000$:
2899 012146 001402 BIT #BIT06,@#52 ;UFD MODE?
2900 012150 000137 015262 BEQ 1$ ;IF NOT, GO DO THE TEST
2901 JMP SLEND ;IF TRUE, SKIP ALL SLU TESTS
2902 ; TRY TO ACCESS SLU REGISTERS
2903
2904 012154 013701 000004 1$: MOV ERRVEC,R1 ;SAVE TIMEOUT VECTOR
2905 012160 012737 012214 000004 MOV #3$,ERRVEC ;POINT NEW ONE TO PROGRAM AREA
2906 012166 012737 000340 000006 MOV #340,ERRVEC+2 ;AT PRIORITY 7
2907 012174 012702 177560 MOV #RCSR,R2 ;START ACCESSING WITH RCSR
2908 012200 012703 177566 MOV #XBUF,R3
2909 012204 012704 000002 MOV #2,R4
2910 012210 005712 2$: TST (R2) ;ACCESS SLU REGISTER
2911 012212 000403 BP 4$ ;IF NO TIMEOUT, CONTINUE
2912 012214 010237 001126 3$: MOV R2,$BDDAT ;STORE ADDRESS THAT TIMED OUT
2913 012220 104012 ERROR +12 ;TIMEOUT ACCESSING SLU REGISTER
2914 012222 020322 4$: CMP R3,(R2)+ ;LAST REGISTER ACCESSED?
2915 012224 103771 BLO 2$ ;IF NOT, BRANCH
2916 012226 012702 176500 MOV #RCSR1,R2 ;GET POINTERS TO SLU 1
2917 012232 012703 176506 MOV #XBUF1,R3 ;GET LAST ADDRESS ON SLU 1

```

TEST - SERIAL LINE UNIT REGISTERS

2918 012236 077414
2919 012240 010137 000004
2920

SOB R4.2\$
MOV R1.ERRVEC

;RESTORE TIMEOUT VECTOR

TEST - XCSR BIT 7

2922
2923
2924
2925
2926
2927
2928
2929
2930
2931
2932
2933
2934
2935
2936
2937
2938
2939
2940

2941
2942
2943
2944
2945
2946
2947
2948
2949
2950

2951
2952
2953
2954
2955
2956
2957
2958
2959
2960
2961
2962
2963
2964
2965
2966
2967
2968
2969
2970
2971
2972

012244 000004
012246 122737 000001 001220
012254 001006
012256 005737 001206
012262 001403
012264 012704 000002
012270 000453
012272
012272 032777 010000 166640
012300 001424
012302 104401 012310
012306 000416

012344
012344 012701 100000
012350 077101
012352
012352 012701 001000
012356 012702 177564
012362 012703 177566
012366 012704 000002
012372 105712
012374 100401
012376 077103
012400 105712
012402 100401
012404 104013
012406 012713 000000
012412 105712
012414 100001
012416 104013
012420 012702 176504
012424 012703 176506
012430 012701 001000
012434 077422

```
.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
:ENDIF
:LET XBUF=#NULL
:WAIT FOR XCSR<07>=#1
:LET XBUF=#NULL
:IF XCSR<07> NE 0 THEN
:ERROR (READY DIDN'T GO LOW)
:ENDIF
:ENDROUTINE

:*****
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 30037$ ;;TYPE ASCIZ STRING
BR 30036$ ;;GET OVER THE ASCIZ
30037$: .ASCIZ <15><12>/TEST 16 - XCSR BIT 7 TEST/
30036$: MOV #100000,R1
1001$: SOB R1,1001$ ; WAIT FOR LAST CHARACTER
1000$: MOV #1000,R1 ;COUNTER FOR ABOUT 200MICROSEC.
MOV #XCSR,R2
MOV #XBUF,R3
MOV #2,R4
1$: TSTB (R2) ;XCSR<7> READY 1?
BMI 2$ ;IF SO, EXIT WAIT LOOP
SOB R1,1$ ;IF NOT 1, CONTINUE WAITING
2$: TSTB (R2) ;XCSR<7>=#1?
BMI 3$ ;IF YES, BRANCH
ERROR +13 ;XCSR<7> DOES NOT BECOME 1
3$: MOV #NULL,(R3) ;TRY TO TRANSMIT NULL CHARACTER
TSTB (R2) ;XCSR<7>=0
BPL 4$
ERROR +13 ;XMIT READY DIDN T GO LOW
4$: MOV #XCSR1,R2
MOV #XBUF1,R3
MOV #1000,R1
SOB R4,1$
```

TEST - RCSR BIT 7 AND XCSR BIT 2

```

2974 .SBTTL TEST RCSR BIT 7 AND XCSR BIT 2
2975 ;CHECK THAT RCSR<07> CAN BE 0 AND 1 AND THAT XCSR<02> WORKS PROPERLY.
2976
2977 ;RCSR <07> RECEIVER DONE
2978 ;XCSR <02> MAINTENANCE
2979
2980 ;ROUTINE TEST
2981 ;.(CHECK RCSR<07> AND XCSR<07>)
2982 ;. WAIT FOR XCSR<07>=#1
2983 ;. LET XCSR<02>=#1 (LOOP BACK MODE)
2984 ;. LET XBUF=#125
2985 ;. WAIT FOR RCSR<07>=#1 NO MORE THAN 200MSEC
2986 ;. IF RCSR<07> NE #1 THEN
2987 ;. ERROR (RCSR<07> DOES NOT BECOME 1 OR XCSR<02>DOES NOT
2988 ;. WORK)
2989 ;. ENDIF
2990 ;. IF RBUF NE #125 THEN
2991 ;. ERROR
2992 ;. ENDIF
2993 ;. IF RCSR<07> NE #0 THEN
2994 ;. ERROR (RCSR<07>DOES NOT GO LOW)
2995 ;. ENDIF
2996 ;. LET XCSR<02>=#0
2997 ;ENDROUTINE
2998
2999 *****
3000 012436 000004 ;TST17: SCOPE
3001 012440 122737 000001 001220 CMPB #APTENV,$ENV ;ARE WE IN APT MODE?
3002 012446 001006 BNE 1002$ ;IF NOT: DO THIS TEST
3003 012450 005737 001206 TST $PASS ;FIRST PASS??
3004 012454 001403 BEQ 1002$ ;IF YES, DO IT
3005 012456 012703 000002 MOV #2,R3 ; LOOP COUNT OF 1
3006 012462 000536 BR 12$ ; GO SET UP POINTERS TO SLU 1
3007 012464 032777 010000 166446 1002$: BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
3008 012472 001434 BEQ 1000$ ; THEN TYPE TEST TRACE
3009 012474 104401 012502 TYPE 30039$ ;:TYPE ASCIZ STRING
3010 012500 000426 BR 30038$ ;:GET OVER THE ASCIZ
3011 012556 30039$: .ASCIZ <15><12>/TEST 17 - RCSR BIT 7 AND XCSR BIT 2 TEST/
3012 012556 012701 100000 30038$: MOV #100000,R1
3013 012562 077101 1001$: SOB R1,1001$ ; WAIT FOR LAST CHARACTER
3014 012564 1000$:
3015 012564 012701 000013 MOV #13,R1 ;COUNTER FOR ABOUT 200MICROSEC.
3016 012570 012702 177560 MOV #RCSR,R2 ;
3017 012574 012703 000002 MOV #2,R3
3018 012600 105762 000004 1$: TSTB 4(R2) ;XCSR<7> READY 1?
3019 012604 100375 BPL 1$ ;IF NOT 1, CONTINUE WAITING
3020 012606 052762 000004 000004 2$: BIS #BIT02,4(R2) ;SET LOOP BACK MODE
3021 012614 032762 000004 000004 BIT #BIT02,4(R2) ;GOT SET OK?
3022 012622 001004 BNE 3$ ;IF YES, BRANCH
3023 012624 005062 000004 CLR 4(R2) ;PESET TO PRINT ERROR
3024 012630 104034 ERROR +34 ;XCSR<2> DOES NOT BECOME 1
3025 012632 000457 BR TST20 ;;EXIT TEST
3026 012634 012701 060000 3$: MOV #60000,R1 ;STALL IN CASE XCSR<2> SETS READ

```

TEST - RCSR BIT 7 AND XCSR BIT 2

```
3027 012640 105712          4$:  TSTB    (R2)          ;IF RECEIVER READY SET?
3028 012642 100402          BMI     5$              ;IF SET, BRANCH
3029 012644 077103          SOB    R1,4$           ;OTHERWISE, STAY FOR A WHILE
3030 012646 000402          BR     6$              ;IF NOT READY, BRANCH
3031 012650 005762 000002    5$:  TST     2(R2)       ;READ RBUF
3032
3033          ; TRANSMIT XON AND CHECK RCSR<7>
3034
3035 012654 012762 000021 000006  6$:  MOV     #21,6(R2)      ;TRANSMIT A CHARACTER
3036 0 2662 012701 060000     MOV     #60000,R1      ;COUNTER TO WAIT
3037 012666 105712          7$:  TSTB    (R2)          ;RCSR<7> READY 1?
3038 012670 100401          BMI     8$              ;IF YES, EXIT WAIT LOOP
3039 012672 077103          SOB    R1,7$           ;OTHERWISE, CONTINUE WAITING
3040 012674 105712          8$:  TSTB    (R2)          ;RCSR<7>=1?
3041 012676 100403          BMI     9$              ;IF YES, BRANCH
3042 012700 005062 000004     CLR     4(R2)          ;RESET XCSR<2>
3043 012704 104014          ERROR  +14             ;RECEIVER READY DIDN'T COME UP
3044 012706 016237 000002 001126  9$:  MOV     2(R2), $BDDAT  ;STORE RECEIVED DATA
3045 012714 022737 000021 001126     CMP     #21, $BDDAT    ;DATA RECEIVED OK?
3046 012722 001406          BEQ    10$             ;IF YES, BRANCH
3047 012724 012737 000021 001124     MOV     #21, $GDDAT
3048 012732 005062 000004     CLR     4(R2)          ;RESET TO ENABLE SLU
3049 012736 104015          ERROR  +15             ;WRONG CHARACTER RECEIVED
3050 012740 105712          10$: TSTB    (R2)          ;RCSR<7>=0?
3051 012742 100003          BPL    11$             ;IF ZERO, BRANCH
3052 012744 005062 000004     CLR     4(R2)          ;RESET TO ENABLE SLU
3053 012750 104016          ERROR  +16             ;RCSR<07><<0 AFTER READING RBUF
3054 012752 042762 000004 000004 11$:  BIC     #BIT02,4(R2)   ;DISABLE LOOP BACK MODE
3055 012760 012702 176500    12$:  MOV     #RCSR1,R2     ;POINT TO SLU 1
3056 012764 012701 000013     MOV     #13,R1        ;SET UP COUNTER
3057 012770 077375          SOB    R3,1$          ;
3058
```

N6

0010

TEST RESET AND XCSR<2!0>

```

3060 .SBTTL TEST RESET AND XCSR<2!0>
3061 ;CHECK THAT RESET CLEARS XCSR<0!2>.
3062 ;ROUTINE TEST
3063 ;. (CHECK RCSR<07> AND XCSR<07> AND RESET)
3064 ;. LET XCSR<02,00>=#1 (LOOP BACK MODE)
3065 ;. EXECUTE "RESET"
3066 ;. IF XCSR<02!00> NE #0 THEN
3067 ;. ERROR
3068 ;. ENDIF
3069 ;. LET XCSR<02>=#0
3070 ;ENDROUTINE
3071
3072 ;*****
3073 012772 000004 ;ST20: SCOPE
3074 012774 122737 000001 001220 CMPB #APTENV,$ENV ;ARE WE IN APT MODE?
3075 013002 001006 BNE 1001$ ;IF NOT: DO THIS TEST
3076 013004 005737 001206 TST $PASS ;FIRST PASS??
3077 013010 001403 BEQ 1001$ ;IF YES, DO IT
3078 013012 012703 000002 MOV #2,R3 ; LOOP COUNT OF 1
3079 013016 000447 BR 10$ ; GO SET UP POINTERS TO SLU 1
3080 013020 032777 010000 166112 1001$ BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
3081 013026 001426 BEQ 1000$ ; THEN TYPE TEST TRACE
3082 013030 104401 013036 TYPE 30041$ ;TYPE ASCIZ STRING
3083 013034 000423 BR 30040$ ;GET OVER THE ASCIZ
3084 013104 ;:30041$: .ASCIZ <15><12>/TEST 20 - RESET AND XCSR<0!2> TEST/
3085 013104 ;:30040$:
3086 013104 012702 177564 MOV #XCSR,R2 ;
3087 013110 012703 000002 MOV #2,R3 ;
3088 013114 052712 000005 1$: BIS #BIT02!BIT00,(R2) ;LOOP BACK MODE
3089 ; EXECUTE RESET AND VALIDATE THAT XCSR<7,2> BECOMES <1,0>
3090 ;
3091 013120 000005 RESET ;EXECUTE RESET
3092 013122 032712 000005 BIT #BIT02!BIT00,(R2) ;XCSR<2,0> CLEAR?
3093 013126 001403 BEQ 10$
3094 013130 042712 000005 BIC #BIT02!BIT00,(R2) ;CLEAR THE BITS SO WE CAN REPORT
3095 013134 104022 ERROR +22 ;XCSR<2,0> NOT CLEARED ON RESET
3096 013136 012702 176504 10$: MOV #XCSR1,R2 ;
3097 013142 077314 SOB R3,1$ ;

```

TEST - RESET AND INTERRUPT ENABLE BITS

```

3098 .SBTTL TEST - RESET AND INTERRUPT ENABLE BITS
3099 ;CHECK THAT INTERRUPTS DON'T HAPPEN AT PRIORITY 4 AND THAT RESET
3100 ;CLEARS XCSR<06> AND RCSR<06>.
3101 ;
3102 ;RCSR <06> RECEIVER INTERRUPT ENABLE
3103 ;XCSR <06> TRANSMITTER INTERRUPT ENABLE
3104 ;
3105 ;ROUTINE TEST
3106 ;.
3107 ;. LET 60=#ADDRESS_OF_ILLEGAL_INTERRUPT_XRCSR
3108 ;. LET 64=#ADDRESS_OF_ILLEGAL_INTERRUPT_XRCSR
3109 ;. SET PRIORITY TO 4
3110 ;. LET XCSR<02>=#1 (LOOPBACK MODE)
3111 ;. LET XCSR<06>=#1 (ENABLE TRANSMIT INTERRUPTS)
3112 ;. LET RCSR<06>=#1 (ENABLE RECEIVE INTERRUPTS)
3113 ;. WAIT FOR XCSR<07>=#1 (READY TO TRANSMIT)
3114 ;. LET XBUF=#NULL (SEND A CHARACTER)
3115 ;. WAIT FOR ILLEGAL INTERRUPTS (ABOUT 200MSEC)
3116 ;. EXECUTE "RESET"
3117 ;. IF XCSR<06> NE #0 OR RCSR<06> NE #0 OR XRCSR NE #0 THEN
3118 ;. ERROR
3119 ;. ENDF
3120 ;. RESTORE PRIORITY TO NORMAL
3121 ;. ENDRoutine
3122 ;ROUTINE ILLEGAL_INTERRUPT_XRCSR
3123 ;. INCREMENT XRCSR
3124 ;. ENDRoutine
3125 ;
3126 ;*****
3127 013144 000004 1ST21: SCOPE
3128 013146 122737 000001 001220 CMPB #APTENV,$ENV ;ARE WE IN APT MODE?
3129 013154 001015 BNE 1001$ ;IF NOT: DO THIS TEST
3130 013156 005737 001206 TST $PASS ;FIRST PASS??
3131 013164 012705 000002 BEQ 1001$ ;IF YES, DO IT
3132 013170 013737 000004 003012 MOV #2,R5 ; LOOP COUNT OF 1
3133 013176 012737 013666 000004 MOV @#4,SAVTIM ; STORE UNEXPECTED TIMEOUT
3134 013204 000137 013636 JMP 20$ ; SET UP TIMEOUT
3135 013210 032777 010000 165722 1001$: BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
3136 013216 001433 BEQ 1000$ ; THEN TYPE TEST TRACE
3137 013220 104401 013226 TYPE 30043$ ;;TYPE ASCIZ STRING
013224 000430 BR 30042$ ;;GET OVER THE ASCIZ
013306 ;.ASCIZ <15><12>/TEST 21 SLU RESET AND INTERRUPT ENABLE TEST/
3138 013306 30042$:
3139 013306 012737 013666 000004 1000$: MOV #100$,@#4 ;
3140 013314 012701 000060 MOV #60,R1 ; R1 POINTS TO SLU 0 INTERRUPT VECTOR
3141 013320 012704 177560 MOV #RCSR,R4 ; R4 POINTS TO SLU 0 REGISTERS
3142 013324 012705 000002 MOV #2,R5 ; R5 IS THE LOOP COUNT
3143 ;
3144 ; CHECK THAT INTERRUPTS ENABLE BITS FOR RECEIVER AND TRASMITTER OF SLU
3145 ; ARE CLEARED BY RESET
3146 ;
3147 013330 052764 000100 000004 1$: BIS #BIT06,4(R4) ;SET INTERRUPT ENABLE BIT IN XCSR
3148 013336 032764 000100 000004 BIT #BIT06,4(R4) ;GOT SET OK?
3149 013344 001001 BNE 2$ ;IF YES, BRANCH
3150 013346 104067 ERROR +67 ;IN BIT 6 OF XCSR

```


TEST RESET AND INTERRUPT ENABLE BITS

```

3151 013350 052714 000100      2$:  BIS      #BIT06,(R4)      ;SET INTERRUPT ENABLE BIT IN RCSR
3152 013354 032714 000100      BIT      #BIT06,(R4)      ;GOT SET OK?
3153 013360 001001      BNE     3$                ;IF YES, BRANCH
3154 013362 104067      ERROR   +67              ;IN BIT 6 OF RCSR
3155 013364 000005      3$:  RESET    ;INLINE BUS RESET
3156 013366 032764 000100 000004  BIT      #BIT06,4(R4)     ;XMIT INTERRUPT ENABLE BIT CLEARED?
3157 013374 001401      BEQ     4$                ;IF CLEARED, BRANCH
3158 013376 104022      ERROR   +22              ;INTERRUPT ENABLE NOT CLEARED ON RESET
3159 013400 032714 000100      4$:  BIT      #BIT06,(R4)     ;RECEIVE INTERRUPT ENBLE CLEARED?
3160 013404 001401      BEQ     5$                ;IF CLEARED, BRANCH
3161 013406 104022      ERROR   +22              ;INTERRUPT ENABLE NOT CLEARED ON RESET
3162
3163      ; CHECK THAT TRANSMIT INTERRUPTS DON'T HAPPEN AT PRIORITY HIGHER THAN 3
3164
3165 013410      5$:
3166 013410 012761 013454 000004  MOV      #9$,4(R1)        ;POINT XMIT VECTOR TO PROGRAM AREA
3167 013416 012761 000340 000006  MOV      #340,6(R1)       ;AT PRIORITY 7
3168 013424 052764 000100 000004  BIS      #BIT06,4(R4)     ;SET INTERRUPT ENABLE BIT IN XCSR
3169 013432 012702 000340      MOV      #340,R2         ;SET PRIORITY TO 7
3170 013436 000402      BR       7$              ;GO WAIT IN CASE OF INTERRUPTS
3171 013440 162702 000040      6$:  SUB      #40,R2         ;LOWER PRIORITY LEVEL
3172 013444 106402      7$:  MTPS    R2             ;SET PRIORITY
3173 013446 004737 026772      JSR     PC,DELAY         ;TIMER ROUTINE ;MC
3174
3175      ;8$:
3176 013452 000402      SOB     R3,8$           ;WAIT FOR INTERRUPTS
3177 013454 104021      BR      10$            ;IF INTERRUPTS DIDN'T HAPPENED, BRANCH
3178 013456 022626      9$:  ERROR   +21          ;INTERRUPTS HAPPEN AT WRONG PRIORITY
3179 013460 022702 000200      10$: CMP     (SP)+,(SP)+    ;CLEAN UP THE STACK
3180 013464 001365      BNE     6$              ;AT PRIORITY 4?
3181 013466 106427 000340      MTPS    #340            ;IF NOT LAST ONE, CONTINUE
3182 013472 042764 000100 000004  BIC     #BIT06,4(R4)     ;RESTORE PRIORITY 7
3183
3184      ; CHECK THAT RECEIVE INTERRUPTS DON'T HAPPEN AT PRIORITY HIGHER THAN 3
3185
3186 013500 012711 013562      MOV      #15$,(R1)       ;POINT RECEIVE VECTOR TO PROGRAM AREA
3187 013504 012761 000340 000002  MOV      #340,2(R1)       ;AT PRIORITY 7
3188 013512 105764 000004      11$: TSTB    4(R4)          ;TRANSMITTER READY
3189 013516 100375      BPL     11$             ;IF NOT, WAIT
3190 013520 012764 000000 000006  MOV      #NULL,6(R4)      ;TRY TO TRANSMIT NULL
3191 013526 052714 000100      BIS      #BIT06,(R4)     ;SET INTERRUPT ENABLE BIT IN RCSR
3192 013532 012702 000340      MOV      #340,R2         ;SET PRIORITY TO 7
3193 013536 000402      BR      13$            ;GO WAIT IN CASE OF INTERRUPTS
3194 013540 162702 000040      12$: SUB      #40,R2         ;LOWER PRIORITY LEVEL
3195 013544 052764 000004 000004  13$: BIS      #BIT02,4(R4)  ;SET LOOP BACK MODE
3196 013552 106402      MTPS    R2             ;SET PRIORITY
3197 013554 004737 026772      JSR     PC,DELAY         ;TIMER ROUTINE ;MC
3198
3199      ;14$:
3200 013560 000405      SOB     R3,14$          ;WAIT FOR INTERRUPTS
3201 013562 042764 000004 000004  BR      16$            ;IF INTERRUPTS DIDN'T HAPPENED, BRANCH
3202 013570 104021      15$: BIC     #BIT02,4(R4)  ;CLEAR LOOP BACK MODE
3203 013572 022626      ERROR   +21          ;INTERRUPTS HAPPEN AT WRONG PRIORITY
3204 013574 022702 000200      16$: CMP     (SP)+,(SP)+    ;CLEAN UP THE STACK
3205 013600 001357      CMP     #200,R2         ;AT PRIORITY 4?
3206
3207      BNE     12$          ;IF NOT LAST ONE, CONTINUE
3208
3209      ; CLEAN UP BEFORE NEXT TEST

```

TEST - RESET AND INTERRUPT ENABLE BITS

```

3208
3209 013602 106427 000340      MTPS   #340      ;RESTORE PRIORITY 7
3210 013606 042714 000100      BIC    #BIT06,(R4) ;CLEAR INTERRUPT ENABLE BIT
3211 013612 012702 000400      MOV    #400,R2    ;STALL DELAY
3212 013616 105714          17$:  TSTB   (R4)      ;RECEIVE READY?
3213 013620 100401          BMI    18$        ;STOP WAITING, IF SO
3214 013622 077203          SOB    R2,17$    ;OTHERWISE, STAY IN THE LOOP
3215 013624 005764 000002      TST    2(R4)     ;READ CHARACTER TRANSMITTED
3216 013630 042764 000004 000004 18$:  BIC    #BIT02,4(R4) ;CLEAR LOOP BACK MODE
3217 013636 012701 000300      MOV    #300,R1   ;POINT TO SLU #1 VECTOR
3218 013642 012704 176500      MOV    #RCSR1,R4 ;POINT TO SLU #1 REGISTER
3219 013646 005305          DEC    R5
3220 013650 001402          BEQ    19$
3221 013652 000137 013330      JMP
3222 013656 013737 003012 000004 19$:  MOV    SAVTIM,@#4 ;RESTORE UNEXPECTED TIMEOUT
3223 013664 000417          BR     TST22    ;;GO TO THE NEXT TEST
3224
3225          ;
3226          ; DEBUG PURPOSES
3227          ;
3228
3229 013666 106427 000340      100$: MTPS   #340      ;RESTORE PRIORITY 7
3230 013672 042714 000100      BIC    #BIT06,(R4) ;CLEAR INTERRUPT ENABLE BIT
3231 013676 012702 000400      MOV    #400,R2    ;STALL DELAY
3232 013702 105714          170$: TSTB   (R4)      ;RECEIVE READY?
3233 013704 100401          BMI    180$      ;STOP WAITING, IF SO
3234 013706 077203          SOB    R2,170$   ;OTHERWISE, STAY IN THE LOOP
3235 013710 005764 000002      180$: TST    2(R4)     ;READ CHARACTER TRANSMITTED
3236 013714 042764 000004 000004 BIC    #BIT02,4(R4) ;CLEAR LOOP BACK MODE
3237 013722 000000          HALT
3238

```

TEST - INTERRUPT PRIORITY FOR SLU

```

3240 .SBTTL TEST - INTERRUPT PRIORITY FOR SLU
3241 ;CHECK THAT INTERRUPTS HAPPEN AT PRIORITY 3 AND THAT THEY CLEAR
3242 ;RCSR<06> AND XCSR<06>.
3243
3244 ;ROUTINE TEST
3245 .. LET 60=#ADDRESS_OF_LEGAL_RINTERRUPT
3246 .. LET 64=#ADDRESS_OF_LEGAL_XINTERRUPT
3247 .. LET XCSR<02> #1
3248 .. SET PRIORITY TO #3
3249 .. WAIT FOR XINTERRUPT=#3
3250 .. IF XCSR<07> EQ #1 THEN
3251 ..     ERROR
3252 ..     .ENDIF
3253 ..     WAIT FOR RINTERRUPT=#3
3254 ..     IF RCSR<07> EQ #0 THEN
3255 ..         ERROR
3256 ..     .ENDIF
3257 ..     LET XCSR<02>=#0
3258 ..     SET PRIORITY TO NORMAL
3259 ;ENDROUTINE
3260
3261 ;ROUTINE LEGAL XINTERRUPT
3262 .. LET XBOF=#CHARACTER
3263 .. INCREMENT XINTERRUPT
3264 ;ENDROUTINE
3265
3266 ;ROUTINE LEGAL RINTERRUPT
3267 .. READ RCSR
3268 .. INCREMENT RINTERRUPT
3269 ;ENDROUTINE
3270
3271 ;*****
3272 013724 000004 1ST22: SCOPE
3273 013726 122737 000001 001220  CMPB #APTENV,$ENV ;ARE WE IN APT MODE?
3274 013734 001007  BNE 1002$ ;IF NOT: DO THIS TEST
3275 013736 005737 001206  TST $PASS ;FIRST PASS??
3276 013742 001404  BEQ 1002$ ;IF YES, DO IT
3277 013744 012705 000002  MOV #2,R5 ; LOOP COUNT OF 1
3278 013750 000137 014320  JMP 20$ ; GO SET UP POINTERS TO SLU 1
3279 013754 032777 010000 165156 1002$: BIT #BIT12,$SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
3280 013762 001432  BEQ 1000$ ; THEN TYPE TEST TRACE
3280 013764 104401 013772  TYPE ,30045$ ;:TYPE ASCIZ STRING
3280 013770 000424  BR 30044$ ;:GET OVER THE ASCIZ
3281 014042 012701 100000  ;:30045$: .ASCIZ <15><12>/TEST 22 - SLU INTERRUPT PRIORITY TEST/
3282 014046 077101 1001$: MOV #100000,R1
3283 014050 1000$: SOB R1,1001$ ; WAIT FOR LAST CHARACTER
3284 014050 122737 000001 001220  CMPB #APTENV,$ENV ;RUNNING IN APT MODE?
3285 014056 001003  BNE 100$ ;NO, GO DO TEST
3286 014060 005737 001206  TST $PASS ;FIRST PASS?
3287 014064 001172  BNE TST23 ;:IF APT AND NOT FIRST PASS, EXIT TEST
3288
3289 ; GET READY FOR INTERRUPTS
3290
3291 014066 012701 000060 100$: MOV #60,R1
3292 014066

```

TEST - INTERRUPT PRIORITY FOR SLU

```

3293 014072 012704 177560      MOV    #RCSR,R4      ;
3294 014076 012705 000002      MOV    #2,R5        ;
3295                               ;
3296 014102                10$:  MOV    (R1),-(SP)    ;SAVE PREVIOUS VECTOR ;MC
3297 014102 011146          MOV    4(R1),(SP)    ;"" "" "" ""
3298 014104 016146 000004      MOV    #8$, (R1)    ;STORE RECEIVER VECTOR
3299 014110 012711 014444      MOV    #6$,4(R1)    ;STORE TRANSMITTER VECTOR
3300 014114 012761 014436 000004  MOV    #340,2(R1)   ;AT PRIORITY 7
3301 014122 012761 000340 000002  MOV    #340,6(R1)   ;FOR RECEIVER AND TRANSMITTER
3302 014130 012761 000340 000006  MOV    #BIT02,4(R4) ;SET LOOP BACK MODE
3303 014136 052764 000004 000004  BIS    #BIT02,4(R4) ;DELAY FOR UNEXPECTED CHARACTERS
3304 014144 012701 000140      MOV    #140,R1      ;RECEIVER READY?
3305 014150 105714          1$:  TSTB   (R4)         ;IF YES, BRANCH
3306 014152 100401          BMI    2$           ;OTHERWISE, WAIT JUST IN CASE
3307 014154 077103          SOB    R1,1$       ;READ RECEIVER
3308 014156 005764 000002          2$:  TST    2(R4)
3309                               ;
3310                               ; SET PRIORITIES AND XMIT INTERRUPTS
3311                               ;
3312 014162 012702 000200      MOV    #200,R2      ;START WITH PRIORITY 3
3313                               ;
3314 014166 162702 000040          3$:  SUB    #40,R2      ;LOWER PRIORITY
3315 014172 005037 002320      CLR    TCOUNT     ;CLEAR TRANSMITTER COUNTER ;MC
3316 014176 005037 002316      CLR    RCOUNT     ;CLEAR RECEIVER COUNTER ;MC
3317                               ;
3318                               ; TRANSMITTER INTERRUPT HERE
3319                               ;
3320                               ;
3321 014202 106402          4$:  MTPS   R2          ;TRY TO DO AT LOWER PRIORITY
3322 014204 052764 000100 000004  BIS    #BIT06,4(R4) ;LOOP BACK & INTERRUPT ENABLE
3323 014212 004737 026772      JSR    PC,DELAY     ;WAIT DELAY FOR INTERRUPT ;MC
3324 014216 042764 000100 000004  BIC    #BIT06,4(R4) ;CLEAR INTERRUPT ENABLE
3325 014224 022737 000001 002320  CMP    #1,TCOUNT    ;ANY INTERRUPT HAPPENED ? ;MC
3326 014232 101041          BHI    5$           ;NO XMIT INTERRUPT ;MC
3327 014234 103450          BLO    200$        ;TOO MANY XMIT INTERRUPTS ;MC
3328                               ;
3329                               ;
3330                               ; RECEIVER INTERRUPT HERE
3331                               ;
3332                               ;
3333 014236 052714 000100          BIS    #BIT06,(R4)  ;SET RECEIVE INTERRUPT
3334 014242 012764 000000 000006  MOV    #NULL,6(R4) ;TRANSMIT NULL
3335 014250 004737 026772      JSR    PC,DELAY     ;WAIT DELAY FOR INTERRUPT ;MC
3336 014254 042714 000100          BIC    #BIT06,(R4)  ;CLEAR INTERRUPT ENABLE
3337 014260 022737 000001 002316  CMP    #1,RCOUNT    ;ONE INTERRUPT HAPPENED ? ;MC
3338 014266 101043          BHI    7$           ;NO RECEIVER INTERRUPTS ;MC
3339 014270 103452          BLO    201$        ;2 MANY RECEIVER INTERRUPTS ;MC
3340                               ;
3341                               ;
3342                               ; IF DONE GET OUT, IF NOT LOOP
3343                               ;
3344 014272 005764 000002          9$:  TST    2(R4)      ;READ RECEIVER BUFFER
3345 014276 005702          TST    R2          ;PRIORITY 0
3346 014300 001332          BNE    3$          ;IF NOT YET, CONTINUE
3347 014302 106427 000340          MTPS   #340        ;RAISE PRIORITY TO 7
3348 014306 005064 000004          CLR    4(R4)       ;CLEAR XCSR
3349 014312 012661 000004          MOV    (SP)+,4(R1) ;GET PREVIOUS VECTOR BACK

```

TFST - INTERRUPT PRIORITY FOR SLU

```

3350 014316 012611          MOV      (SP)+,(R1)          ; " " " "
3351 014320 012704 176500 20$:  MOV      #RCSR1,R4          ;
3352 014324 012701 000300    MOV      #300,R1          ;
3353 014330 005305          DEC      R5                ;
3354 014332 001263          BNE     10$                ;
3355 014334 000446          BR      TST23              ;;IF ERROR, EXIT TEST
3356
3357
3358      ;ERROR ROUTINES
3359      ;
3360
3361 014336 012661 000004 5$:  MOV      (SP)+,4(R1)        ;GET PREVIOUS VECTOR BACK
3362 014342 012611          MOV      (SP)+,(R1)        ;" " " "
3363 014344 042764 000104 000004 BIC     #BIT02!BIT06,4(R4) ;CLEAR LOOP BACK BIT
3364 014352 104070          ERROR   +70              ;NO XMIT INTERRUPTS
3365 014354 000436          BR      TST23              ;;IF ERROR, EXIT TEST
3366
3367 014356 012661 000004 200$: MOV      (SP)+,4(R1)        ;GET PREVIOUS VECTOR BACK
3368 014362 012611          MOV      (SP)+,(R1)        ;" " " "
3369 014364 042764 000104 000004 BIC     #BIT02!BIT06,4(R4) ;CLEAR LOOP BACK MODE BIT
3370 014372 104074          ERROR   +74              ;2 MANY XMIT INTERRUPTS ;MC
3371 014374 000426          BR      TST23              ;;IF ERROR, EXIT TEST
3372
3373 014376 012661 000004 7$:  MOV      (SP)+,4(R1)        ;GET PREVIOUS VECTOR BACK
3374 014402 012611          MOV      (SP)+,(R1)        ;" " " "
3375 014404 042764 000104 000004 BIC     #BIT02!BIT06,4(R4) ;CLEAR LOOP BACK MODE BIT
3376 014412 104071          ERROR   +71              ;NO RECEIVE INTERRUPTS
3377 014414 000416          BR      TST23              ;;IF ERROR, EXIT TEST
3378
3379 014416 012661 000004 201$: MOV      (SP)+,4(R1)        ;GET PREVIOUS VECTOR BACK
3380 014422 012611          MOV      (SP)+,(R1)        ;" " " "
3381 014424 042764 000104 000004 BIC     #BIT02!BIT06,4(R4) ;CLEAR LOOP BACK MODE BIT
3382 014432 104075          ERROR   +75              ;2 MANY RECEIVER INTERRUPTS
3383 014434 000406          BR      TST23              ;;IF ERROR, EXIT TEST
3384
3385      ;INTERRUPT SERVICE ROUTINES
3386      ;
3387
3388
3389 014436 005237 002320 6$:  INC      TCOUNT          ;INCREMENT TRANS. COUNTER ;MC
3390 014442 000002          RTI                    ;RETURN FROM XMIT INTERRUPT ;MC
3391
3392 014444 005237 002316 8$:  INC      RCOUNT          ;INCREMENT RECEIVER COUNTER ;MC
3393 014450 000002          RTI                    ;RETURN FROM RECEIVER INTERRUPT
3394
3395

```

TEST BREAK CONDITION

```

3397 .SBTTL TEST - BREAK CONDITION
3398 ;
3399 ; SLU #1 IS TESTED SINCE SLU #0 IS BEING USED AS THE CONSOLE
3400 ;
3401 ; CHECK THAT SENDING BREAK CAUSES FRAMING ERROR.
3402 ;
3403 ;RCSR <15> ERROR
3404 ; <13> FRAMING ERROR
3405 ; <11> RECEIVED BREAK
3406 ;
3407 ;XCSR <00> TRANSMIT BREAK
3408 ;
3409 ;ROUTINE TEST
3410 ;.. LET XCSR<02>=#1
3411 ;.. LET XCSR<00>=#1
3412 ;.. WAIT FOR RCSR<07>=#1
3413 ;.. IF RBUF<15!13!11> NE #1 THEN
3414 ;.. ERROR (ERROR, FRAMING ERROR, RECEIVE BREAK NE 1)
3415 ;.. ENDIF
3416 ;.. LET XCSR<00>=#0
3417 ;.. IF XCSR<00> NE #0 THEN
3418 ;.. ERROR (XCSR<00> DOES NOT GO LOW)
3419 ;.. ENDIF
3420 ;.. WAIT FOR XCSR<07>=#1
3421 ;.. LET XBUF=#NULL (SEND NULL CHARACTER TO SEE ERROR CLEARED)
3422 ;.. WAIT FOR RCSR<07>=#1
3423 ;.. IF RBUF<15!13!11> NE #0 THEN
3424 ;.. ERROR
3425 ;.. ENDIF
3426 ;.. LET XCSR<00>=#1
3427 ;.. EXECUTE "RESET"
3428 ;.. IF XCSR<00> NE #0 THEN
3429 ;.. ERROR
3430 ;.. ENDIF
3431 ;.. LET XCSR<02>=#0
3432 ;ENDROUTINE
3433 ;
3434 ;*****
3435 014452 000004 ;ST23: SCOPE
3436 014454 032777 010000 164456 BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
3437 014462 001426 BEQ 1000$ ; THEN TYPE TEST TRACE
3438 014540 104401 014472 TYPE 30047$ ;:TYPE ASCIZ STRING
3439 014470 000423 BR 30046$ ;:GET OVER THE ASCIZ
3440 ;:30047$: .ASCIZ <15><12>/TEST 23 - SLU BREAK CONDITION TEST/
3441 ;:30046$:
3442 ;:1000$:
3443 ;
3444 ; SEND BREAK AND CHECK ERROR BITS IN RBUF
3445 ;
3446 ;
3447 ;
3448 ;
3449 014540 052737 000004 176504 1$: BIS #BIT02,XCSR1 ; TRANSMIT IN LOOP BACK
3450 014546 052737 000001 176504 BIS #BIT00,XCSR1 ; SET SEND BREAK BIT
3451 014554 032737 000001 176504 BIT #BIT00,XCSR1 ; GOT SET OK?
3452 014562 001001 BNE 2$ ; IF YES, BRANCH
3453 014564 104027 ERROR +27 ; WRITING 1 TO XCSR<0>
3454 014566 012701 000100 2$: MOV #100,R1 ; STALL DELAY

```

TEST - BREAK CONDITION

```

3450 014572 105737 176500      4$:  TSTB  RCSR1      ;RECEIVER READY?
3451 014576 100401              BMI    5$          ;IF YES, BRANCH
3452 014600 077104              SOB    R1,4$      ;WAIT JUST IN CASE OF A CHARACTER
3453 014602 005737 176502      5$:  TST   RBUF1     ;READ A CHARACTER
3454 014606 052737 000001 176504  BIS   #BIT00,XCSR1 ;TRANSMIT BREAK
3455 014614 012701 001000      MOV   #1000,R1    ;ANOTHER DELAY TO GET BREAK
3456 014620 077101              SOB    R1,6$      ;WAIT A WHILE
3457 014622 105737 176500      6$:  TSTB  RCSR1     ;RECEIVER READY?
3458 014626 100375              BPL   7$          ;IF NOT, WAIT
3459 014630 013737 176502 001126  MOV   RBUF1,$BDDAT ;STORE WHATEVER RECEIVED
3460 014636 022737 124000 001126  CMP   #BIT15!BIT13!BIT11,$BDDAT ;ALL ERROR BITS SET?
3461 014644 001405              BEQ   8$          ;IF YES, BRANCH
3462 014646 042737 000004 176504  BIC   #BIT02,XCSR1 ;RESET TO ENABLE SLU
3463 014654 104025              ERROR +25         ;BREAK DOES NOT CAUSE ERRORS
3464 014656 000443              BR    TST24      ;;EXIT
3465 014660 042737 000001 176504  8$:  BIC   #BIT00,XCSR1 ;CLEAR TRANSMIT BREAK
3466 014666 032737 000001 176504  BIT   #BIT00,XCSR1 ;GOT CLEARED OK?
3467 014674 001405              BEQ   9$          ;IF YES, BRANCH
3468 014676 042737 000004 176504  BIC   #BIT02,XCSR1 ;RESET TO ENABLE SLU
3469 014704 104027              ERROR +27         ;ERROR WRITING 0 TO XCSR<0>
3470 014706 000427              BR    TST24      ;;EXIT
3471
3472      ; CHECK THAT BREAK CONDITION IS CLEARED
3473
3474 014710      9$:  MOV   SAVBR,BCSR ;RESTORE BCSR
3475 014710 105737 176504      10$: TSTB  XCSR1     ;XMIT READY?
3476 014714 100375              BPL   10$        ;IF NOT, WAIT
3477 014716 012737 000177 176506  MOV   #177,XBUF1 ;TRY TO TRANSMIT DELETE
3478 014724 105737 176500      11$: TSTB  RCSR1     ;RECEIVER READY
3479 014730 100375              BPL   11$        ;IF NOT, WAIT
3480 014732 013737 176502 001126  MOV   RBUF1,$BDDAT ;STORE RECEIVE BUFFER
3481 014740 032737 124000 001126  BIT   #BIT15!BIT13!BIT11,$BDDAT ;ERRORS CLEARED?
3482 014746 001404              BEQ   12$        ;IF YES, BRANCH
3483 014750 042737 000004 176504  BIC   #BIT02,XCSR1 ;RESET TO ENABLE SLU
3484 014756 104025              ERROR +25         ;BREAK NOT CLEARED ON NEXT CHARACTER
3485 014760 042737 000004 176504  12$: BIC   #BIT02,XCSR1 ;CLEAR LOOP BACK MODE
3486
3487

```

TEST - OVERRUN CONDITION

```

3489 .SBTTL TEST - OVERRUN CONDITION
3490 :CHECK OVERRUN CONDITION
3491 :
3492 :RCSR <14> OVERRUN ERROR
3493 :
3494 : SLU #1 IS TESTED SINCE SLU #0 IS BEING USED AS THE CONSOLE
3495 :
3496 :ROUTINE TEST
3497 :   LET XCSR<02>=#1 (LOOPBACK MODE)
3498 :   WAIT FOR XCSR<07>=#1
3499 :   LET XBUF=#252
3500 :   WAIT FOR XCSR<07>=#1
3501 :   LET XBUF=#125 (SEND THE 2ND W/O READING THE 1ST CHARACTER)
3502 :   WAIT FOR RCSR<07>=#1
3503 :   STALL FOR LOWEST BAUD RATE TO GET 2ND CHARACTER
3504 :   IF LOW BYTE OF RBUF NE #125 THEN
3505 :       ERROR (1ST CHARACTER WASN'T OVERRUN)
3506 :   ENDIF
3507 :   IF RBUF<15!14> NE #1 THEN
3508 :       ERROR (NO OVERRUN BIT SET)
3509 :   ENDIF
3510 :   WAIT FOR XCSR<07>=#1
3511 :   LET XBUF=#NULL
3512 :   WAIT FOR RCSR<07>=#1
3513 :   IF RBUF<15!14> NE #0 THEN
3514 :       ERROR (WASN'T CLEARED ON THE NEXT CHARACTER RECEIVED)
3515 :   ENDIF
3516 :   LET XCSR<02>=#0
3517 :ENDROUTINE
3518 :
3519 :

```

```

*****
3520 014766 000004 010000 164142 ST24: SCOPE
3521 014776 001432          BIT #BIT12,SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
3522 015000 104401 015006 BEQ 100$ ; THEN TYPE TEST TRACE
          TYPE #30049$ ;:TYPE ASCIZ STRING
          BR #30048$ ;:GET OVER THE ASCIZ
          ;:30049$: .ASCIZ <15><12>/TEST 24 - SLU OVERRUN CONDITION TEST/
          ;:30048$:
3523 015056 012701 100000 MOV #100000,R1
3524 015062 077101 1001$: SOB R1,1001$ ; WAIT FOR LAST CHARACTER
3525 015064 052737 000004 176504 100$: BIS #BIT02,XCSR1 ;SET LOOP BACK MODE
3526 015072 105737 176504 1$: TSTB XCSR1 ;READY TO TRANSMIT?
3527 015076 100375 BPL 1$ ;IF NOT, WAIT
3528 015100 012737 000021 176506 MOV #21,XBUF1 ;TRANSMIT A CHARACTER
3529 015106 105737 176500 2$: TSTB RCSR1 ;RECEIVE READY?
3530 015112 100375 BPL 2$ ;IF NOT, WAIT
3531 015114 105737 176504 3$: TSTB XCSR1 ;READY TO TRANSMIT?
3532 015120 100375 BPL 3$ ;IF NOT, WAIT
3533 015122 012737 000177 176506 MOV #177,XBUF1 ;TRANSMIT THE 2ND CHARACTER
3534 015130 012703 175000 MOV #175000,R3 ;STALL FOR THE 2ND CHARACTER $$$
3535 015134 077301 4$: SOB R3,4$ ;WAIT A WHILE
3536 015136 013737 176502 001126 MOV RBUF1,$BDDAT ;STORE RECEIVED DATA
3537 015144 012737 140177 001124 MOV #140177,$GDDAT ;EXPETED ATTERN
3538 015152 122737 000177 001126 CMPB #177,$BDDAT ;2ND CHARACTER RECEIVED?
3539 015160 001404 BEQ 5$ ;IF YES, BRANCH
3540 015162 042737 000004 176504 BIC #BIT02,XCSR1 ;RESET TO ENABLE SLU
3541 015170 104031 ERROR +31 ;2ND CHARACTER DIDN'T OVERRUN 1ST

```


TEST - OVERRUN CONDITION

```

3542 015172 122737 000300 001127 5$: CMPB #BIT7!BIT6,$BDDAT+1 ;OVERRUN ERROR BITS SET?
3543 015200 001404 BEQ 6$ ;IF YES, BRANCH
3544 015202 005037 176504 CLR XCSR1 ;RESET TO ENABLE SLU
3545 015206 104032 ERROR +32 ;OVERRUN DOES NOT SET ERRORS BITS
3546 015210 000426 BR TST25 ;;EXIT
3547
3548 ; SEND NEXT CHARACTER TO CLEAR OVERRUN CONDITIONS
3549
3550 015212 105737 176504 6$: TSTB XCSR1 ;TRANSMITTER READY?
3551 015216 100375 BPL 6$ ;IF NOT, BRANCH AND WAIT
3552 015220 012737 000000 176506 MOV #NULL,XBUF1 ;TRANSMIT NULL CHARACTER
3553 015226 105737 176500 7$: TSTB RCSR1 ;RECEIVER READY?
3554 015232 100375 BPL 7$ ;IF NOT, BRANCH AND WAIT
3555 015234 032737 140000 176502 BIT #BIT15!BIT14,RBUF1 ;ANY ERRORS SET?
3556 015242 001404 BEQ 8$ ;IF NOT, BRANCH
3557 015244 042737 000004 176504 BIC #BIT02,XCSR1 ;RESET TO ENABLE SLU
3558 015252 104033 ERROR +33 ;OVERRUN NOT CLEARED ON NEXT CHAR.
3559 015254 042737 000004 176504 8$: BIC #BIT02,XCSR1 ;CLEAR LOOP BACK MODE BIT
3560
3561 015262 SLEND: ;LAST SLU TEST
3562 015262 000401 BR TST25 ;;GO TO THE NEXT TEST
3563
3564 015264 000240 NOSLU: NOP ;WE SKIPPED ALL SLU TEST IN APT MODE ON MORE THAN 1 PASS
3565
3566

```

TEST - LED'S ON

```

3568 .SBTTL TEST - LED'S ON
3569 ;LED'S ON
3570 ;THIS TEST WILL INITIALIZE BDR TO CONTAIN A ROTATING PATTERN
3571 ;DISPLAYED IN LED'S.
3572 ;
3573 ;ROUTINE TEST
3574 ;. WHILE A KEY NOT RECEIVED FROM KEYBOARD DO
3575 ;. STALL ALLOWING TIME TO SEE PATTERN
3576 ;. ROTATE LEFT TO LIGHT UP NEXT LED'S
3577 ;. ENDDO
3578 ;ENDROUTINE
3579
3580
3581
3582
3583
3584 015266 000004
3585 015270 032777 010000 163642
3586 015276 001423
3587 015300 104401 015306
3588 015304 000420
3589 015346
3590 015346 005005
3591 015350 032737 000001 000052
3592 015356 001413
3593 015360 005737 001206
3594 015364 001010
3595 015366 122737 000001 001220
3596 015374 001404
3597 015376 005105
3598 015400 012737 000005 002314
3599 015406 012704 000020
3600 015412 012701 000000
3601 015416 110137 177520
3602 015422 012703 000004
3603 015426 012702 177777
3604 015432 077201
3605 015434 077304
3606 015436 000241
3607
3608
3609
3610 015440
3611 015440 005201
3612 015442 077413
3613 015444 005705
3614 015446 001411
3615 015450 104401 001170
3616 015454 005337 002314
3617 015460 001352
3618
3619
3620 015462 005737 177562

```

```

*****
TST25: SCOPE
BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
BEQ 1000$ ; THEN TYPE TEST TRACE
TYPE 30051$ ;;TYPE ASCIZ STRING
BR 30050$ ;;GET OVER THE ASCIZ
30051$: .ASCIZ <15><12>/TEST 25 - KDJ11-DA LED TESTS/
30050$:
1000$:
CLR R5 ;FLAG IN NO INTERRUPT MODE
BIT #BIT00,@52 ;IF RUNNING IN CHAIN MODE
BEQ 1$ ;SKIP PRINTOUTS
TST $PASS ;1ST PASS?
BNE 1$ ;IF NOT, SKIP PRINTOUTS
CMPB #APTENV,$ENV ;APT MODE?
BEQ 1$ ;YES, SKIP PRINTOUT S
COM R5 ;CLEAR FLAG IN INTERRUPT MODE
MOV #5,LEDCNT ;DO PATTERN 5 TIMES ;MC
1$: MOV #20,R4 ;FOR EACH LOOP
MOV #0,R1 ;START WITH 1
2$: MOVB R1,@#NATREG ;TURN OFF FIRST LED
MOV #4,R3 ;STALL DELAY
3$: MOV #177777,R2 ;STALL DELAY
4$: SOB R2,4$ ;WAIT A WHILE
SOB R3,3$ ;WAIT A WHILE
;
7$: INC R1 ; 1.....19 OCTAL ;MC
SOB R4,2$ ; 20.....0 OCTAL ;MC
TST R5 ;RUNNING IN INTERACTIVE MODE?
BEQ 6$ ;IF NOT, EXIT
;
;REPEAT PATTERN 5 TIMES
5$: TST RBUF ;READ BUFFER

```

TEST - LED S ON

3621	015466	062706	000004		ADD	#4,SP					
3622	015472	112737	000000	177520	6\$:	MOVB	#0,NATREG				
3623	015500	000400			BR	TST26		::EXIT TEST			
3624											

:ADJUST STACK
:NO MORE

TEST - DIFFERENT LEVELS OF INTERRUPTS

```

3626 .SBTTL TEST - DIFFERENT LEVELS OF INTERRUPTS
3627 ;DIFFERENT LEVELS OF INTERRUPTS
3628 ;THIS TEST WILL PROGRAM Q22 BUS EXERCISER TO INTERRUPT AT DIFFERENT
3629 ;LEVELS. ARBITRATION BETWEEN DIFFERENT LEVELS OF INTERRUPTS AND
3630 ;PIRQ'S WILL BE TESTED.
3631 ;
3632 ;CHECK DIFFERENT LEVELS OF INTERRUPTS.
3633 ;ROUTINE TEST
3634 ;. SET VECTOR TO INTERRUPT DMA
3635 ;. FOR INTERRUPTS FROM 4 TO 7 DO
3636 ;. . ENABLE INTERRUPTS
3637 ;. . SET PRIORITY=INTERUPT
3638 ;. . IF INTERRUPT FLAG SET THEN
3639 ;. . . ERROR
3640 ;. . . ENDF
3641 ;. . . ENABLE INTERRUPTS
3642 ;. . . SET PRIORITY=INTERRUPT-1
3643 ;. . . IF INTERRUPT FLAG NOTSET THEN
3644 ;. . . . ERROR
3645 ;. . . ENDF
3646 ;. . . LET INTERRUPT_DMA=0
3647 ;. . . ENDDO
3648 ;ENDROUTINE
3649 ;
3650 ;ROUTINE INTERUPT DMA
3651 ;. LET INTERRUPT_FLAG=1
3652 ;RETURN
3653 ;ENDROUTINE
3654 ;
3655 ;*****
3656 015502 000004
3657 ;ST26: SCOPE
3658 ; BIT #BIT07,@#52 ;UFD MODE?
3659 ; SKIP NE,<IF SO, EXIT TEST>
3660 015504 005737 002274 ;TST CSR1 ;AT LEAST ONE Q22BE FOUND?
3661 015510 001530 ;BEQ TST27 ;IF NOT, EXIT TEST
3662 015512 032777 010000 163420 ;BIT #BIT12,@SWR ;IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
3663 015520 001444 ;BEQ 1000$ ;THEN TYPE TEST TRACE
3664 015522 104401 015530 ;TYPE 30053$ ;TYPE ASCIZ STRING
3665 015526 000441 ;BR 30052$ ;GET OVER THE ASCIZ
3666 ;30053$: .ASCIZ <15><12>/TEST 26 - ARBITRATION BETWEEN CPU PRIORITY AND Q BUS INTERRU
3667 ;30052$:
3668 ;1000$:
3669 ;
3670 ; SETUP INITIAL PRIORITY TO 7
3671 ;
3672 ; MOV CSR1,R3 ;DO FOR FIRST FOUND Q22BE
3673 ; MOV #5$,@VQBE1 ;POINT INTERRUPT VECTOR TO PROGRAM
3674 ; MOV #340,@VQPR1 ;AT PRIORITY 7
3675 ; MOV #Q22EN,R0 ;START WITH 4 FOR INTERRUPTS
3676 ; MOV #340,R1 ;LOW BOUNDARY FOR NO INTERRUPTS
3677 ;
3678 ; CHECK THAT INTERRUPTS DON'T HAPPEN AT PRIORITY HIGHER THAN BR
3679 ;
3680 ; 2$:
3681 ; 4$: MTPS #200 ;SET PRIORITY NOT TO INTERRUPT
3682 ; JSR PC,Q22INT ;ENABLE INTERRUPTS
3683 ; MOV (R0)+,@CSR2 ;CLEAR GO BIT

```

TEST - DIFFERENT LEVELS OF INTERRUPTS

```

3679 015676 000240      NOP
3680 015700 000240      NOP
3681 015702 000240      NOP
3682 015704 000403      BR      6$
3683 015706 104037      5$:  ERROR +37      ;IF NO INTERRUPT BRANCH
3684 015710 005726      TST    (SP)+      ;INTERRUPTS HAPPEN
3685 015712 005726      TST    (SP)+      ;RESTORE STACK
3686 015714
3687
3688      ; INTERRUPT AT ALL LEVELS
3689
3690 015714 012777 015762 164764 INQ22: MOV    #5$,@VQBE1      ;POINT INTERRUPT VECTOR TO PROGRAM
3691 015722 012777 000340 164360      MOV    #340,@VQPR1      ;AT PRIORITY 7
3692 015730 012700 003002      MOV    #Q22EN,R0      ;START WITH 4 FOR INTERRUPTS
3693
3694      ; CHECK THAT INTERRUPTS HAPPEN AT PRIORITY LOWER THAN BR
3695
3696 015734 106427 000140      4$:  MTPS  #140      ;SET PRIORITY TO INTERRUPT
3697 015740 004737 026210      JSR    PC,Q22INT      ;ENABLE INTERRUPTS
3698 015744 011077 164326      MOV    (R0),@CSR2    ;CLEAR GO BIT
3699 015750 000240      NOP
3700 015752 000240      NOP      ;WAIT A WHILE
3701 015754 000240      NOP
3702 015756 104037      ERROR +37
3703 015760 000402      BR      6$
3704 015762 005726      5$:  TST    (SP)+      ;INTERRUPTS DON'T HAPPEN
3705 015764 005726      TST    (SP)+      ;DON'T RESTORE STACK
3706 015766 106427 000340      6$:  MTPS  #340      ;RESTORE STACK
3707

```

;BACK TO 7

TEST ARBITRATION BETWEEN PIRQ'S AND INTERRUPTS

```

3709 .SBTTL TEST - ARBITRATION BETWEEN PIRQ'S AND INTERRUPTS
3710 ;CHECK PRIORITY ORDER BETWEEN PIRQ'S AND INTERRUPTS.
3711 ;ROUTINE TEST
3712 ;. IF UFD THEN
3713 ;. EXIT TEST
3714 ;. ENDF
3715 ;. DO FOR I FROM #6 DOWN TO #3
3716 ;. SET PRIORITY TO I
3717 ;. ENABLE INTERRUPT(I+1) AND PIRQ(I+1)
3718 ;. IF INTERRUPT(I+1) WAS BEFORE PIRQ(I+1) THEN
3719 ;. ERROR
3720 ;. ENDF
3721 ;. ENDDO
3722 ;ENDROUTINE
3723
3724 ;*****
3725 015772 000004 TST27: SCOPE
3726 ; BIT #BIT07,@#52 ;UFD MODE?
3727 ; SKIP NE,<IF SO, EXIT TEST>
3728 015774 005737 002274 TST CSR1 ;AT LEAST ONE Q22BE FOUND?
3729 016000 001521 BEQ TST30 ;;IF NOT, EXIT TEST
R 3729 016002 032777 010000 163130 BIT #BIT12,@SWR ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
3730 016010 001441 BEQ 1000$ ; THEN TYPE TEST TRACE
3731 016012 104401 016020 TYPE #30055$ ;;TYPE ASCIZ STRING
016016 000436 BR #30054$ ;;GET OVER THE ASCIZ
3732 016114 ;:30055$: .ASCIZ <15><12>/TEST 27 - ARBITRATION BETWEEN PIRQ S AND Q-BUS INTERRUPTS/
3733 016114 30054$:
3734 016122 012777 016214 164164 MOV #3$,@VQBE1 ;SETUP Q22BE VECTOR
3735 016130 012737 016224 000240 MOV #340,@VQPR1 ;AT PRIORITY 7
3736 016136 012737 000340 000242 MOV #4$,PIRQVEC ;SETUP PIRQ VECTOR
3737 016144 012700 003002 MOV #340,PIRQVEC+2 ;AT PRIORITY 7
3738 016150 012704 016242 MOV #Q22EN,P0 ;POINT THRU PRIORITIES FOR Q22BE
3739 016154 013703 002274 MOV #PIRQT,R4 ;POINTER THRU PIRQ'S
3740 016160 012702 000140 MOV CSR1,R3 ;DO FOR FIRST Q22BE
3741 016164 106427 000340 2$: MOV #140,R2 ;START WITH CPU PRIORITY AT 7
3742 016170 012437 177772 MTPS #340 ;RAISE PRIORITY TO 7
3743 016174 004737 026210 MOV (R4)+,PIRQ ;SET PRIORITY FOR PIRQ'S
3744 016200 012077 164072 JSR PC,Q22INT ;INITIALISE Q22BE TO INTERRUPT
3745 016204 106402 MOV (R0)+,@CSR2 ;SET DONE BIT
3746 016206 000240 MTPS R2 ;LOWER PRIORITY
3747 016210 000240 NOP
3748 016212 000240 NOP
3749 016214 104035 3$: ERROR +35 ;PIRQ'S DON'T TAKE OVER BIRQ S
3750 016216 005726 TST (SP)+ ;CLEAN UP STACK
3751 016220 005726 TST (SP)+
3752 016222 000402 BR 5$ ;BRANCH AROUND PIRQ INTERRUPT
3753 016224 005726 4$: TST (SP)+ ;CLEAN UP STACK
3754 016226 005726 TST (SP)+
3755 016230 005037 177772 5$: CLR PIRQ ;CLEAR ANY REQUESTS
3756 016234 005077 164036 CLR @CSR2 ;CLEAR JUST IN CASE
3757 016240 000401 BR TST30 ;;GO TO THE NEXT TEST
3758
3759 016242 010000 PIRQT: .WORD 10000 ;PIRQ S 4

```

TEST ARBITRATION BETWEEN PIRQ S AND INTERRUPTS

```

3761
3762 016244 000004          ;*****
3763 016246 005037 177572  †ST30: SCOPE
3764 016252 000137 030014          CLR    @#SRO          ;DISABLE MMU (DO NOT REMOVE !!!)
3765 016256 123727 001220 000001 VIREOP: CMPB  $ENV,#1          ; if not APT, don't worry about
3766 016264 001005          BNE    1$              ;
3767 016266 005737 003030          TST   CCHPAS          ; maintain cache routin pascnt
3768 016272 001402          BEQ   1$
3769 016274 005337 003030          DEC   CCHPAS
3770
3771
3772 016300 000205          1$:   rts             ; This VIREOP ROUTINE to provide common End of Pass exit point
3773

```

GLOBAL ERROR MESSAGES

3775					.SBTTL GLOBAL ERROR MESSAGES	
3776	016302				.SWTSEL:	
3777	016302	015	012	113	.ASCII <15><12>/KDJ11-DA CPU DIAGNOSTIC - COKDDAO/<15><12><12>	
	016305	104	112	061		
	016310	061	055	104		
	016313	101	040	103		
	016316	120	125	040		
	016321	104	111	101		
	016324	107	116	117		
	016327	123	124	111		
	016332	103	040	055		
	016335	040	103	117		
	016340	113	104	104		
	016343	101	060	015		
	016346	012	012			
3778	016350	123	127	111	.ASCII /SWITCH REGISTER SELECTION:/<12><12><15>	
	016353	124	103	110		
	016356	040	122	105		
	016361	107	111	123		
	016364	124	105	122		
	016367	040	123	105		
	016372	114	105	103		
	016375	124	111	117		
	016400	116	072	012		
	016403	012	015			
3779	016405	102	111	124	.ASCII /BIT NUMBER	USE/<12><15>
	016410	040	116	125		
	016413	105	102	105		
	016416	122	011	011		
	016421	011	011	125		
	016424	123	105	012		
	016427	015				
3780	016430	055	055	055	.ASCII /-----	-----/<12><15>
	016433	055	055	055		
	016436	055	055	055		
	016441	055	011	011		
	016444	011	055	055		
	016447	055	055	055		
	016452	055	055	055		
	016455	055	055	055		
	016460	055	055	055		
	016463	055	055	055		
	016466	055	055	012		
	016471	015				
3781	016472	011	061	065	.ASCII /	15 HALT ON ERROR/<12><15>
	016475	011	011	011		
	016500	110	101	114		
	016503	124	040	117		
	016506	116	040	105		
	016511	122	122	117		
	016514	122	012	015		
3782	016517	011	061	064	.ASCII /	14 LOOP ON PRESENT TEST/<12><15>
	016522	011	011	011		
	016525	114	117	117		
	016530	120	040	117		
	016533	116	040	120		
	016536	122	105	123		

GLOBAL ERROR MESSAGES

	016541	105	116	124		
	016544	040	124	105		
	016547	123	124	012		
	016552	015				
3783	016553	011	061	063	.ASCII /	13
	016556	011	011	011		
	016561	111	116	110		
	016564	111	102	111		
	016567	124	040	105		
	016572	122	122	117		
	016575	122	040	124		
	016600	131	120	105		
	016603	117	125	124		
	016606	123	012	015		
3784	016611	011	061	062	.ASCII /	12
	016614	011	011	011		
	016617	105	116	101		
	016622	102	114	105		
	016625	040	124	105		
	016630	123	124	040		
	016633	124	122	101		
	016636	103	111	116		
	016641	107	012	015		
3785	016644	011	061	061	.ASCII /	11
	016647	011	011	011		
	016652	111	116	110		
	016655	111	102	111		
	016660	124	040	111		
	016663	124	105	122		
	016666	101	124	111		
	016671	117	116	123		
	016674	012	015			
3786	016676	011	061	060	.ASCII /	10
	016701	011	011	011		
	016704	102	105	114		
	016707	114	040	117		
	016712	116	040	105		
	016715	122	122	117		
	016720	122	012	015		
3787	016723	011	040	071	.ASCII /	9
	016726	011	011	011		
	016731	114	117	117		
	016734	120	040	117		
	016737	116	040	105		
	016742	122	122	117		
	016745	122	012	015		
3788	016750	011	040	070	.ASCII /	8
	016753	011	011	011		
	016756	114	117	117		
	016761	120	040	117		
	016764	116	040	124		
	016767	105	123	124		
	016772	040	111	116		
	016775	040	123	127		
	017000	122	074	065		
	017003	055	060	076		
	017006	012	015			

GLOBAL ERROR MESSAGES

3789	017010	011	040	067	.ASCII /	7	INHIBIT THE CHECK PARITY TEST/<12><15>
	017013	011	011	011			
	017016	111	116	110			
	017021	111	102	111			
	017024	124	040	124			
	017027	110	105	040			
	017032	103	110	105			
	017035	103	113	040			
	017040	120	101	122			
	017043	111	124	131			
	017046	040	124	105			
	017051	123	124	012			
	017054	015					
3790	017055	011	040	066	.ASCII /	6	Not used/<12><15>
	017060	011	011	011			
	017063	116	157	164			
	017066	040	165	163			
	017071	145	144	012			
	017074	015					
3791	017075	011	065	055	.ASCIZ /	5-0	Subtest number to loop on (BIT 8)/<12><12><15>
	017100	050	011	011			
	017103	011	123	165			
	017106	142	164	145			
	017111	163	164	040			
	017114	156	165	155			
	017117	142	145	162			
	017122	040	164	157			
	017125	040	154	157			
	017130	157	160	040			
	017133	157	156	040			
	017136	050	102	111			
	017141	124	040	070			
	017144	051	012	012			
	017147	015	000				
3792							
3793	017151	102	101	123	EM1: .ASCIZ /BASIC INSTRUCTION SET ERROR/		
	017154	111	103	040			
	017157	111	116	123			
	017162	124	122	125			
	017165	103	124	111			
	017170	117	116	040			
	017173	123	105	124			
	017176	040	105	122			
	017201	122	117	122			
	017204	000					
3794	017205	115	115	125	EM2: .ASCIZ /MMU ERROR/		
	017210	040	105	122			
	017213	122	117	122			
	017216	000					
3795	017217	106	120	120	EM3: .ASCIZ /FPP ERROR/		
	017222	040	105	122			
	017225	122	117	122			
	017230	000					
3796	017231				EM51:		
3797	017231	103	110	105	EM54: .ASCIZ /CHECKSUM ERROR IN 16 BIT ROM /		
	017234	103	113	123			
	017237	125	115	040			

GLOBAL ERROR MESSAGES

	017242	105	122	122	
	017245	117	122	040	
	017250	111	116	040	
	017253	061	066	055	
	017256	102	111	124	
	017261	040	122	117	
	017264	115	040	000	
3798	017267	124	111	115	EM56: .ASCIZ /TIMEOUT READING LKS/
	017272	105	117	125	
	017275	124	040	122	
	017300	105	101	104	
	017303	111	116	107	
	017306	040	114	113	
	017311	123	000		
3799	017313	114	113	123	EM57: .ASCIZ /LKS<07> DOES NOT BECOME 1/
	017316	074	060	067	
	017321	076	040	104	
	017324	117	105	123	
	017327	040	116	117	
	017332	124	040	102	
	017335	105	103	117	
	017340	115	105	040	
	017343	061	000		
3800	017345	111	114	114	EM61: .ASCIZ /ILLEGAL LKS INTERRUPTS/
	017350	105	107	101	
	017353	114	040	114	
	017356	113	123	040	
	017361	111	116	124	
	017364	105	122	122	
	017367	125	120	124	
	017372	123	000		
3801	017374	114	113	123	EM63: .ASCIZ /LKS READY DOESN'T GO LOW/
	017377	040	122	105	
	017402	101	104	131	
	017405	040	104	117	
	017410	105	123	116	
	017413	047	124	040	
	017416	107	117	040	
	017421	114	117	127	
	017424	000			
3802	017425	127	122	117	EM64: .ASCIZ /WRONG NUMBER OF LKS INTERRUPTS/
	017430	116	107	040	
	017433	116	125	115	
	017436	102	105	122	
	017441	040	117	106	
	017444	040	114	113	
	017447	123	040	111	
	017452	116	124	105	
	017455	122	122	125	
	017460	120	124	123	
	017463	000			
3803	017464	114	113	123	EM65: .ASCIZ /LKS INTERRUPTS HAPPEN AT WRONG PRIORITY/
	017467	040	111	116	
	017472	124	105	122	
	017475	122	125	120	
	017500	124	123	040	
	017503	110	101	120	

GLOBAL ERROR MESSAGES

	017506	120	105	116	
	017511	040	101	124	
	017514	040	127	122	
	017517	117	116	107	
	017522	040	120	122	
	017525	111	117	122	
	017530	111	124	131	
	017533	000			
3804	017534	122	105	123	EM71: .ASCIZ /RESET DOESN'T CLEAR LKS<06>/
	017537	105	124	040	
	017542	104	117	105	
	017545	123	116	047	
	017550	124	040	103	
	017553	114	105	101	
	017556	122	040	114	
	017561	113	123	074	
	017564	060	066	076	
	017567	000			
3805	017570	124	111	115	EM72: .ASCIZ /TIMEOUT READING SLU REGISTERS/
	017573	105	117	125	
	017576	124	040	122	
	017601	105	101	104	
	017604	111	116	107	
	017607	040	123	114	
	017612	125	040	122	
	017615	105	107	111	
	017620	123	124	105	
	017623	122	123	000	
3806	017626	105	122	122	EM73: .ASCIZ /ERROR IN YMIT READY/
	017631	117	122	040	
	017634	111	116	040	
	017637	130	115	111	
	017642	124	040	122	
	017645	105	101	104	
	017650	131	000		
3807	017652	122	103	123	EM74: .ASCIZ /RCSR<7> DOESN'T BECOME 1/
	017655	122	074	067	
	017660	076	040	104	
	017663	117	105	123	
	017666	116	047	124	
	017671	040	102	105	
	017674	103	117	115	
	017677	105	040	061	
	017702	000			
3808	017703	127	122	117	EM75: .ASCIZ /WRONG CHARACTER RECEIVED/
	017706	116	107	040	
	017711	103	110	101	
	017714	122	101	103	
	017717	124	105	122	
	017722	040	122	105	
	017725	103	105	111	
	017730	126	105	104	
	017733	000			
3809	017734	122	103	123	EM76: .ASCIZ /RCSR<07> NOT CLEARED AFTER READING RBUF/
	017737	122	074	060	
	017742	067	076	040	
	017745	116	117	124	

GLOBAL ERROR MESSAGES

	017750	040	103	114	
	017753	105	101	122	
	017756	105	104	040	
	017761	101	106	124	
	017764	105	122	040	
	017767	122	105	101	
	017772	104	111	116	
	017775	107	040	122	
	020000	102	125	106	
	020003	000			
3810	020004	130	103	123	EM77: .ASCIZ /XCSR<07> NOT SET ON RESET/
	020007	122	074	060	
	020012	067	076	040	
	020015	116	117	124	
	020020	040	123	105	
	020023	124	040	117	
	020026	116	040	122	
	020031	105	123	105	
	020034	124	000		
3811	020036	122	103	123	EM100: .ASCIZ /RCSR<07> NOT CLEARED ON RESET/
	020041	122	074	060	
	020044	067	076	040	
	020047	116	117	124	
	020052	040	103	114	
	020055	105	101	122	
	020060	105	104	040	
	020063	117	116	040	
	020066	122	105	123	
	020071	105	124	000	
3812	020074	123	114	125	EM101: .ASCIZ /SLU INTERRUPTS HAPPEN AT 4/
	020077	040	111	116	
	020102	124	105	122	
	020105	122	125	120	
	020110	124	123	040	
	020113	110	101	120	
	020116	120	105	116	
	020121	040	101	124	
	020124	040	064	000	
3813	020127	122	105	123	EM102: .ASCIZ /RESET DOES NOT CLEAR PROPER BITS IN SLU REGISTERS/
	020132	105	124	040	
	020135	104	117	105	
	020140	123	040	116	
	020143	117	124	040	
	020146	103	114	105	
	020151	101	122	040	
	020154	120	122	117	
	020157	120	105	122	
	020162	040	102	111	
	020165	124	123	040	
	020170	111	116	040	
	020173	123	114	125	
	020176	040	122	105	
	020201	107	111	123	
	020204	124	105	122	
	020207	123	000		
3814	020211	124	122	101	EM103: .ASCIZ /TRANSMIT INTERRUPT DOES NOT CLEAR XCSR<07>/
	020214	116	123	115	

GLOBAL ERROR MESSAGES

	020217	111	124	040	
	020222	111	116	124	
	020225	105	122	122	
	020230	125	120	124	
	020233	040	104	117	
	020236	105	123	040	
	020241	116	117	124	
	020244	040	103	114	
	020247	105	101	122	
	020252	040	130	103	
	020255	123	122	074	
	020260	060	067	076	
3815	020263	000			
	020264	122	105	103	EM104: .ASCIZ /RECEIVE INTERRUPTS DON'T CLEAR RCSR<07>/
	020267	105	111	126	
	020272	105	040	111	
	020275	116	124	105	
	020300	122	122	125	
	020303	120	124	123	
	020306	040	104	117	
	020311	116	047	124	
	020314	040	103	114	
	020317	105	101	122	
	020322	040	122	103	
	020325	123	122	074	
	020330	060	067	076	
	020333	000			
3816	020334	102	122	105	EM105: .ASCIZ /BREAK CONDITION DOES NOT SET RBUF PROPERLY/
	020337	101	113	040	
	020342	103	117	116	
	020345	104	111	124	
	020350	111	117	116	
	020353	040	104	117	
	020356	105	123	040	
	020361	116	117	124	
	020364	040	123	105	
	020367	124	040	122	
	020372	102	125	106	
	020375	040	120	122	
	020400	117	120	105	
	020403	122	114	131	
	020406	000			
3817	020407	122	102	125	EM106: .ASCIZ /RBUF <15-11> WASN'T CLEARED ON NEXT CHARACTER/
	020412	106	040	074	
	020415	061	065	055	
	020420	061	061	076	
	020423	040	127	101	
	020426	123	116	047	
	020431	124	040	103	
	020434	114	105	101	
	020437	122	105	104	
	020442	040	117	116	
	020445	040	116	105	
	020450	130	124	040	
	020453	103	110	101	
	020456	122	101	103	
	020461	124	105	122	

GLOBAL ERROR MESSAGES

3818	020464	000			
	020465	123	114	125	EM107: .ASCIZ /SLU INTERRUPTS DON'T HAPPEN/
	020470	040	111	116	
	020473	124	105	122	
	020476	122	125	120	
	020501	124	123	040	
	020504	104	117	116	
	020507	047	124	040	
	020512	110	101	120	
	020515	120	105	116	
	020520	000			
3819	020521	105	122	122	EM110: .ASCIZ /ERROR IN WRITING TO SLU REGISTERS/
	020524	117	122	040	
	020527	111	116	040	
	020532	127	122	111	
	020535	124	111	116	
	020540	107	040	124	
	020543	117	040	123	
	020546	114	125	040	
	020551	122	105	107	
	020554	111	123	124	
	020557	105	122	123	
	020562	000			
3820	020563	106	111	122	EM111: .ASCIZ /FIRST CHARACTER WAS NOT OVERRUN BY THE SECOND/
	020566	123	124	040	
	020571	103	110	101	
	020574	122	101	103	
	020577	124	105	122	
	020602	040	127	101	
	020605	123	040	116	
	020610	117	124	040	
	020613	117	126	105	
	020616	122	122	125	
	020621	116	040	102	
	020624	131	040	124	
	020627	110	105	040	
	020632	123	105	103	
	020635	117	116	104	
	020640	000			
3821	020641	117	126	105	EM112: .ASCIZ /OVERRUN CONDITION DOES NOT SET PROPER BITS IN RBUF/
	020644	122	122	125	
	020647	116	040	103	
	020652	117	116	104	
	020655	111	124	111	
	020660	117	116	040	
	020663	104	117	105	
	020666	123	040	116	
	020671	117	124	040	
	020674	123	105	124	
	020677	040	120	122	
	020702	117	120	105	
	020705	122	040	102	
	020710	111	124	123	
	020713	040	111	116	
	020716	040	122	102	
	020721	125	106	000	
3822	020724	117	126	105	EM113: .ASCIZ /OVERRUN BITS WERE NOT CLEARED ON THE NEXT CHARACTER/

GLOBAL ERROR MESSAGES

	020727	122	122	125	
	020732	116	040	102	
	020735	111	124	123	
	020740	040	127	105	
	020743	122	105	040	
	020746	116	117	124	
	020751	040	103	114	
	020754	105	101	122	
	020757	105	104	040	
	020762	117	116	040	
	020765	124	110	105	
	020770	040	116	105	
	020773	130	124	040	
	020776	103	110	101	
	021001	122	101	103	
	021004	124	105	122	
	021007	000			
3823	021010	105	122	122	EM114: .ASCIZ \ERROR ON XCSR<2>\
	021013	117	122	040	
	021016	117	116	040	
	021021	130	103	123	
	021024	122	074	062	
	021027	076	000		
3824	021031	105	122	122	EM123: .ASCIZ /ERROR IN Q22BE DMA CYCLES/
	021034	117	122	040	
	021037	111	116	040	
	021042	121	062	062	
	021045	102	105	040	
	021050	104	115	101	
	021053	040	103	131	
	021056	103	114	105	
	021061	123	000		
3825	021063	120	111	122	EM124: .ASCIZ /PIRQ INTERRUPTS DON'T TAKE PRIORITY OVER Q BUS INTERRUPTS/
	021066	121	040	111	
	021071	116	124	105	
	021074	122	122	125	
	021077	120	124	123	
	021102	040	104	117	
	021105	116	047	124	
	021110	040	124	101	
	021113	113	105	040	
	021116	120	122	111	
	021121	117	122	111	
	021124	124	131	040	
	021127	117	126	105	
	021132	122	040	121	
	021135	040	102	125	
	021140	123	040	111	
	021143	116	124	105	
	021146	122	122	125	
	021151	120	124	123	
	021154	000			
3826	021155	116	117	040	EM125: .ASCIZ /NO POWER DOWN TRAP TO 24 OCCUR/
	021160	120	117	127	
	021163	105	122	040	
	021166	104	117	127	
	021171	116	040	124	

GLOBAL ERROR MESSAGES

	021174	122	101	120	
	021177	040	124	117	
	021202	040	062	064	
	021205	040	117	103	
	021210	103	125	122	
	021213	000			
3827	021214	105	122	122	EM126: .ASCIZ /ERROR DOING Q22BE INTERRUPTS/
	021217	117	122	040	
	021222	104	117	111	
	021225	116	107	040	
	021230	121	062	062	
	021233	102	105	040	
	021236	111	116	124	
	021241	105	122	122	
	021244	125	120	124	
	021247	123	000		
3828	021251	105	122	122	EM127: .ASCIZ /ERROR IN OPERATION OF PMG COUNTER/
	021254	117	122	040	
	021257	111	116	040	
	021262	117	120	105	
	021265	122	101	124	
	021270	111	117	116	
	021273	040	117	106	
	021276	040	120	115	
	021301	107	040	103	
	021304	117	125	116	
	021307	124	105	122	
	021312	000			
3829	021313	125	116	105	EM130: .ASCIZ /UNEXPECTED TRAP TO 4/
	021316	130	120	105	
	021321	103	124	105	
	021324	104	040	124	
	021327	122	101	120	
	021332	040	124	117	
	021335	040	064	000	
3830	021340	105	122	122	EM131: .ASCIZ /ERROR WRITING TO LKS<6>/
	021343	117	122	040	
	021346	127	122	111	
	021351	124	111	116	
	021354	107	040	124	
	021357	117	040	114	
	021362	113	123	074	
	021365	066	076	000	
3831	021370	105	122	122	EM132: .ASCIZ /ERROR IN MAINTENANCE REGISTER/
	021373	117	122	040	
	021376	111	116	040	
	021401	115	101	111	
	021404	116	124	105	
	021407	116	101	116	
	021412	103	105	040	
	021415	122	105	107	
	021420	111	123	124	
	021423	105	122	000	
3832	021426	105	122	122	EM135: .ASCIZ /ERROR IN THE MEMORY DATA PATH/
	021431	117	122	040	
	021434	111	116	040	
	021437	124	110	105	

GLOBAL ERROR MESSAGES

	021442	040	115	105	
	021445	115	117	122	
	021450	131	040	104	
	021453	101	124	101	
	021456	040	120	101	
	021461	124	110	000	
3833	021464	124	111	115	EM136: .ASCIZ /TIMED OUT IN ACCESSING LOCATION 0/
	021467	105	104	040	
	021472	117	125	124	
	021475	040	111	116	
	021500	040	101	103	
	021503	103	105	123	
	021506	123	111	116	
	021511	107	040	114	
	021514	117	103	101	
	021517	124	111	117	
	021522	116	040	060	
	021525	000			
3834	021526	124	111	115	EM137: .ASCIZ /TIMED OUT IN TRYING TO ACCESS MEMGRY/
	021531	105	104	040	
	021534	117	125	124	
	021537	040	111	116	
	021542	040	124	122	
	021545	131	111	116	
	021550	107	040	124	
	021553	117	040	101	
	021556	103	103	105	
	021561	123	123	040	
	021564	115	105	115	
	021567	117	122	131	
	021572	000			
3835	021573	105	122	122	EM140: .ASCIZ /ERROR IN FUNCTIONAL REVISION BITS ON THE NATIVE REGISTER/
	021576	117	122	040	
	021601	111	116	040	
	021604	106	125	116	
	021607	103	124	111	
	021612	117	116	101	
	021615	114	040	122	
	021620	105	126	111	
	021623	123	111	117	
	021626	116	040	102	
	021631	111	124	123	
	021634	040	117	116	
	021637	040	124	110	
	021642	105	040	116	
	021645	101	124	111	
	021650	126	105	040	
	021653	122	105	107	
	021656	111	123	124	
3836	021661	105	122	000	
	021664	105	122	122	EM141: .ASCIZ /ERROR IN THE INDICATOR BITS ON THE NATIVE REGISTER/
	021667	117	122	040	
	021672	111	116	040	
	021675	124	110	105	
	021700	040	111	116	
	021703	104	111	103	
	021706	101	124	117	

GLOBAL ERROR MESSAGES

	021711	122	040	102	
	021714	111	124	123	
	021717	040	117	116	
	021722	040	124	110	
	021725	105	040	116	
	021730	101	124	111	
	021733	126	105	040	
	021736	122	105	107	
	021741	111	123	124	
	021744	105	122	000	
3837	021747	105	122	122	EM142: .ASCIZ /ERROR IN THE BOOT SELECT SWITCHES ON THE NATIVE REGISTER/
	021752	117	122	040	
	021755	111	116	040	
	021760	124	110	105	
	021763	040	102	117	
	021766	117	124	040	
	021771	123	105	114	
	021774	105	103	124	
	021777	040	123	127	
	022002	111	124	103	
	022005	110	105	123	
	022010	040	117	116	
	022013	040	124	110	
	022016	105	040	116	
	022021	101	124	111	
	022024	126	105	040	
	022027	122	105	107	
	022032	111	123	124	
	022035	105	122	000	
3838	022040	124	111	115	EM143: .ASCIZ /TIMED OUT IN ACCESS TO THE NATIVE REGISTER/
	022043	105	104	040	
	022046	117	125	124	
	022051	040	111	116	
	022054	040	101	103	
	022057	103	105	123	
	022062	123	040	124	
	022065	117	040	124	
	022070	110	105	040	
	022073	116	101	124	
	022076	111	126	105	
	022101	040	122	105	
	022104	107	111	123	
	022107	124	105	122	
	022112	000			
3839	022113	102	111	124	EM144: .ASCIZ /BIT 7 IN LTC IS NOT TOGGLING BETWEEN 0 AND 1/
	022116	040	067	040	
	022121	111	116	040	
	022124	114	124	103	
	022127	040	111	123	
	022132	040	116	117	
	022135	124	040	124	
	022140	117	107	107	
	022143	114	111	116	
	022146	107	040	102	
	022151	105	124	127	
	022154	105	105	116	
	022157	040	060	040	

GLOBAL ERROR MESSAGES

	022162	101	116	104	
	022165	040	061	000	
3840	022170	124	111	115	EM145: .ASCIZ /TIMEOUT IN ACCESSING THE LKS REGISTER/
	022173	105	117	125	
	022176	124	040	111	
	022201	116	040	101	
	022204	103	103	105	
	022207	123	123	111	
	022212	116	107	040	
	022215	124	110	105	
	022220	040	114	113	
	022223	123	040	122	
	022226	105	107	111	
	022231	123	124	105	
	022234	122	000		
3841	022236	105	122	122	EM146: .ASCIZ /ERROR IN STUCK AT ZERO BITS ON MER/
	022241	117	122	040	
	022244	111	116	040	
	022247	123	124	125	
	022252	103	113	040	
	022255	101	124	040	
	022260	132	105	122	
	022263	117	040	102	
	022266	111	124	123	
	022271	040	117	116	
	022274	040	115	105	
	022277	122	000		
3842	022301	103	117	125	EM147: .ASCIZ \COULD NOT SET ONE OF THE R/W BITS ON THE MER\
	022304	114	104	040	
	022307	116	117	124	
	022312	040	123	105	
	022315	124	040	117	
	022320	116	105	040	
	022323	117	106	040	
	022326	124	110	105	
	022331	040	122	057	
	022334	127	040	102	
	022337	111	124	123	
	022342	040	117	116	
	022345	040	124	110	
	022350	105	040	115	
	022353	105	122	000	
3843	022356	102	111	124	EM150: .ASCIZ /BITS 0,2,14,15 ON THE MER DID NOT CLEAR ON RESET/
	022361	123	040	060	
	022364	054	062	054	
	022367	061	064	054	
	022372	061	065	040	
	022375	117	116	040	
	022400	124	110	105	
	022403	040	115	105	
	022406	122	040	104	
	022411	111	104	040	
	022414	116	117	124	
	022417	040	103	114	
	022422	105	101	122	
	022425	040	117	116	
	022430	040	122	105	

GLOBAL ERROR MESSAGES

	022433	123	105	124	
	022436	000			
3844	022437	124	111	115	EM151: .ASCIZ /TIMEOUT IN ACCESSING THE MER REGISTER/
	022442	105	117	125	
	022445	124	040	111	
	022450	116	040	101	
	022453	103	103	105	
	022456	123	123	111	
	022461	116	107	040	
	022464	124	110	105	
	022467	040	115	105	
	022472	122	040	122	
	022475	105	107	111	
	022500	123	124	105	
	022503	122	000		
3845	022505	105	122	122	EM152: .ASCIZ /ERROR IN THE DATA SHORTS AND STUCK AT MEMORY TEST/
	022510	117	122	040	
	022513	111	116	040	
	022516	124	110	105	
	022521	040	104	101	
	022524	124	101	040	
	022527	123	110	117	
	022532	122	124	123	
	022535	040	101	116	
	022540	104	040	123	
	022543	124	125	103	
	022546	113	040	101	
	022551	124	040	115	
	022554	105	115	117	
	022557	122	131	040	
	022562	124	105	123	
	022565	124	000		
3846	022567	120	101	122	EM153: .ASCIZ /PARITY ABORT OCCURED WITH PARITY ERROR DISABLED/
	022572	111	124	131	
	022575	040	101	102	
	022600	117	122	124	
	022603	040	117	103	
	022606	103	125	122	
	022611	105	104	040	
	022614	127	111	124	
	022617	110	040	120	
	022622	101	122	111	
	022625	124	131	040	
	022630	105	122	122	
	022633	117	122	040	
	022636	104	111	123	
	022641	101	102	114	
	022644	105	104	000	
3847	022647	120	101	122	EM154: .ASCIZ /PARITY ABORT DID NOT OCCUR WITH PARITY ERROR ENABLED/
	022652	111	124	131	
	022655	040	101	102	
	022660	117	122	124	
	022663	040	104	111	
	022666	104	040	116	
	022671	117	124	040	
	022674	117	103	103	
	022677	125	122	040	

GLOBAL ERROR MESSAGES

	022702	127	111	124	
	022705	110	040	120	
	022710	101	122	111	
	022713	124	131	040	
	022716	105	122	122	
	022721	117	122	040	
	022724	105	116	101	
	022727	102	114	105	
	022732	104	000		
3848	022734	115	105	122	EM155: .ASCIZ /MER DIDN'T HAVE CORRECT ADDRESS BITS 11-17/
	022737	040	104	111	
	022742	104	116	047	
	022745	124	040	110	
	022750	101	126	105	
	022753	040	103	117	
	022756	122	122	105	
	022761	103	124	040	
	022764	101	104	104	
	022767	122	105	123	
	022772	123	040	102	
	022775	111	124	123	
	023000	040	061	061	
	023003	055	061	067	
	023006	000			
3849	023007	115	105	122	EM156: .ASCIZ /MER READ EXTENDED ADDRESS BITS HAS FAILED/
	023012	040	122	105	
	023015	101	104	040	
	023020	105	130	124	
	023023	105	116	104	
	023026	105	104	040	
	023031	101	104	104	
	023034	122	105	123	
	023037	123	040	102	
	023042	111	124	123	
	023045	040	110	101	
	023050	123	040	106	
	023053	101	111	114	
	023056	105	104	000	
3850	023061	105	122	122	EM157: .ASCIZ /ERROR IN THE QUICK VERIFY MEMORY TEST/
	023064	117	122	040	
	023067	111	116	040	
	023072	124	110	105	
	023075	040	121	125	
	023100	111	103	113	
	023103	040	126	105	
	023106	122	111	106	
	023111	131	040	115	
	023114	105	115	117	
	023117	122	131	040	
	023122	124	105	123	
	023125	124	000		
3851	023127	105	122	122	EM160: .ASCIZ /ERROR IN RCSR <6>/
	023132	117	122	040	
	023135	111	116	040	
	023140	122	103	123	
	023143	122	040	074	
	023146	066	076	000	

GLOBAL ERROR MESSAGES

3852	023151	116	117	040	EM161: .ASCIZ /NO XMIT INTERRUPTS HAVE OCCURED/
	023154	130	115	111	
	023157	124	040	111	
	023162	116	124	105	
	023165	122	122	125	
	023170	120	124	123	
	023173	040	110	101	
	023176	126	105	040	
	023201	117	103	103	
	023204	125	122	105	
	023207	104	000		
3853	023211	116	117	040	EM162: .ASCIZ /NO RECIEVE INTERRUPTS HAVE OCCURED/
	023214	122	105	103	
	023217	111	105	126	
	023222	105	040	111	
	023225	116	124	105	
	023230	122	122	125	
	023233	120	124	123	
	023236	040	110	101	
	023241	126	105	040	
	023244	117	103	103	
	023247	125	122	105	
	023252	104	000		
3854	023254	125	116	105	EM163: .ASCIZ /UNEXPECTED PARITY ABORT HAS OCCURED/
	023257	130	120	105	
	023262	103	124	105	
	023265	104	040	120	
	023270	101	122	111	
	023273	124	131	040	
	023276	101	102	117	
	023301	122	124	040	
	023304	110	101	123	
	023307	040	117	103	
	023312	103	125	122	
	023315	105	104	000	
3855	023320	115	105	122	EM164: .ASCIZ /MER DIDN'T HAVE CORRECT ADDRESS BITS# 0 AND 15/
	023323	040	104	111	
	023326	104	116	047	
	023331	124	040	110	
	023334	101	126	105	
	023337	040	103	117	
	023342	122	122	105	
	023345	103	124	040	
	023350	101	104	104	
	023353	122	105	123	
	023356	123	040	102	
	023361	111	124	123	
	023364	043	040	060	
	023367	040	101	116	
	023372	104	040	061	
	023375	065	000		
3856	023377	124	117	117	EM165: .ASCIZ /TOO MANY TRANSIVER INTERRUPTS HAPPENED/
	023402	040	115	101	
	023405	116	131	040	
	023410	124	122	101	
	023413	116	123	111	
	023416	126	105	122	

GLOBAL ERROR MESSAGES

	023421	040	111	116	
	023424	124	105	122	
	023427	122	125	120	
	023432	124	123	040	
	023435	110	101	120	
	023440	120	105	116	
	023443	105	104	000	
3857	023446	124	117	117	EM166: .ASCIZ /TOO MANY RECEIVER INTERRUPTS HAPPENED/
	023451	040	115	101	
	023454	116	131	040	
	023457	122	105	103	
	023462	105	111	126	
	023465	105	122	040	
	023470	111	116	124	
	023473	105	122	122	
	023476	125	120	124	
	023501	123	040	110	
	023504	101	120	120	
	023507	105	116	105	
	023512	104	000		
3858					
3859	023514	040	124	105	DH1: .ASCII / TEST ERROR/<15><12>
	023517	123	124	011	
	023522	105	122	122	
	023525	117	122	015	
	023530	012			
3860	023531	040	040	043	.ASCIZ / # PC/
	023534	011	040	120	
	023537	103	000		
3861	023541	040	124	105	DH4: .ASCII / TEST ERROR EXPCTED RECEIVED/<15><12>
	023544	123	124	011	
	023547	105	122	122	
	023552	117	122	011	
	023555	105	130	120	
	023560	103	124	105	
	023563	104	011	122	
	023566	105	103	105	
	023571	111	126	105	
	023574	104	015	012	
3862	023577	040	040	043	.ASCIZ / # PC DATA DATA/
	023602	011	040	120	
	023605	103	011	040	
	023610	104	101	124	
	023613	101	040	040	
	023616	040	040	040	
	023621	104	101	124	
	023624	101	000		
3863	023626	040	124	105	DH5: .ASCII / TEST ERROR HITMIS DATA IN DATA IN/<15><12>
	023631	123	124	011	
	023634	105	122	122	
	023637	117	122	011	
	023642	110	111	124	
	023645	115	111	123	
	023650	011	104	101	
	023653	124	101	040	
	023656	111	116	011	
	023661	104	101	124	

GLOBAL ERROR MESSAGES

	023664	101	040	111					
	023667	116	015	012					
3864	023672	040	040	043		.ASCIZ	/ #	PC	REG. CACHE MEMORY/
	023675	011	040	120					
	023700	103	011	040					
	023703	122	105	107					
	023706	056	011	103					
	023711	101	103	110					
	023714	105	011	115					
	023717	105	115	117					
3865	023722	122	131	000					
	023725	040	124	105	DH7:	.ASCII	/ TEST	ERROR	ADDRESS MSER/<15><12>
	023730	123	124	011					
	023733	105	122	122					
	023736	117	122	011					
	023741	101	104	104					
	023744	122	105	123					
	023747	123	011	115					
	023752	123	105	122					
3866	023755	015	012						
	023757	040	040	043		.ASCIZ	/ #	PC	ACCESSED/
	023762	011	040	120					
	023765	103	011	101					
	023770	103	103	105					
	023773	123	123	105					
3867	023776	104	000						
	024000	040	124	105	DH24:	.ASCII	/ TEST	ERROR	NUMBER/<15><12>
	024003	123	124	011					
	024006	105	122	122					
	024011	117	122	011					
	024014	116	125	115					
	024017	102	105	122					
3868	024022	015	012						
	024024	040	040	043		.ASCIZ	/ #	PC	OF HITS/
	024027	011	040	120					
	024032	103	011	117					
	024035	106	040	110					
	024040	111	124	123					
3869	024043	000							
	024044	105	122	122	DH27:	.ASCII	\ERROR	ERROR	MSER HIT/MISS\<15><12>
	024047	117	122	011					
	024052	105	122	122					
	024055	117	122	011					
	024060	115	123	105					
	024063	122	011	110					
	024066	111	124	057					
	024071	115	111	123					
3870	024074	123	015	012					
	024077	040	040	043		.ASCIZ	/ #	PC/	
	024102	011	040	120					
3871	024105	103	000						
	024107	040	124	105	DH41:	.ASCII	/ TEST	ERROR	INSTRUCTION/<15><12>
	024112	123	124	011					
	024115	105	122	122					
	024120	117	122	011					
	024123	111	116	123					
	024126	124	122	125					

GLOBAL ERROR MESSAGES

	024131	103	124	111					
	024134	117	116	015					
	024137	012							
3872	024140	040	040	043	.ASCIZ	/	#	PC	OPCODE/
	024143	011	040	120					
	024146	103	011	040					
	024151	117	120	103					
	024154	117	104	105					
	024157	000							
3873	024160	040	124	105	DH43:	.ASCII	/	TEST	ERROR EXPECTD RECEIVD CACHE/<15><12>
	024163	123	124	011					
	024166	105	122	122					
	024171	117	122	011					
	024174	105	130	120					
	024177	105	103	124					
	024202	104	011	122					
	024205	105	103	105					
	024210	111	126	104					
	024213	011	103	101					
	024216	103	110	105					
	024221	015	012						
3874	024223	040	040	043	.ASCIZ	/	#	PC	DATA DATA LOCATION/
	024226	011	040	120					
	024231	103	011	040					
	024234	104	101	124					
	024237	101	011	040					
	024242	104	101	124					
	024245	101	011	114					
	024250	117	103	101					
	024253	124	111	117					
	024256	116	000						
3875	024260	040	124	105	DH47:	.ASCII	/	TEST	ERROR ADDRESS ADDRESS/<15><12>
	024263	123	124	011					
	024266	105	122	122					
	024271	117	122	011					
	024274	101	104	104					
	024277	122	105	123					
	024302	123	011	101					
	024305	104	104	122					
	024310	105	123	123					
	024313	015	012						
3876	024315	040	040	043	.ASCIZ	/	#	PC	<21-16> <15-0>/
	024320	011	040	120					
	024323	103	011	074					
	024326	062	061	055					
	024331	061	066	076					
	024334	011	040	074					
	024337	061	065	055					
	024342	060	076	000					
3877	024345	040	124	105	DH65:	.ASCII	/	TEST	ERROR PRIORITY/<15><12>
	024350	123	124	011					
	024353	105	122	122					
	024356	117	122	011					
	024361	120	122	111					
	024364	117	122	111					
	024367	124	131	015					
	024372	012							

GLOBAL ERROR MESSAGES

3878	024373	040	040	043	.ASCIZ / #	PC	LEVEL/	
	024376	011	040	120				
	024401	103	011	114				
	024404	105	126	105				
	024407	114	000					
3879	024411	040	124	105	DH72: .ASCII / TEST	ERROR	ADDRESS/<15><12>	
	024414	123	124	011				
	024417	105	122	122				
	024422	117	122	011				
	024425	101	104	104				
	024430	122	105	123				
	024433	123	015	012				
3880	024436	040	040	043	.ASCIZ / #	PC	FAILED/	
	024441	011	040	120				
	024444	103	011	106				
	024447	101	111	114				
	024452	105	104	000				
3881	024455	040	124	105	DH105: .ASCII / TEST	ERROR	RBUF/<15><12>	
	024460	123	124	011				
	024463	105	122	122				
	024466	117	122	011				
	024471	122	102	125				
	024474	106	015	012				
3882	024477	040	040	043	.ASCIZ / #	PC/		
	024502	011	040	120				
	024505	103	000					
3883	024507	040	124	105	DH115: .ASCII / TEST	ERROR	MSER ADDRESS/<15><12>	
	024512	123	124	011				
	024515	105	122	122				
	024520	117	122	011				
	024523	115	123	105				
	024526	122	011	101				
	024531	104	104	122				
	024534	105	123	123				
	024537	015	012					
3884	024541	040	040	043	.ASCIZ / #	PC	ACCESSED/	
	024544	011	040	120				
	024547	103	040	011				
	024552	011	101	103				
	024555	103	105	123				
	024560	123	105	104				
	024563	000						
3885	024564	040	124	105	DH134: .ASCII / TEST	ERROR	DATA PAR VIRTUAL/<15><12>	
	024567	123	124	011				
	024572	105	122	122				
	024575	117	122	011				
	024600	011	040	040				
	024603	040	104	101				
	024606	124	101	040				
	024611	040	120	101				
	024614	122	040	040				
	024617	040	126	111				
	024622	122	124	125				
	024625	101	114	015				
	024630	012						
3886	024631	040	040	043	.ASCIZ / #	PC	PATTERN READ ADDRESS/	
	024634	011	040	120				

GLOBAL ERROR MESSAGE:

024637	103	040	011				
024642	120	101	124				
024645	124	105	122				
024650	116	040	040				
024653	040	040	122				
024656	105	101	104				
024661	040	040	040				
024664	040	040	040				
024667	040	040	101				
024672	104	104	122				
024675	105	123	123				
024700	000						
3887							
3888						.EVEN	
3889	024702	001162	001116	000000	DT1:	.WORD	\$TMP1,\$ERRPC,0
3890	024710	001162	001116	000001	DT4:	.WORD	\$TMP1,\$ERRPC,R1,CCR,0
	024716	177746	000000				
3891	024722	001162	001116	000002	DT5:	.WORD	\$TMP1,\$ERRPC,R2,R1,\$GDDAT,0
	024730	000001	001124	000000			
3892	024736	001162	001116	001122	DT7:	.WORD	\$TMP1,\$ERRPC,\$BDADR,MSER,0
	024744	177744	000000				
3893	024750	001162	001116	001124	DT14:	.WORD	\$TMP1,\$ERRPC,\$GDDAT,TSTLOC,0
	024756	003162	000000				
3894	024762	001162	001116	001124	DT17:	.WORD	\$TMP1,\$ERRPC,\$GDDAT,RECDAT,0
	024770	004030	000000				
3895	024774	001162	001116	000003	DT24:	.WORD	\$TMP1,\$ERRPC,R3,0
	025002	000000					
3896	025004	001162	001116	177744	DT27:	.WORD	\$TMP1,\$ERRPC,MSER,R3,0
	025012	000003	000000				
3897	025016	001162	001116	001124	DT35:	.WORD	\$TMP1,\$ERRPC,\$GDDAT,MSER,0
	025024	177744	000000				
3898	025030	001162	001116	001126	DT41:	.WORD	\$TMP1,\$ERRPC,\$BDDAT,0
	025036	000000					
3899	025040	001162	001116	000001	DT43:	.WORD	\$TMP1,\$ERRPC,R1,RECDAT,\$BDADR,0
	025046	004030	001122	000000			
3900	025054	001162	001116	172354	DT47:	.WORD	\$TMP1,\$ERRPC,KIPAR6,\$BDADR,0
	025062	001122	000000				
3901	025066	001162	001116	000001	DT50:	.WORD	\$TMP1,\$ERRPC,R1,\$BDADR,0
	025074	001122	000000				
3902	025100	001162	001116	001124	DT51:	.WORD	\$TMP1,\$ERRPC,\$GDDAT,PCR,0
	025106	177522	000000				
3903	025112	001162	001116	001124	DT52:	.WORD	\$TMP1,\$ERRPC,\$GDDAT,BCSR,0
	025120	177520	000000				
3904	025124	001162	001116	001124	DT64:	.WORD	\$TMP1,\$ERRPC,\$GDDAT,LK5-L,0
	025132	002340	000000				
3905	025136	001162	001116	001124	DT65:	.WORD	\$TMP1,\$ERRPC,\$GDDAT,0
	025144	000000					
3906	025146	001162	001116	001124	DT75:	.WORD	\$TMP1,\$ERRPC,\$GDDAT,\$BDDAT,0
	025154	001126	000000				
3907	025160	001162	001116	177562	DT105:	.WORD	\$TMP1,\$ERRPC,RBUF,0
	025166	000000					
3908	025170	001162	001116	001126	DT115:	.WORD	\$TMP1,\$ERRPC,\$BDDAT,KIPAR6,\$BDADR,0
	025176	172354	001122	000000			
3909	025204	001162	001122	000000	DT130:	.WORD	\$TMP1,\$BDDAT,0
3910	025212	001162	001116	001124	DT134:	.WORD	\$TMP1,\$ERRPC,\$GDDAT,\$BDDAT,KIPAR2,\$BDADR,0
	025220	001126	172344	001122			
	025226	000000					

MODIFIED ERROR MESSAGE TYPEOUT ROUTINE

```

3912 .SBTTL MODIFIED ERROR MESSAGE TYPEOUT ROUTINE
3913 ;*****
3914 ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
3915 ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS FROM THE "ERROR TABLE" ($ERRTB),
3916 ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
3917 ;*
3918 ;*THE ONLY DIFFERENCE BETWEEN THIS ROUTINE AND THE ORIGINAL "$ERRTYP" FROM
3919 ;*SYSMAC IS THAT YOU CAN PASS INFORMATION IN GENERAL PURPOSE REGISTERS TO THIS
3920 ;*ROUTINE. THE GENERAL PURPOSE REGISTERS USED ARE TO BE SPECIFIED IN DT*
3921 ;*FORMAT. RO SHOULD NOT BE USED.
3922
3923 025230          ERTYPE:
3924 025230 005037 001162          CLR      $TMP1          ;; JUST CLEAR IT
3925 025234 113737 001102 001162  MOVB    $TSTNM,$TMP1    ;; STORE TEST NUMBER
3926 025242 104401 001175          TYPE    $CRLF          ;; "CARRIAGE RETURN" & "LINE FEED"
3927 025246 010046          MOV     RO,-(SP)        ;; SAVE RO
3928 025250 005000          CLR     RO              ;; PICKUP THE ITEM INDEX
3929 025252 153700 001114          BISB   @#$ITEMB,RO
3930 025256 001004          BNE     1$
3931
3932 025260 013746 001116          MOV     $ERRPC,-(SP)    ;; IF ITEM NUMBER IS ZERO, JUST
3933
3934 025264 104402          TYPCC          ;; TYPE THE PC OF THE ERROR
3935 025266 000426          BR      6$              ;; SAVE $ERRPC FOR TYPEOUT
3936 025270 005300          1$: DEC     RO              ;; ERROR ADDRESS
3937 025272 006300          ASL    RO              ;; GO TYPE--OCTAL ASCII(ALL DIGITS)
3938 025274 006300          ASL    RO              ;; GET OUT
3939 025276 006300          ASL    RO              ;; ADJUST THE INDEX SO THAT IT WILL
3940 025300 062700 001324          ADD    @#$ERRTB,RO    ;; WORK FOR THE ERROR TABLE
3941 025304 012037 025314          MOV    (RO)+,2$
3942 025310 001404          BEQ   3$
3943 025312 104401          TYPE
3944 025314 000000          2$: .WORD 0          ;; ERROR MESSAGE POINTER GOES HERE
3945 025316 104401 001175          TYPE    $CRLF          ;; "CARRIAGE RETURN" & "LINE FEED"
3946 025322 012037 025332          3$: MOV    (RO)+,4$        ;; PICKUP "DATA HEADER" POINTER
3947 025326 001404          BEQ   5$
3948 025330 104401          TYPE
3949 025332 000000          4$: .WORD 0          ;; SKIP TYPEOUT IF 0
3950 025334 104401 001175          TYPE    $CRLF          ;; TYPE THE "DATA HEADER"
3951 025340 011000          5$: MOV    (RO),RO        ;; "DATA HEADER" POINTER GOES HERE
3952 025342 001004          BNE   7$
3953 025344 012600          6$: MOV    (SP)+,RO      ;; "CARRIAGE RETURN" & "LINE FEED"
3954 025346 104401 001175          TYPE    $CRLF          ;; RESTORE RO
3955 025352 0C0207          RTS     PC              ;; "CARRIAGE RETURN & "LINE FEED"
3956 025354          7$:
3957 025354 021027 000005          CMP    (RO),#5        ;; RETURN
3958 025360 101021          BHI   9$
3959 025362 042737 000700 025416          BIC    #BIT8!BIT7!BIT6,8$ ;; GENERAL PURPOSE REGISTER?
3960 025370 011037 001160          MOV    (RO),$TMP0     ;; IF NOT, GO TYPE DATA
3961 025374 000337 001160          SWAB  $TMP0          ;; CLEAR BITS FOR SOURCE REGISTER
3962 025400 006237 001160          ASR   $TMP0          ;; SAVE (RO)
3963 025404 006237 001160          ASR   $TMP0          ;; GET REGISTER NUMBER TO HIGH BYTE
3964 025410 053737 001160 025416          BIS   $TMP0,8$       ;; GET REGISTER NUMBER TO BITS 8 6
3965
3966 025416 010046          8$: MOV    RO,-(SP)      ;; SET BITS IN MOV INSTRUCTION
3967 025420 005720          TST   (RO)+          ;; ACCORDING TO REGISTER NUMBER
3968 025422 000401          BR    10$           ;; MOVE CONTEXT OF REGISTER TO STACK
                          ;; ADVANCE POINTER
                          ;; GO TYPE

```

MODIFIED ERROR MESSAGE TYPEOUT ROUTINE

```

3969 025424 013046          9$:   MOV    @ (RO)+, -(SP)  ;; IF NOT GPR, SAVE @ (RO)+ FOR TYPEOUT
3970 025426 104402          10$:  TYPOC                ;; GO TYPE--OCTAL ASCI (ALL DIGITS)
3971 025430 005710                TST    (RO)           ;; IS THERE ANOTHER NUMBER?
3972 025432 001744                BEQ    6$             ;; BR IF NO
3973 025434 104401 025442        TYPE   11$           ;; TYPE TWO(2) SPACES
3974 025440 000745                BR     7$
3975 025442 040      040      000 11$:  .ASCIZ  / /           ;; TWO(2) SPACES
3976
3977
3978
3979
3980
3981
3982
3983
3984
3985
3986
3987
3988
3989
3990
3991
3992
3993
3994
3995
3996
3997
3998
3999
4000
4001 025446 012701 172240      INITMM: MOV    #172240, R1      ;BASE ADDRESS OF SIPARS
4002 025452 004737 025610      JSR    PC, LDPARS
4003 025456 012701 172260      MOV    #172260, R1      ;BASE ADDRESS OF SDPARS
4004 025462 004737 025610      JSR    PC, LDPARS
4005 025466 012701 172340      MOV    #172340, R1      ;BASE ADDRESS OF KIPARS
4006 025472 004737 025610      JSR    PC, LDPARS
4007 025476 012701 172360      MOV    #172360, R1      ;BASE ADDRESS OF KOPARS
4008 025502 004737 025610      JSR    PC, LDPARS
4009 025506 012701 177640      MOV    #177640, R1      ;BASE ADDRESS OF UIPARS
4010 025512 004737 025610      JSR    PC, LDPARS
4011 025516 012701 177660      MOV    #177660, R1      ;BASE ADDRESS OF UDPARS
4012 025522 004737 025610      JSR    PC, LDPARS
4013 025526 012701 177600      MOV    #177600, R1      ;BASE ADDRESS OF UIPDRS
4014 025532 004737 025640      JSR    PC, LDPDRS
4015 025536 012701 177620      MOV    #177620, R1      ;BASE ADDRESS OF UDPDRS
4016 025542 004737 025640      JSR    PC, LDPDRS
4017 025546 012701 172300      MOV    #172300, R1      ;BASE ADDRESS OF KIPDRS
4018 025552 004737 025640      JSR    PC, LDPDRS
4019 025556 012701 172320      MOV    #172320, R1      ;BASE ADDRESS OF KOPDRS
4020 025562 004737 025640      JSR    PC, LDPDRS
4021 025566 012701 172200      MOV    #172200, R1      ;BASE ADDRESS OF SIFDRS
4022 025572 004737 025640      JSR    PC, LDPDRS
4023 025576 012701 172220      MOV    #172220, R1      ;BASE ADDRESS OF SUPDRS
4024 025602 004737 025640      JSR    PC, LDPDRS
4025 025606 000207                RTS    PC              ;RETURN

```

.SBTTL GLOBAL SUBROUTINES SECTION

;;
; THE GLOBAL SUBROUTINES SECTION CONTAINS THE SUBROUTINES
; THAT ARE USED IN MORE THAN ONE TEST.
;--

;;
; FUNCTIONAL DESCRIPTION:
; SUBROUTINE TO INITIALIZE ALL THE MMU REGISTERS

; INPUTS: NONE

; OUTPUTS: NONE

; SUBORDINATE ROUTINES USED: LOAD PARS
; LOAD PDRS

; FUNCTIONAL SIDE EFFECTS: NONE

; CALLING SEQUENCE: JSR PC, INITMM

GLOBAL SUBROUTINES SECTION

```

4027
4028
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4039
4040
4041
4042
4043
4044
4045
4046
4047 025610 012702 000006
4048 025614 005003
4049 025616 010321
4050 025620 062703 000200
4051 025624 077204
4052 025626 012721 002000
4053 025632 012711 177600
4054 025636 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

```

4056
4057
4058
4059
4060
4061
4062
4063
4064
4065
4066
4067
4068
4069
4070
4071
4072
4073
4074
4075
4076
4077 025640 012702 000006
4078 025644 012721 177406
4079 025650 077203
4080 025652 012721 077406
4081 025656 012711 077406
4082 025662 000207

```

```

: **
: FUNCTIONAL DESCRIPTION:
: SUBROUTINE TO INITIALIZE ALL THE MMU PAGE DECRIPTOR 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

```

4084
4085      ;*
4086      ; FUNCTIONAL DESCRIPTION:
4087      ;   SUBROUTINE TO HANDLE PARITY ERROR ABORTS FROM THE RAM STORE RAM TESTS.
4088
4089      ; INPUTS:
4090      ;   MEMORY SYSTEM ERROR REGISTER CONTAINS BITS INDICATING FAILURE
4091
4092      ; OUTPUTS: NONE
4093
4094      ; SUBORDINATE ROUTINES USED: NONE
4095
4096      ; FUNCTIONAL SIDE EFFECTS: NONE
4097
4098      ; CALLING SEQUENCE: CALLED BY PARITY ABORT
4099      ;   MOV    @#114, SLOC00 ;SAVE CONTENTS OF PARITY ABORT VECTOR
4100      ;   MOV    #DSPA, @#114 ;LET VECTOR POINT TO PARITY ABORT ROUTINE
4101
4102      ;   (CACHE PARITY ERROR OCCURS)
4103 025664 011637 001122      RAMPAR: MOV    (SP), $BDADR      ;STOR ADDESS TRAPPED
4104 025670 032737 000100 177744      BIT    #BIT06, MSER      ;IF LOW BYTE PARITY ERROR
4105 025676 001401          BEQ    1$              ;THEN
4106 025700 104004          ERROR  +4              ;ERROR
4107 025702          1$:: BIT    #BIT07, MSER      ;IF HIGH BYTE PARITY ERROR
4108          BEQ    2$              ;THEN
4109          ERROR  +4              ;ERROR
4110 025702 032737 000040 177744      2$: BIT    #BIT05, MSER      ;IF TAG PARITY ERROR
4111 025710 001401          BEQ    3$              ;THEN
4112 025712 104004          ERROR  +4              ;ERROR
4113 025714 005037 177744      3$: CLR    MSER              ;INITIALIZE MSER AFTER ERROR
4114 025720 000002          RTI              ;RETURN
4115 025722 005237 002340      LKSINT: INC    LKSFL              ;INCREMENT FLAG
4116 025726 000002          RTI
4117
4118      .SBTTL Q22BE SIZE ROUTINE
4119      ;THIS ROUTINE WILL AUTOSIZE FOR UP TO TWO Q22 BUS EXERCISERS. IF NONE
4120      ;FOUND LOCATIONS CSR1 AND CSR12 WILL BE LEFT ZEROES. THIS ROUTINE WILL
4121      ;ONLY RUN IN NOT UFD MODE.
4122
4123 025730 032737 001000 177750      Q22SIZ: BIT    #BIT09, MAIREG      ;JNTIBUS SYSTEM?
4124 025736 001401          BEQ    1$              ;IF NOT, ADVANCE TO ROUTINE
4125 025740 000207          RTS    PC              ;OTHERWISE, RETURN
4126
4127      ; PREPARE TO DO SIZING
4128
4129 025742 013701 000004          1$: MOV    ERRVEC, R1      ;STORE TIMEOUT VECTOR
4130 025746 012737 026122 000004      MOV    #7$, ERRVEC      ;POINT NEW TO PROGRAM
4131 025754 012737 000340 000006      MOV    #340, ERRVEC+2  ;AT PRIORITY 7
4132 025762 005037 001160          CLR    $TMP0           ;CLEAR Q22BE COUNTER
4133 025766 012702 170000          MOV    #170000, R2     ;FIRST POSSIBLE ADDRESS
4134 025772 012703 000510          MOV    #510, R3       ;VECTOR FOR IT
4135 025776 000404          BR    3$              ;TRY THOSE VALUES
4136
4137      ; NOW DO ACTUAL SIZING
4138
4139 026000 062702 000020          2$: ADD    #20, R2      ;GET CSR FOR NEXT Q22BE
4140 026004 062703 000004          ADD    #4, R3         ;GET VECTOR FOR NEXT ONE

```

Q22BE SIZE ROUTINE

```

4141 026010 005712      3$:   TST      (R2)                ;TRY TO ACCESS CSR
4142
4143                   ; IF NO TIMEOUT, STORE EXISTING ADDRESSES TO REGISTERS
4144                   ;
4145 026012 005737 001160      TST      $TMP0                ;FIRST Q22BE FOUND?
4146 026016 001010          BNE      4$                  ;IF SECOND, BRANCH
4147 026020 012705 002274      MOV      #CSR1,R5           ;START WITH CSR1 FOR 1ST
4148 026024 010237 002312      MOV      R2,SIMGOA         ;SIMULTANEOUS GO
4149 026030 062737 000016 002312  ADD      #16,SIMGOA        ;ADDRESS
4150 026036 000402          BR       5$                  ;BRANCH TO INITIALISE
4151 026040 012705 002322      4$:   MOV      #CSR12,R5      ;START WITH CSR12 FOR 2ND
4152 026044 012704 000004      5$:   MOV      #4,R4         ;INITIALISE 5 REGISTERS
4153 026050 010715          MOV      R2,(R5)           ;INITIALISE CSR1
4154 026052 011565 000002      6$:   MOV      (R5),2(R5)     ;STORE TO NEXT ONE
4155 026056 005725          TST      (R5)+             ;GET NEXT ADDRESS
4156 026060 062715 000002      ADD      #2,(R5)         ;GET ADDRESS, POINT NEXT
4157 026064 077406          SOB      R4,6$            ;DO FOR NEXT 4 REGISTERS
4158 026066 010365 000002      MOV      R3,2(R5)        ;STORE INTERRUPT VECTOR
4159 026072 010365 000004      MOV      R3,4(R5)        ;AND PRIORITY
4160 026076 062765 000002 000004  ADD      #2,4(R5)
4161 026104 005237 001160      INC      $TMP0            ;COUNT Q22BE S
4162 026110 022737 000002 001160  CMP      #2,$TMP0         ;TWO FOUND?
4163 026116 001406          BEQ      9$                ;IF SO, STOP SIZING
4164 026120 000402          BR       8$                ;OTHERWISE, CONTINUE SIZING
4165
4166                   ; ON TIMEOUT TRY TO LOOK AT NEXT ADDRESS RANGE
4167                   ;
4168 026122 005726          7$:   TST      (SP)+         ;RESTORE STACK FROM
4169 026124 005726          TST      (SP)+         ;TIMEOUT
4170 026126 022702 170160      8$:   CMP      #170160,R2   ;AT THE LAST POSSIBLE?
4171 026132 001322          BNE      2$                ;IF NOT, BRANCH
4172 026134 005737 002274      9$:   TST      CSR1         ;1 FOUND?
4173 026140 001402          BEQ      10$             ;IF NONE, BRANCH
4174 026142 104401 026154      TYPE    ,ONQ22           ;TYPE FOUND
4175 026146 010137 000004      10$:  MOV      R1,ERRVEC      ;RESTORE TIMEOUT VECTOR
4176 026152 000207          RTS      PC              ;RETURN
4177
4178 026154      012      015      121  ONOQ22: .ASCIZ <12><15>/Q22BE USED DURING TESTING/
      026157      062      062      102
      026162      105      040      125
      026165      123      105      104
      026170      040      104      125
      026173      122      111      116
      026176      107      040      124
      026201      105      123      124
      026204      111      116      107
      026207      000
4179
4180 .EVEN
4181 .SBTTL Q22BE INTERRUPT INITIALISE ROUTINE
4182 ;THIS ROUTINE WILL INITIALISE Q22BE TO INTERRUPT AT A PRIORITY AT (R0)+
4183 ;AT THE STARTING ADDRESS IN R3. THE TEST HAVE TO SET ACTUAL DONE BIT
4184 ;BY CLEARING GO.
4185 026210 005013      Q22INT: CLR      (R3)                ;CLEAR TRANSFER TYPE IN CSR1
4186 026212 052710 000001      BIS      #BIT00,(R0)     ;ZERO DONE
4187 026216 011063 000002      MOV      (R0),2(R3)     ;SET PRIORITY IN CSR2
4188 026222 042710 000001      BIC      #BIT00,(R0)     ;PREPARE TO SET DONE

```

Q22BE INTERRUPT INITIALISE ROUTINE

```

4189 026226 000207          RTS      PC
4190
4191          .SBTTL DMATR N DATO CYCLE THRU Q22BE
4192          ;THIS ROUTINE PERFORMS DATO FROM A LOCATION TEMP THRU THE FIRST
4193          ;FOUND Q22BE STARTING AT LOCATION @CSR1. RO HAS 0 IF ONLY 1 TRANSFER IS
4194          ;TO BE PERFORMED. OTHERWISE 16 BLOCK MODE TRANSFERS ARE TO BE PERFORMED.
4195          ;IN THE LATTER CASE ADDRESS AND WORD COUNT HAS TO BE LOADED BEFORE.
4196
4197 026230 012777 012525 154046 DMATR N: MOV      #12525,@DATA          ;DATA USED
4198 026236 005700          TST      RO              ;DO 1 WORD?
4199 026240 001404          BEQ      1$              ;IF YES, BRANCH
4200 026242 012777 001001 154026          MOV      #BIT09!BIT00,@CSR2 ;BLOCK MODE, GO
4201 026250 000414          BR       2$              ;BRANCH TO DO IT
4202 026252 012777 001501 154014 1$: MOV      #1601,@CSR1      ;RESET LATENCY COUNT,DATO
4203 026260 012777 002740 154012          MOV      #TEMP,@BA        ;LOAD DMA ADDRESS
4204 026266 012777 177777 154006          MOV      #177777,@WC      ;DO 1 WORD
4205 026274 012777 000001 153774          MOV      #BIT00,@CSR2    ;DO IT
4206 026302 105777 153770          2$: TSTB   @CSR2          ;DMA DONE?
4207 026306 100375          BPL      2$              ;WAIT TILL DONE
4208 026310 000207          3$: RTS      PC          ;RETURN FROM SUBROUTINE
4209
4210          .SBTTL DMARD DATI THRU Q22BE
4211          ;THIS ROUTINE PERFORMS DATI CYCLE THRU Q22BE IN EITHER BLOCK MODE OR A SINGLE
4212          ;TRANSFER MODE. MEMORY LOCATION USED IS TEMP. RO IS ZERO FOR SINGLE TRANSFER
4213
4214 026312 012777 002740 153760 DMARD:  MOV      #TEMP,@BA          ;LOAD DMA ADDRESS
4215 026320 005700          TST      RO              ;DO 1 WORD?
4216 026322 001412          BEQ      1$              ;IF YES, BRANCH
4217 026324 012777 001507 153742          MOV      #1507,@CSR1     ;16 DATI B
4218 026332 012777 177770 153742          MOV      #177770,@WC    ;DO 8 WORD
4219 026340 012777 001001 153730          MOV      #BIT09!BIT00,@CSR2 ;BLOCK MODE GO
4220 026346 000411          BR       2$              ;GO CHECK
4221 026350 012777 001407 153716 1$: MOV      #1407,@CSR1     ;RESET LATENCY COUNT,DATI
4222
4223 026356 012777 177777 153716          MOV      #177777,@WC    ;LOAD NEW DATA TO DATA R.
4224 026364 012777 000001 153704          MOV      #BIT00,@CSR2   ;DO 1 WORD
4225 026372 105777 153700          2$: TSTB   @CSR2          ;DO IT
4226 026376 100375          BPL      2$              ;DMA DONE?
4227 026400 000207          3$: RTS      PC          ;RETURN FROM SUBROUTINE
4228
4229
4230
4231 026402 011637 001122          TOUT:  MOV      (SP), $BDADR ;STORE TRAPPED PC
4232 026406 104041          ERROR  +41              ;UNEXPECTED TRAP
4233 026410 000002          RTI
4234
4235
4236          ;MMU GLOBAL SUBROUTINES
4237
4238
4239
4240          ;ROUTINE TO INITIALIZE MEMORY MANAGEMENT
4241
4242 026412 010046          MMU:   MOV      RO,-(SP)   ;SAVE CONTENTS OF REGISTERS
4243 026414 010146          MOV      R1,-(SP)
4244 026416 010246          MOV      R2,-(SP)
4245 026420 012700 177600          MOV      #177600,RO
4246 026424 004737 026512          JSR      PC,PDR         ;INIT I AND D USER PDR'S

```

DMARD DATI THRU Q22BE

4247	026430	004737	026534		JSR	PC,PAR		;INIT I USER PAR'S
4248	026434	004737	026534		JSR	PC,PAR		;INIT D USER PAR'S
4249	026440	012700	172200		MOV	#172200,R0		
4250	026444	004737	026512		JSR	PC,PDR		;INIT I AND D SUP PDR S
4251	026450	004737	026534		JSR	PC,PAR		;INIT I SUP PAR'S
4252	026454	004737	026534		JSR	PC,PAR		;INIT D SUP PAR'S
4253	026460	004737	026512		JSR	PC,PDR		;INIT I AND D KER PDR S
4254	026464	004737	026534		JSR	PC,PAR		;INIT I KER PAR'S
4255	026470	004737	026534		JSR	PC,PAR		;INIT D KER PAR'S
4256	026474	012737	000027	172516	MOV	#27,@#172516		;INIT MMR3
4257	026502	012602			MOV	(SP)+,R2		;RESTORE REGISTERS
4258	026504	012601			MOV	(SP)+,R1		
4259	026506	012600			MOV	(SP)+,R0		
4260	026510	000207			RTS	PC		;RETURN
4261								
4262								;ROUTINE TO INITIALIZE PDR'S
4263								
4264	026512	005002			PDR:	CLR	R2	;INIT CNTR
4265	026514	012720	077406		PDR1:	MOV	#77406,(R0)+	;INIT PDR
4266	026520	062702	000001			ADD	#1,R2	;INCREMENT CNTR
4267	026524	022702	000020			CMP	#16,R2	;ARE WE DONE?
4268	026530	001371				BNE	PDR1	;BRANCH IF NOT
4269	026532	000207				RTS	PC	;RETURN
4270								
4271								;ROUTINE TO INITIALIZE PAR'S
4272								
4273	026534	005001			PAR:	CLR	R1	;SETUP TO INIT PAR
4274	026536	010120			PAR1:	MOV	R1,(R0)+	;INIT PAR
4275	026540	062701	000200			ADD	#200,R1	;GET READY FOR NEXT PAR
4276	026544	022701	001600			CMP	#1600,R1	;REACHED A PAR?
4277	026550	001372				BNE	PAR1	;BRANCH IF NOT
4278	026552	012720	177600			MOV	#177600,(R0)+	;INIT PAR?
4279	026556	000207				RTS	PC	;RETURN
4280								
4281								;TIME OUT ROUTINE
4282								
4283	026560	005205			ADDTRP:	INC	R5	;INCREMENT TIME OUT FLAG
4284	026562	000002				RTI		;RETURN
4285								
4286								;MMU TRAP ROUTINE
4287								
4288	026564	023727	003026	000001	MMUTRP:	CMP	FLAG,#1	;ARE WE EXPECTING AN ABORT
4289	026572	001401				BEQ	1\$;YES GO ON
4290	026574	104002				ERROR	+2	;NO GO TO ERROR
4291	026576	010046			1\$:	MOV	R0,-(SP)	;SAVE CONTENTS OF REG 0
4292	026600	013700	177776			MOV	@#177776,R0	;SAVE A COPY OF PSW
4293	026604	072027	177764			ASH	#-14,R0	;LOOK AT BITS<15:14>
4294	026610	020027	000002			CMP	R0,#2	;WAS PS<15:14>=10
4295	026614	001001				BNE	OK	;NO GO ON
4296	026616	000411				BR	NOTOK	;YES CHANGE BITS TO 00
4297	026620	013700	177776		OK:	MOV	@#177776,R0	;SAVE A COPY OF PSW
4298	026624	072027	000002			ASH	#2,R0	;LOOK AT BITS<13:12>
4299	026630	072027	177764			ASH	#-14,R0	
4300	026634	020027	000002			CMP	R0,#2	;WAS PS<13:12>=10
4301	026640	001002				BNE	OK1	;NO GO ON
4302	026642	005066	000004		NOTOK:	CLR	4(SP)	;CLEAR ILLEGAL MODE FROM OLD PSW
4303	026646	013737	177572	003040	OK1:	MOV	@#177572,SAVMRO	;SAVE A COPY OF MMRO

DMARD DATI THRU Q22BE

4304	026654	013737	177574	003042	MOV	@#177574,SAVMR1	;SAVE A COPY OF MMR1
4305	026662	013737	177576	003044	MOV	@#177576,SAVMR2	;SAVE A COPY OF MMR2
4306	026670	005037	177572		CLR	@#177572	;CLEAR ABORT BITS AND TURN MMU OFF
4307	026674	005037	003026		CLR	FLAG	;CLEAR MMU ABORT FLAG
4308	026700	012600			MOV	(SP)+,R0	;RESTORE ORIGINAL CONTENTS OF REG 0
4309	026702	000002			RTI		;RETURN

DMARD DATI THRU Q22BE

```

4313      ;FPP COMMON SUBROUTINES
4314 026704 012600 WLDTRP: MOV      (SP)+,R0      ;SAVE PC
4315 026706 012605      MOV      (SP)+,R5      ;SAVE STATUS AND RESTORE STACK
4316 026710 104003      ERROR   +3
4317 026712 000110      JMP      (R0)          ;GO BACK INLINE
4318      :
4319      :
4320      :
4321 026714 000000 TRPFLG: WORD   0
4322 026716 000207 ERRFP:  RTS    R7
4323 026720 000207 ERR:   RTS    R7
4324      :
4325      :
4326      :
4327      :
4328      :
4329      :SUBROUTINE DATA VERFICATION -
4330      :
4331      : CALLED BY   JSR    R7,DATVER
4332      :
4333      : INPUT:      (R4)=EXPECTED DATA
4334      :              (R1)=RECEIVED DATA
4335      :
4336      : THIS ROUTINE VERIFIES THAT THE 4 CONSECUTIVE WORDS STARTING WITH (R4) ARE
4337      : EQUAL TO THE FOUR WORDS ADDRESSED BY (R1). THE CONTENTS OF R4, AND R1 ARE NOT
4338      : DISTURBED.
4339      : LOCATION "COUNT" , IF NOT EQUAL TO 0 SIGNIFIES DATA ERROR
4340      : IF THE STATUS IS FLOATING MODE, THE LAST TWO BYTES OF RECEIEVED
4341      : ARE SIMPLY CHECKED FOR ZEROS
4342      :
4343      :
4344 026722 010446 DATVFR: MOV      R4,-(SP)      ;SAVE R4
4345 026724 010146      MOV      R1,-(SP)      ;SAVE R1
4346 026726 012737 000003 003120      MOV      #3,COUNT      ;SET UP ITERATION COUNT
4347 026734 000137 026752      JMP      DAT1
4348      :
4349 026740 010446 DATVER: MOV      R4,-(SP)      ;SAVE R4
4350 026742 010146      MOV      R1,-(SP)      ;SAVE R1
4351 026744 012737 000005 003120      MOV      #5,COUNT      ;SET UP ITERATION COUNT
4352 026752 005337 003120 DAT1:  DEC      COUNT
4353 026756 001402      BEQ      2$          ;BRANCH IF DONE
4354 026760 022421      CMP      (R4)+,(R1)+
4355 026762 001773      BEQ      DAT1
4356 026764 012601      ?$      MOV      (SP)+,R1      ;RESTORE R1
4357 026766 012604      MOV      (SP)+,R4      ;RESTORE R4
4358 026770 000207      RTS      R7          ;GO BACK TO CALLING ROUTINE
4359      :

```

DMARD DATI THRU Q22BE

```

4361
4362
4363
4364
4365
4366
4367
4368 026772
4369 026772 013746 000004
4370 026776 013746 000006
4371 027002 010046
4372 027004 010146
4373
4374 027006 012737 027034 000004
4375 027014 106737 000006
4376
4377 027020 012700 000001
4378
4379 027024 012701 011610
4380
4381 027030 005737 160000
4382
4383 027034 022626
4384 027036 077104
4385 027040 077007
4386
4387 027042 012601
4388 027044 012600
4389 027046 012637 000006
4390 027052 012637 000004
4391
4392 027056 000207
4393

```

```

*****
: DELAY - INTERRUPT TIMER
: This test is called by the interrupt receiver and transmitter
:
:-----

```

```

DELAY:
      MOV    @#4,-(SP)      ;SAVE
      MOV    @#6,-(SP)      ;  OLD
      MOV    R0,-(SP)      ;  STACK
      MOV    R1,-(SP)      ;  AND REGS
      MOV    #2,@#4        ;SET TRAP
      MFPS   @#6           ;PUT PSW ON PS / SAME PRIORITY
      MOV    #1.,R0        ;SET OUTER LOOP COUNT
      3$:   MOV    #5000.,R1 ;SET INNER LOOP 1/200TH
      1$:   TST    @#160000 ;TAKES ABOUT 10u SEC
      2$:   CMP    (SP)+,(SP)+ ;CLEAN UP STACK
            SOB    R1,1$     ;DO INNER LOOP
            SOB    R0,3$     ;DO OUTER LOOP
      MOV    (SP)+,R1      ;RESTORE
      MOV    (SP)+,R0      ;  WHAT
      MOV    (SP)+,@#6     ;  WE
      MOV    (SP)+,@#4     ;  SAVED
      RTS    PC

```

DMARD DATI THRU Q228E

```

4395
4396
4397
4398
4399
4400
4401
4402 027060
4403 027060 012737 000000 172340
4404 027066 012737 000200 172342
4405 027074 012737 000400 172344
4406 027102 012737 000600 172346
4407 027110 012737 001000 172350
4408 027116 012737 001200 172352
4409 027124 012737 001400 172354
4410 027132 012737 177600 172356
4411 027140 012702 000010
4412 027144 012701 172300
4413 027150 012721 077406
4414 027154 077203
4415 027156 012737 000001 177572
4416 027164 012737 000020 172516
4417 027172 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:
MOV #0,@#KIPAR0 ; SUBR SET UP MMU REGISTERS
MOV #200,@#KIPAR1 ; POINT TO PAGE 1 TO ITSELF
MOV #400,@#KIPAR2 ; POINT TO PAGE 2 TO ITSELF
MOV #600,@#KIPAR3 ; POINT TO PAGE 3 TO ITSELF
MOV #1000,@#KIPAR4 ; POINT TO PAGE 4 TO ITSELF
MOV #1200,@#KIPAR5 ; POINT TO PAGE 5 TO ITSELF
MOV #1400,@#KIPAR6 ; POINT TO PAGE 6 TO ITSELF
MO #177600,@#KIPAR7 ; POINT I/O PAGE TO IT'S PLACE IN 22 BIT
MOV #10,R2 ; BGND0
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
    
```


DMARD DATI THRU Q22BE

4419
4420
4421
4422
4423
4424
4425
4426
4427
4428
4429
4430
4431
4432
4433
4434
4435
4436
4437
4438
4439
4440
4441
4442
4443
4444
4445
4446
4447
4448
4449
4450
4451
4452
4453
4454
4455
4456
4457
4458
4459
4460
4461
4462
4463
4464
4465
4466
4467
4468
4469
4470
4471
4472
4473
4474

```

*****
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:
10$:  MOV    #1600,@#KIPAR2
      MOV    #40000,R1
      MOVB  #-1,(R1)
      INC   R1
      CMP   R1,#60000
      BLO  20$
      ADD   #200,@#KIPAR2
      BIC  #160000,R1
      BIS  #40000,R1
      CMP   @#KIPAR2,#20000
      BEQ  40$
20$:  MOVB  #0,(R1)
      INC   R1
      CMP   R1,#60000
      BLO  30$
      ADD   #200,@#KIPAR2
      BIC  #160000,R1
      BIS  #40000,R1
      CMP   @#KIPAR2,#20000
      BEQ  40$
30$:  MOVB  #0,(R1)
      INC   R1
      CMP   R1,#60000
      BLO  40$
      ADD   #200,@#KIPAR2
      BIC  #160000,R1
      BIS  #40000,R1
      CMP   @#KIPAR2,#20000
      BEQ  40$
40$:  BR    10$
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 Q22BE

```

4476
4477
4478
4479
4480
4481
4482
4483
4484
4485
4486
4487
4488
4489
4490 027360
4491 027360 012737 001600 172344
4492 027366 012701 040000
4493 027372 012702 000003
4494 027376 160402
4495 027400
4496 027400 060401
4497 027402 020127 060000
4498 027406 103413
4499 027410 062737 000200 172344
4500 027416 042701 160000
4501 027422 052701 040000
4502 027426 023727 172344 020000
4503 027434 001435
4504 027436 005703
4505 027440 001012
4506 027442 120511
4507 027444 001411
4508 027446 010537 001124
4509 027452 111137 001126
4510 027456 010137 001122
4511 027462 104066
4512 027464 000401
4513 027466
4514 027466 110511
4515 027470
4516 027470 060201
4517 027472 020127 060000
4518 027476 103740
4519 027500 062737 000200 172344
4520 027506 042701 160000
4521 027512 052701 040000
4522 027516 023727 172344 020000
4523 027524 001401
4524 027526 000724
4525 027530 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
BGND0
: 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
: ENDF
: 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
: ENDF
: ELSE
: WRITE PATTERN TO MEMORY
: ENDF
: 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
: ENDF
: DOUNTIL WE HAVE READ ALL ADDRESSES
: ENDSUBR READ_WRITE_MEMORY

```

DMARD DATI THRU Q228E

```

4527
4528
4529
4530
4531
4532
4533
4534
4535 027532
4536 027532 012737 001600 172344
4537 027540 012701 040000
4538 027544
4539 027544 111137 001126
4540 027550 122737 177777 001126
4541 027556 001406
4542 027560 012737 177777 001124
4543 027566 010137 001122
4544 027572 104066
4545 027574 005201
4546 027576 020127 060000
4547 027602 103413
4548 027604 062737 000200 172344
4549 027612 042701 160000
4550 027616 052701 040000
4551 027622 023727 172344 020000
4552 027630 001435
4553 027632 111137 001126
4554 027636 122737 177777 001126
4555 027644 001406
4556 027646 112737 177777 001124
4557 027654 011137 001122
4558 027660 104066
4559 027662 062701 000002
4560 027666 020127 060000
4561 027672 103724
4562 027674 062737 000200 172344
4563 027702 042701 160000
4564 027706 052701 040000
4565 027712 023727 172344 020000
4566 027720 001401
4567 027722 000710
4568 027724 000207
4569

```

```

*****
; 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 _0W 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
: DOUNTIL WE HAVE READ ALL ADDRESSES
ENDSUBR READ_LOCATIONS_0,1

```

DMARD DATI THRU Q228E

4571
 4572
 4573
 4574
 4575
 4576
 4577
 4578
 4579
 4580
 4581
 4582
 4583
 4584
 4585
 4586
 4587
 4588
 4589
 4590
 4591
 4592
 4593
 4594 027726
 4595 027726 104401 027734
 4596 027732 000207
 4597
 4598 027734 105 122 122
 027737 117 122 040
 027742 104 105 124
 027745 105 103 124
 027750 105 104 040
 027753 111 116 040
 027756 112 061 061
 027761 040 106 114
 027764 117 101 124
 027767 111 116 107
 027772 040 120 117
 027775 111 116 124
 030000 040 120 122
 030003 117 103 105
 030006 123 123 117
 030011 122 056 000
 4599
 4600

```

:$$$
: SUBROUTINE - DETERMINE FLOATING POINT ACCELERATOR (DETFPA)
: THIS SUBROUTINE IS CALLED IF AN ERROR IS DETECTED DURING EXECUTION OF THE
: FLOATING POINT TESTS.
: IT DETERMINES WHETHER OR NOT THE FLOATING POINT ACCELERATOR CHIP OPTION
: IS PRESENT ON THE CPU BOARD AND PRINTS THE APPROPRIATE ERROR MESSAGE.
: THIS DETERMINATION IS MADE BASED ON THE "FPA AVAILABLE" FLAG, BIT 8
: OF THE MAINTENANCE REGISTER AT LOCATION 1777750. IF THE FPA BIT IS SET
: THEN THE FLOATING POINT ACCELERATOR CHIP IS INSTALLED ON THE CPU BOARD AND
: AN ERROR MESSAGE IS PRINTED WHICH STATES THAT THE FLOATING POINT ERROR IS
: DUE TO THIS CHIP. OTHERWISE, THE J11 IS BLAMED FOR THE FLOATING POINT ERROR.
:$$$
: 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

4603
 4604
 4605

```

.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:
030014      030014      032737      000100      000052      BIT #BIT06,@#52
030022      030022      001030      000000      000000      BNE $GET42
030024      030024      004537      016256      000000      jsr r5,vireop
030030      030030      005037      001102      000000      CLR $TSTNM          ;;ZERO THE TEST NUMBER
030034      030034      005037      001164      000000      CLR $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
030040      030040      005237      001206      000000      INC $PASS           ;;INCREMENT THE PASS NUMBER
030044      030044      042737      100000      001206      BIC #100000,$PASS  ;;DON'T ALLOW A NEG. NUMBER
030052      030052      005327      000000      000000      DEC (PC)+           ;;LOOP?
030054      030054      000001      000000      000000      $EOPCT: .WORD 1
030056      030056      003022      000000      000000      BGT $DOAGN          ;;YES
030060      030060      012737      000000      000000      MOV (PC)+,@(PC)+  ;;RESTORE COUNTER
030062      030062      000001      000000      000000      $ENDCT: .WORD 1
030064      030064      030054      000000      000000      $EOPCT
030066      030066      104401      030133      000000      TYPE $ENDMG        ;;TYPE "END PASS #"
030072      030072      013746      001206      000000      MOV $PASS,-(SP)    ;;SAVE $PASS FOR TYPEOUT
030076      030076      104405      000000      000000      TYPDS              ;;GO TYPE--DECIMAL ASCII WITH SIGN
030100      030100      104401      030130      000000      TYPE $ENULL        ;;TYPE A NULL CHARACTER
030104      030104      013700      000042      000000      $GET42: MOV @#42,R0  ;;GET MONITOR ADDRESS
030110      030110      001405      000000      000000      BEQ $DOAGN         ;;BRANCH IF NO MONITOR
030112      030112      000005      000000      000000      RESET             ;;CLEAR THE WORLD
030114      030114      004710      000000      000000      $ENDAD: JSR PC,(R0) ;;GO TO MONITOR
030116      030116      000240      000000      000000      NOP               ;;SAVE ROOM
030120      030120      000240      000000      000000      NOP               ;;FOR
030122      030122      000240      000000      000000      NOP               ;;ACT11
030124      030124      000137      000000      000000      $DOAGN: JMP @ (PC)+  ;;RETURN
030126      030126      005060      000000      000000      $RTNAD: .WORD LOOP
030130      030130      377      377      000      $ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
030133      030133      015      012      105      $ENDMG: .ASCIZ <15><12>/END PASS #/
030136      030136      116      104      040
030141      030141      120      101      123
030144      030144      123      040      123
030147      030147      000      000      043
    
```

4606

```

.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
030150      030150      104407      000000      000000      $SCOPE: CKSWR      ;;TEST FOR CHANGE IN SOFT-SWR
    
```

SCOPE HANDLER ROUTINE

```

030152 052737 001000 177520      BIJ #1000,BCSR ;ENABLE
030160 032777 040000 150752 1$: BIT #BIT14,@SWR      ;;LOOP ON PRESENT TEST?
030166 001117      BNE $OVER          ;;YES IF SW14=1
                        ;####START OF CODE FOR THE XOR TESTER####
030170 000416      XTSTR: BR 6$      ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                        ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
030172 013746 000004      MOV @#ERRVEC, -(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
030176 012737 030216 000004      MOV #5,@#ERRVEC    ;;SET FOR TIMEOUT
030204 005737 177060      TST @#177060      ;;TIME OUT ON XOR?
030210 012637 000004      MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
030214 000466      BR $SVLAD        ;;GO TO THE NEXT TEST
030216 022626      5$: CMP (SP)+,(SP)+  ;;CLEAR THE STACK AFTER A TIME OUT
030220 012637 000004      MOV (SP)+,@#ERRVEC ;;RESTORE THE ERROR VECTOR
030224 000426      BR 7$          ;;LOOP ON THE PRESENT TEST
030226      6$: ;####END OF CODE FOR THE XOR TESTER####
030226 032777 000400 150704      BIT #BIT08,@SWR    ;;LOOP ON SPEC. TEST?
030234 001407      BEQ 2$          ;;BR IF NO
030236 017746 150676      MOV @SWR, -(SP)    ;;SET DESIRED TEST NUM. FROM SWR
030242 042716 000300      BIC #SWRMK, (SP)  ;;STRIP AWAY UNDESIRED BITS
030246 122637 001102      CMPB (SP)+,$STNM   ;;ON THE RIGHT TEST?
030252 001465      BEQ $OVER        ;;BR IF YES
030254 105737 001103      2$: TSTB $ERFLG    ;;HAS AN ERROR OCCURRED?
030260 001421      BEQ 3$          ;;BR IF NO
030262 123777 001115 001103      CMPB $ERMAX,$ERFLG ;;MAX. ERRORS FOR THIS TEST OCCURRED?
030270 101015      BHI 3$          ;;BR IF NO
030272 032777 001000 150640      BIT #BIT09,@SWR   ;;LOOP ON ERROR?
030300 001404      BEQ 4$          ;;BR IF NO
030302 013777 001110 001106      7$: MOV $LPERR,$LPADR  ;;SET LOOP ADDRESS TO LAST SCOPE
030310 000446      BR $OVER
030312 105037 001103      4$: CLRB $ERFLG      ;;ZERO THE ERROR FLAG
030316 005037 001164      CLR $TIMES       ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
030322 000415      BR 1$          ;;ESCAPE TO THE NEXT TEST
030324 032777 004000 150606      3$: BIT #BIT11,@SWR  ;;INHIBIT ITERATIONS?
030332 001011      BNE 1$          ;;BR IF YES
030334 005737 001206      TST $PASS        ;;IF FIRST PASS OF PROGRAM
030340 001406      BEQ 1$          ;;INHIBIT ITERATIONS
030342 005237 001104      INC $ICNT        ;;INCREMENT ITERATION COUNT
030346 023737 001164 001104      CMP $TIMES,$ICNT  ;;CHECK THE NUMBER OF ITERATIONS MADE
030354 002024      BGE $OVER        ;;BR IF MORE ITERATION REQUIRED
030356 012737 000001 001104      1$: MOV #1,$ICNT    ;;REINITIALIZE THE ITERATION COUNTER
030364 013737 030450 001164      MOV $MXCNT,$TIMES  ;;SET NUMBER OF ITERATIONS TO DO
030372 105237 001102      $SVLAD: INCB $STNM     ;;COUNT TEST NUMBERS
030376 113737 001102 001204      MOVB $STNM,$TESTN  ;;SET TEST NUMBER IN APT MAILBOX
030404 011637 001106      MOV (SP),$LPADR    ;;SAVE SCOPE LOOP ADDRESS
030410 011637 001110      MOV (SP),$LPERR    ;;SAVE ERROR LOOP ADDRESS
030414 005037 001166      CLR $ESCAPE      ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
030420 112737 000001 001115      MOVB #1,$ERMAX    ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
030426 013777 001102 150506      $OVER: MOV $STNM,@DISPLAY ;;DISPLAY TEST NUMBER
030434 013716 001106      MOV $LPADR,(SP)   ;;FUDGE RETURN ADDRESS
030440 042737 001000 177520      BIC #1000,BCSR ;DISABLE
030446 000002      RTI
030450 000001      $MXCNT: 1      ;;MAX. NUMBER OF ITERATIONS

```

4607

```

.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

```


ABORT ROUTINE FOR LCP/ORION UFD MODE

```

030706 001443          BEQ      ABORTZ      ;JUST GO BACK 0 CHAIN IF IT IS (NO ERROR)
030710 020027 000003  CMP      RO,#3      ;IS IS A +C ?
030714 001404          BEQ      ABORTC      ;BR TO LOAD +C ON XXDP+ STACK (NO ERROR)
030716 005737 004150  TST      UQUIET      ;TEST FOR USER QUIET MODE
030722 001443          BEQ      NOABRT     ;IF FIELD-SERVICE MODE, CONTINUE NORMAL OPERATION
                                ; BECAUSE FIELD-SERVICE MODE DOES NOT QUIT ON ERROR
                                ;SET DRERR THEN LEAVE
030724 000422          BR       ABORTE     ;RESTORE EMT LOCATION (30)
030726 013737 004142 000030 ABORTC: MOV    SAV30,30 ;RESTORE EMT PRIORITY LOCATION (32)
030734 013737 004144 000032  MOV    SAV32,32 ;GET XXDP STACK LOC. INTO RO FROM MONITOR
030742 104043          EMT      +43      ;FIND END OF STACK
030744 105720          1$:  TSTB    (RO)+
030746 001376          BNE     1$
030750 112760 000057 177777  MOVB   #' /, 1(RO) ;LOAD SLASH OVER ZERO
030756 112720 000136          MOVB   #' +,(RO)+ ;LOAD UPARROW
030762 112720 000103          MOVB   #' C,(RO)+ ;LOAD C
030766 105010          CLRB   (RO)      ;MAKE NEW END TO STACK
030770 000412          BR       ABORTZ     ;NOW LEAVE
030772 013737 004142 000030 ABORTE: MOV    SAV30,30 ;RESTORE EMT LOCATION (30)
031000 013737 004144 000032  MOV    SAV32,32 ;RESTORE EMT PRIORITY LOCATION (32)
031006 104042          EMT      +42      ;GET DCA LOCATION INTO RO FROM MONITOR
031010 012760 177777 000042  MOV    #-1,42(RO) ;SET A -1 INTO LOCATION DRERR IN MONITOR
031016 013700 000042  ABORTZ: MOV    @#42,RO ;AND PUT THE MONITOR RETURN ADDRESS IN RO
031022 005037 000042          CLR    @#42      ;CLEAR MONITOR RETURN FLAG
031026 000137 030114  NOABRT: JMP    $ENDAD ;RETURN TO MONITOR-DO NOT PUSH STACK HERE
031032 000207          RTS     PC      ;IF NOTUFD RETURN TO MAINLINE

4608 .SBTTL APT COMMUNICATIONS ROUTINE
*****
031034 112737 000001 031300 $ATY1: MOVB   #1,$FFLG ;:TO REPORT FATAL ERROR
031042 112737 000001 031276 $ATY3: MOVB   #1,$MFLG ;:TO TYPE A MESSAGE
031050 000403          BR       $ATYC
031052 112737 000001 031300 $ATY4: MOVB   #1,$FFLG ;:TO ONLY REPORT FATAL ERROR
031060 $ATYC:
031060 010046          MOV    RO,-(SP) ;:PUSH RO ON STACK
031062 010146          MOV    R1,-(SP) ;:PUSH R1 ON STACK
031064 105737 031276          TSTB   $MFLG ;:SHOULD TYPE A MESSAGE?
031070 001450          BEQ    5$ ;:IF NOT: BR
031072 122737 000001 001220 CMPB   #APTENV,$ENV ;:OPERATING UNDER APT?
031100 001031          BNE   3$ ;:IF NOT: BR
031102 132737 000100 001221 BITB   #APTPOOL,$ENVM ;:SHOULD SPOOL MESSAGES?
031110 001425          BEQ    3$ ;:IF NOT: BR
031112 017600 000004          MOV    @4(SP),RO ;:GET MESSAGE ADDR.
031116 062766 000002 000004 ADD    #2,4(SP) ;:BUMP RETURN ADDR.
031124 005737 001200          1$:  TST    $MSGTYPE ;:SEE IF DONE W/ LAST XMISSION?
031130 001375          BNE   1$ ;:IF NOT: WAIT
031132 010037 001214          MOV    RO,$MSGAD ;:PUT ADDR IN MAILBOX
031136 105720          2$:  TSTB   (RO)+ ;:FIND END OF MESSAGE
031140 001376          BNE   2$
031142 163700 001214          SUB    $MSGAD,RO ;:SUB START OF MESSAGE
031146 006200          ASR   RO ;:GET MESSAGE LENGTH IN WORDS
031150 010037 001216          MOV    RO,$MSGLGT ;:PUT LENGTH IN MAILBOX
031154 012737 000004 001200 MOV    #4,$MSGTYPE ;:TELL APT TO TAKE MSG.
031162 000413          BR       5$
031164 017637 000004 031210 3$:  MOV    @4(SP),4$ ;:PUT MSG ADDR IN JSR LINKAGE
031172 062766 000002 000004 ADD    #2,4(SP) ;:BUMP RETURN ADDRESS
031200 013746 177776          MOV    177776,(SP) ;:PUSH 177776 ON STACK
031204 004737 031302          JSR   PC,$TYPE ;:CALL TYPE MACRO
031210 000000          4$:  .WORD  0

```


APT COMMUNICATIONS ROUTINE

```

031212 031212 105737 031300 5$:
031212 001416 10$: TSTB $FFLG ;; SHOULD REPORT FATAL ERROR?
031220 005737 001220 BEQ 12$ ;; IF NOT: BR
031224 001413 TST $ENV ;; RUNNING UNDER APT?
031226 005737 001200 11$: TST $MSGTYPE ;; FINISHED LAST MESSAGE?
031232 001375 BNE 11$ ;; IF NOT: WAIT
031234 017637 000004 001202 MOV @4(SP), $FATAL ;; GET ERROR #
031242 062766 000002 000004 ADD #2, 4(SP) ;; BUMP RETURN ADDR.
031250 005237 001200 INC $MSGTYPE ;; TELL APT TO TAKE ERRCR
031254 105037 031300 12$: CLRB $FFLG ;; CLEAR FATAL FLAG
031260 105037 031277 CLRB $LFLG ;; CLEAR LOG FLAG
031264 105037 031275 CLRB $MFLG ;; CLEAR MESSAGE FLAG
031270 012601 MOV (SP)+, R1 ;; POP STACK INTO R1
031272 012600 MOV (SP)+, R0 ;; POP STACK INTO R0
031274 000207 RTS PC ;; RETURN
031276 000 $MFLG: .BYTE 0 ;; MESSG. FLAG
031277 000 $LFLG: .BYTE 0 ;; LOG FLAG
031300 000 $FFLG: .BYTE 0 ;; FATAL FLAG
        .EVEN

```

000200
000001
000100
000040

4609

APTSIZE=200
APTENV=001
APTSPOOL=100
APTCSUP=040
.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

```

031302 105737 001157 $TYPE: TSTB $TPFLG ;; IS THERE A TERMINAL?
031306 100002 BPL 1$ ;; BR IF YES
031310 000000 HALT ;; HALT HERE IF NO TERMINAL
031312 000430 BR 3$ ;; LEAVE
031314 010046 1$: MOV R0, (SP) ;; SAVE R0
031316 017600 000002 MOV @2(SP), R0 ;; GET ADDRESS OF ASCIZ STRING
031322 122737 000001 001220 CMPB #APTENV, $ENV ;; RUNNING IN APT MODE
031330 001011 BNE 62$ ;; NO, GO CHECK FOR APT CONSOLE
031332 132737 000100 001221 BITB #APTSPOOL, $ENVM ;; SPOOL MESSAGE TO APT
031340 001405 BEQ 62$ ;; NO, GO CHECK FOR CONSOLE
031342 010037 031352 MOV R0, 61$ ;; SETUP MESSAGE ADDRESS FOR APT
031346 004737 031042 JSR PC, $ATY3 ;; SPOOL MESSAGE TO APT
031352 000000 61$: .WORD 0 ;; MESSAGE ADDRESS
031354 132737 000040 001221 62$: BITB #APTCSUP, $ENVM ;; APT CONSOLE SUPPRESSED
031362 001003 BNE 60$ ;; YES, SKIP TYPE OUT
031364 112046 2$: MOVB (R0)+, (SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
031366 001005 BNE 4$ ;; BR IF IT ISN'T THE TERMINATOR
031370 005726 TST (SP)+ ;; IF TERMINATOR POP IT OFF THE STACK

```

TYPE ROUTINE

```

031372 012600          60$:  MOV    (SP)+,R0      ;;RESTORE R0
031374 062716 000002  3$:  ADD    #2,(SP)      ;;ADJUST RETURN PC
031400 000002          RTI                    ;;RETURN
031402 122716 000011  4$:  CMPB   #HT,(SP)      ;;BRANCH IF <HT>
031406 001430          BEQ    8$                    ;;
031410 122716 000200          CMPB   #CRLF,(SP)    ;;BRANCH IF NOT <CRLF>
031414 001006          BNE    5$                    ;;
031416 005726          TST    (SP)+          ;;POP <CR><LF> EQUIV
031420 104401          TYPE                    ;;TYPE A CR AND LF
031422 001175          $CRLF
031424 105037 031632          CLRB   $CHARCNT      ;;CLEAR CHARACTER COUNT
031430 000755          BR     2$                    ;;GET NEXT CHARACTER
031432 004737 031514  5$:  JSR    PC,$TYPEC      ;;GO TYPE THIS CHARACTER
031436 123726 001156  6$:  CMPB   $FILLC,(SP)+  ;;IS IT TIME FOR FILLER CHARS.?
031442 001350          BNE    2$                    ;;IF NO GO GET NEXT CHAR.
031444 013746 001154          MOV    $NULL,-(SP)   ;;GET # OF FILLER CHARS. EDED
;;AND THE NULL CHAR.
031450 105366 000001  7$:  DECB   1(SP)          ;;DOES A NULL NEED TO BE TYPED?
031454 002770          BLT    6$                    ;;BR IF NO -GO POP THE NULL OFF OF STACK
031456 004737 031514          JSR    PC,$TYPEC      ;;GO TYPE A NULL
031462 105337 031632          DECB   $CHARCNT      ;;DO NOT COUNT AS A COUNT
031466 000770          BR     7$                    ;;LOOP
031470 112716 000040          .HORIZONTAL TAB PROCESSOR
031474 004737 031514  8$:  MOVB   #' (SP)        ;;REPLACE TAB WITH SPACE
031500 132737 000007 031632  9$:  JSR    PC,$TYPEC      ;;TYPE A SPACE
031506 001372          BITB   #7,$CHARCNT   ;;BRANCH IF NOT AT
031510 005726          BNE    9$                    ;;TAB STOP
031512 000724          TST    (SP)+          ;;POP SPACE OFF STACK
031514          BR     2$                    ;;GET NEXT CHARACTER
031514 105777 147424          $TYPEC:  TSTB   @TKS          ;;CHAR IN KYBD BUFFER?
031520 100022          BPL    10$             ;;BR IF NOT
031522 017746 147420          MOV    @TKB,-(SP)    ;;GET CHAR
031526 042716 177600          BIC    #177600,(SP)  ;;STRIP EXTRANEIOUS BITS
031532 122716 000023          CMPB   #XOFF,(SP)    ;;WAS CHAR XOFF
031536 001012          BNE    102$          ;;BR IF NOT
031540          101$:  TSTB   @TKS          ;;WAIT FOR CHAR
031540 105777 147400          BPL    101$          ;;BR IF NOT
031544 100375          MOVB   @TKB,(SP)    ;;GET CHAR
031546 117716 147374          BIC    #177600,(SP)  ;;STRIP IT
031552 042716 177600          CMPB   #XON,(SP)    ;;WAS IT XON?
031556 122716 000021          BNE    101$          ;;BR IF NOT
031562 001366          102$:  TST    (SP)+          ;;FIX STACK
031564          10$:  TST    (SP)+          ;;FIX STACK
031566          TSTB   @TPS          ;;WAIT UNTIL PRINTER IS READY
031572 100375          BPL    10$             ;;BR IF NOT
031574 116677 000002 147350          MOVB   2(SP),@TPB    ;;LOAD CHAR TO BE TYPED INTO DATA REG.
031602 122766 000015 000002          CMPB   #CR,2(SP)    ;;IS CHARACTER A CARRIAGE RETURN?
031610 001003          BNE    1$                    ;;BRANCH IF NO
031612 105037 031632          CLRB   $CHARCNT      ;;YES--CLEAR CHARACTER COUNT
031616 000406          BR     $TYPEX          ;;EXIT
031620 122766 000012 000002  1$:  CMPB   #LF,2(SP)    ;;IS CHARACTER A LINE FEED?
031626 001402          BEQ    $TYPEX          ;;BRANCH IF YES
031630 105227          INCB   (PC)+          ;;COUNT THE CHARACTER
031632 000000          $CHARCNT: .WORD 0  ;;CHARACTER COUNT STORAGE

```

TYPE ROUTINE

4610 031634 000207

031636 017646 000000
 031642 116637 000001
 031650 112637 032063
 031654 062716 000002
 031660 000406
 031662 112737 000001
 031670 112737 000006
 031676 112737 000005
 031704 010346
 031706 010446
 031710 010546
 031712 113704 032063
 031716 005404
 031720 062704 000006
 031724 110437 032062
 031730 113704 032061
 031734 016605 000012
 031740 005003
 031742 006105
 031744 000404
 031746 006105
 031750 006105
 031752 006105
 031754 010503
 031756 006103
 031760 105337 032062
 031764 100016
 031766 042703 177770
 031772 001002
 031774 005704
 031776 001403
 032000 005204
 032002 052703 000060

032061
 032061
 032063
 032060

```

$TYPEX: RTS      PC
.SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
;*****
;THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT
;OCTAL (ASCII) NUMBER AND TYPE IT.
;$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
        MOV      1(SP), $OFILL    ;;LOAD ZERO FILL SWITCH
        MOV      (SP)+, $OMODE+1  ;;NUMBER OF DIGITS TO TYPE
        ADD      #2, (SP)        ;;ADJUST RETURN ADDRESS
        BR       $TYPON
;$TYPOC: MOV      #1, $OFILL     ;;SET THE ZERO FILL SWITCH
        MOV      #6, $OMODE+1    ;;SET FOR SIX(6) DIGITS
;$TYPON: MOV      #5, $OCNT      ;;SET THE ITERATION COUNT
        MOV      R3,-(SP)        ;;SAVE R3
        MOV      R4,-(SP)        ;;SAVE R4
        MOV      R5,-(SP)        ;;SAVE R5
        MOV      $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
        NEG      R4
        ADD      #6,R4          ;;SUBTRACT IT FOR MAX. ALLOWED
        MOV      R4,$OMODE      ;;SAVE IT FOR USE
        MOV      $OFILL,R4      ;;GET THE ZERO FILL SWITCH
        MOV      12(SP),R5     ;;PICKUP THE INPUT NUMBER
        CLR      R3            ;;CLEAR THE OUTPUT WORD
1$:     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
        DECB    $OMODE        ;;TYPE THIS DIGIT?
        BPL     7$            ;;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

4611

```

032006 052703 000040      5$:   BIS      #,R3      ;;MAKE ASCII IF NOT ALREADY
032012 110337 032056      MOV      R3,8$     ;;SAVE FOR TYPING
032016 104401 032056      TYPE     ,8$      ;;GO TYPE THIS DIGIT
032022 105337 032060      7$:   DECB     $OCNT  ;;COUNT BY 1
032026 003347          BGT      2$       ;;BR IF MORE TO DO
032030 002402          BLT      6$       ;;BR IF DONE
032032 005204          INC      R4       ;;INSURE LAST DIGIT ISN'T A BLANK
032034 000744          BR       2$       ;;GO DO THE LAST DIGIT
032036 012605      6$:   MOV      (SP)+,R5  ;;RESTORE R5
032040 012604          MOV      (SP)+,R4  ;;RESTORE R4
032042 012603          MOV      (SP)+,R3  ;;RESTORE R3
032044 016666 000002 000004  MO      2(SP),4(SP) ;;SET THE STACK FOR RETURNING
032052 012616          MO      (SP)+,(SP)
032054 000002          RTI          ;;RETURN
032056 000          8$:   BYTE     0      ;;STORAGE FOR ASCII DIGIT
032057 000          .BYTE   0      ;;TERMINATOR FOR TYPE ROUTINE
032060 000          $OCNT. .BYTE 0      ;;OCTAL DIGIT COUNTER
032061 000          $OFILL: .BYTE 0     ;;ZERO FILL SWITCH
032062 000000          $OMODE: .WORD 0    ;;NUMBER OF DIGITS TO TYPE
.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
$TYPDS:
032064          MOV      R0,-(SP)  ;;PUSH R0 ON STACK
032064 010046          MOV      R1,-(SP)  ;;PUSH R1 ON STACK
032066 010146          MOV      R2,-(SP)  ;;PUSH R2 ON STACK
032070 010246          MOV      R3,-(SP)  ;;PUSH R3 ON STACK
032072 010346          MOV      R5,-(SP)  ;;PUSH R5 ON STACK
032074 010546          MOV      #20200,-(SP) ;;SET BLANK SWITCH AND SIGN
032076 012746 020200          MOV      20(SP),R5  ;;GET THE INPUT NUMBER
032102 016605 000020          BPL      1$       ;;BR IF INPUT IS POS.
032106 100004          NEG      R5       ;;MAKE THE BINARY NUMBER POS.
032110 005405          MOV      #' ,1(SP)  ;;MAKE THE ASCII NUMBER NEG.
032112 112766 000055 000001  1$:   CLR      R0       ;;ZERO THE CONSTANTS INDEX
032120 005000          MOV      #$DBLK,R3  ;;SETUP THE OUTPUT POINTER
032122 012703 032300          MOV      #' ,(R3)+  ;;SET THE FIRST CHARACTER TO A BLANK
032126 112723 000040          2$:   CLR      R2       ;;CLEAR THE BCD NUMBER
032132 005002          MOV      $DTBL(R0),R1 ;;GET THE CONSTANT
032134 016001 032270          3$:   SUB      R1,R5    ;;FORM THIS BCD DIGIT
032140 160105          BLT      4$       ;;BR IF DCNE
032142 002402          INC      R2       ;;INCREASE THE BCD DIGIT BY 1
032144 005202          BR       3$
032146 000774          4$:   ADD      R1,R5    ;;ADD BACK THE CONSTANT
032150 060105          TST      R2       ;;CHECK IF BCD DIGIT=0
032152 005702          BNE     5$       ;;FALL THROUGH IF 0
032154 001002          TSTB    (SP)      ;;STILL DOING LEADING 0'S?
032156 105716          BMI     7$       ;;BR IF YES
032160 100407          5$:   ASLB    (SP)      ;;MSD?
032162 106316          BCC     6$       ;;BR IF NO
032164 103003          MOVB   1(SP),1(R3) ;;YES--SET THE SIGN
032166 116663 000001 177777

```

CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

```

032174 052702 000060      6$:  BIS      #'0,R2      ;;MAKE THE BCD DIGIT ASCII
032200 052702 000040      7$:  BIS      #' R2      ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
032204 110223              MOVVB   R2,(R3)+    ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
032206 005720              TST      (R0)+      ;;JUST INCREMENTING
032210 020027 000010      CMP      R0,#10    ;;CHECK THE TABLE INDEX
032214 002746              BLT      2$        ;;GO DO THE NEXT DIGIT
032216 003002              BGT      8$        ;;GO TO EXIT
032220 010502              MOV      R5,R2     ;;GET THE LSD
032222 000764              BR       6$        ;;GO CHANGE TO ASCII
032224 105726      8$:  TSTB   (SP)+    ;;WAS THE LSD THE FIRST NON-ZERO?
032226 100003              BPL      9$        ;;BR IF NO
032230 116663 177777 177776 MOVVB   1(SP),-2(R3) ;;YES--SET THE SIGN FOR TYPING
032236 105013      9$:  CLRB   (R3)     ;;SET THE TERMINATOR
032240 012605              MOV      (SP)+,R5  ;;POP STACK INTO R5
032242 012603              MOV      (SP)+,R3  ;;POP STACK INTO R3
032244 012602              MOV      (SP)+,R2  ;;POP STACK INTO R2
032246 012601              MOV      (SP)+,R1  ;;POP STACK INTO R1
032250 012600              MOV      (SP)+,R0  ;;POP STACK INTO R0
032252 104401 032300      TYPE   $DBLK    ;;NOW TYPE THE NUMBER
032256 016666 000002 000004 MOV      2(SP),4(SP) ;;ADJUST THE STACK
032264 012616              MOV      (SP)+,(SP)
032266 000002              RTI                    ;;RETURN TO USER
032270 023420      $DTBL: 10000.
032272 001750              1000.
032274 000144              100.
032276 000012              10.
032300

```

4612

```

$DBLK: .BLKW 4
.SBTTL TTY INPUT ROUTINE
;*****
.ENABLE 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.

```

```

032310 022737 000176 001140 $CKSWR: CMP      #SWREG,SWR ;;IS THE SOFT-SWR SELECTED?
032316 001074              BNE      15$      ;;BRANCH IF NO
032320 105777 146620              TSTB   @TKS      ;;CHAR THERE?
032324 100071              BPL      15$      ;;IF NO, DON'T WAIT AROUND
032326 117746 146614              MOVVB  @TKB,-(SP) ;;SAVE THE CHAR
032332 042716 177600              BIC    #+C177,(SP) ;;STRIP-OFF THE ASCII
032336 022726 000007              CMP      #7,(SP)+ ;;IS IT A CONTROL G?
032342 001062              BNE      15$      ;;NO, RETURN TO USER
032344 123727 001134 000001 CMPB    $AUTOB,#1 ;;ARE WE RUNNING IN AUTO MODE?
032352 001456              BEQ      15$      ;;BRANCH IF YES
032354 104401 033045              TYPE   , $CNTLG  ;;ECHO THE CONTROL-G (+G)
032360 104401 033052      $GTSWR: TYPE   $MSWR  ;;TYPE CURRENT CONTENTS
032364 013746 000176              MOV      SWREG,-(SF) ;;SAVE SWREG FOR TYPEOUT
032370 104402              TYPOC                ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
032372 104401 033063              TYPE   , $MNEW    ;;PROMPT FOR NEW SWR
032376 005046      19$:  CLR      -(SP)    ;;CLEAR COUNTER
032400 005046              CLR      -(SP)    ;;THE NEW SWR
032402 105777 146536      7$:  TSTB   @TKS      ;;CHAR THERE?
032406 100375              BPL      7$        ;;IF NOT TRY AGAIN
032410 117746 146532              MOVVB  @TKB,-(SP) ;;PICK UP CHAR
032414 042716 177600              BIC    #+C177,(SP) ;;MAKE IT 7-BIT ASCII
032420 021627 000025      9$:  CMP      (SP),#25 ;;IS IT A CONTROL-U?

```

TTY INPUT ROUTINE

```

032424 001005          BNE      10$          ;; BRANCH IF NOT
032426 104401 033040   TYPE      $CNTLU      ;; YES, ECHO CONTROL-L (+U)
032432 062706 000006   20$:    ADD      #6,SP          ;; IGNORE PREVIOUS INPUT
032436 000757          BR       19$          ;; LET'S TRY IT AGAIN
032440 021627 000015   10$:    CMP      (SP),#15      ;; IS IT A <CR>?
032444 001022          BNE      16$          ;; BRANCH IF NO
032446 005766 000004   TST      4(SP)          ;; YES, IS IT THE FIRST CHAR?
032452 001403          BEQ      11$          ;; BRANCH IF YES
032454 016677 000002 146456   MOV      2(SP),@SWR     ;; SAVE NEW SWR
032462 062706 000006   11$:    ADD      #6,SF          ;; CLEAR UP STACK
032466 104401 001175   14$:    TYPE      $CRLF          ;; ECHO <CR> AND <LF>
032472 123727 001135 000001   CMPB     $INTAG,#1     ;; RE-ENABLE TTY KBD INTERRUPTS?
032500 001003          BNE      15$          ;; BRANCH IF NOT
032502 012777 000100 146434   MOV      #100,@$TKS    ;; RE-ENABLE TTY KBD INTERRUPTS
032510 000002          RTI          ;; RETURN
032512 004737 031514   15$:    JSR      PC,$TYPEC     ;; ECHO CHAR
032516 021627 000060   16$:    CMP      (SP),#60     ;; CHAR < 0?
032522 002420          BLT      18$          ;; BRANCH IF YES
032524 021627 000067   CMP      (SP),#67     ;; CHAR > 7?
032530 003015          BGT      18$          ;; BRANCH IF YES
032532 042726 000060   BIC      #60,(SP)+     ;; STRIP-OFF ASCII
032536 005766 000002   TST      2(SP)          ;; IS THIS THE FIRST CHAR
032542 001403          BEQ      17$          ;; BRANCH IF YES
032544 006316          ASL      (SP)          ;; NO, SHIFT PRESENT
032546 006316          ASL      (SP)          ;; CHAR OVER TO MAKE
032550 006316          ASL      (SP)          ;; ROOM FOR NEW ONE.
032552 005266 000002   17$:    INC      2(SP)          ;; KEEP COUNT OF CHAR
032556 056016 177776   BIS      -2(SP),(SP)   ;; SET IN NEW CHAR
032562 000707          BR       7$          ;; GET THE NEXT ONE
032564 104401 001174   18$:    TYPE      $QUES          ;; TYPE ?<CR><LF>
032570 000720          BR       20$          ;; 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

032572 011646          $RDCHR: MOV      (SP),-(SP)    ;; PUSH DOWN THE PC
032574 016666 000004 000002   MOV      4(SP),2(SP)   ;; SAVE THE PS
032602 105777 146336   1$:    TSTB     @$TKS          ;; WAIT FOR
032606 100375          BPL      1$          ;; A CHARACTER
032610 117766 146332 000004   MOVB     @$TKB,4(SP)   ;; READ THE TTY
032616 042766 177600 000004   BIC      #+C<177>,4(SP) ;; GET RID OF JUNK IF ANY
032624 026627 000004 000023   CMP      4(SP),#23     ;; IS IT A CONTROL-S?
032632 001013          BNE      3$          ;; BRANCH IF NO
032634 105777 146304   2$:    TSTB     @$TKS          ;; WAIT FOR A CHARACTER
032640 100375          BPL      2$          ;; LOOP UNTIL ITS THERE
032642 117746 146300   MOVB     @$TKB,-(SP)   ;; GET CHARACTER
032646 042716 177600   BIC      #+C177,(SP)   ;; MAKE IT 7-BIT ASCII
032652 022627 000021   CMP      (SP)+,#21     ;; IS IT A CONTROL-Q?
032656 001366          BNE      2$          ;; IF NOT DISCARD IT
032660 000750          BR       1$          ;; YES, RESUME
032662 026627 000004 000021 3$:    CMP      4(SP),#$XON   ;; IS IT A RANDOM XON?
032670 001744          BEQ      1$          ;; BRANCH IF YES
032672 026627 000004 000140   CMP      4(SP),#140    ;; IS IT UPPER CASE?
;RAN001
;RAN001

```

TTY INPUT ROUTINE

```

032700 002407          BLT      4$          ;;BRANCH IF YES
032702 026627 000004 000175    CMP      4(SP),#175      ;;IS IT A SPECIAL CHAR?
032710 003003          BGT      4$          ;;BRANCH IF YES
032712 042766 000040 000004    BIC      #40,4(SP)      ;;MAKE IT UPPER CASE
032720 000002          RTI          ;;GO BACK TO USER
4$:
;*****
;THIS ROUTINE WILL INPUT A STRING FROM THE TTY
;CALL:
;*
;*   RDLIN
;*   RETURN HERE
;*
;RDLIN: MOV      R3,(SP)          ;;INPUT A STRING FROM THE TTY
1$: MOV      #$$TTYIN,R3        ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2$: CMP      #$$TTYIN+8.,R3     ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
   BLOS     4$                  ;;SAVE R3
   RDCHR    (SP)+,(R3)          ;;GET ADDRESS
   MOV      #177,(R3)          ;;BUFFER FULL?
10$: CMPB   #177,(R3)          ;;BR IF YES
   BNE     3$                  ;;GO READ ONE CHARACTER FROM THE TTY
4$: TYPE   $QUES                ;;GET CHARACTER
   BR      1$                  ;;IS IT A RUBOUT
3$: MOV      (R3),9$           ;;SKIP IF NOT
   TYPE   9$                  ;;TYPE A '?'
   CMPB   #15,(R3)+           ;;CLEAR THE BUFFER AND LOOP
   BNE     2$                  ;;ECHO THE CHARACTER
   CLRB   -1(R3)              ;;CHECK FOR RETURN
   TYPE   $LF                 ;;LOOP IF NOT RETURN
   MOV    (SP)+,R3            ;;CLEAR RETURN (THE 15)
   MOV    (SP)-,(SP)          ;;TYPE A LINE FEED
   MOV    4(SP),2(SP)         ;;RESTORE R3
   MOV    #$$TTYIN,4(SP)     ;;ADJUST THE STACK AND PUT ADDRESS OF THE
   RTI                          ;;FIRST ASCII CHARACTER ON IT
9$: .BYTE 0                    ;;RETURN
   .BYTE 0                    ;;STORAGE FOR ASCII CHAR. TO TYPE
$TTYIN: .BLKB 8                ;;TERMINATOR
$CNTLU: .ASCIZ /+U/<15><12>    ;;RESERVE 8 BYTES FOR TTY INPUT
$CNTLG: .ASCIZ /+G/<15><12>    ;;CONTROL "U"
$MSWR:  .ASCIZ <15><12>/SWR = /
$MNEW:  .ASCIZ / NEW = /

```

4613

```

.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
;*   RETURN HERE
;*
;RDOCT: MOV      (SP)-,(SP)    ;;READ AN OCTAL NUMBER
033074 011646          MOV      4(SP),2(SP)      ;;LOW ORDER BITS ARE ON TOP OF THE STACK
033076 016666 000004 000002    MOV      R0,-(SP)          ;;HIGH ORDER BITS ARE IN $HIOCT
033104 010046          MOV      R1,-(SP)          ;;PROVIDE SPACE FOR THE
033106 010146          MOV      R1,-(SP)          ;;INPUT NUMBER
   ;;PUSH R0 ON STACK
   ;;PUSH R1 ON STACK

```

READ AN OCTAL NUMBER FROM THE TTY

```

033110 010246          MOV      R2, (SP)          ;; PUSH R2 ON STACK
033112 104411          RDLIN          ;; READ AN ASCIZ LINE
033114 012600          MOV      (SP)+, R0          ;; GET ADDRESS OF 1ST CHARACTER
033116 005001          CLR      R1              ;; CLEAR DATA WORD
033120 005002          CLR      R2
033122 112046          2$:      MOVVB   (R0)+, -(SP)      ;; PICKUP THIS CHARACTER
033124 001412          BEQ      3$              ;; IF ZERO GET OUT
033126 006301          ASL     R1              ;; *2
033130 006102          ROL     R2
033132 006301          ASL     R1              ;; *4
033134 006102          ROL     R2
033136 006301          ASL     R1              ;; *8
033140 006102          ROL     R2
033142 042716 177770  BIC     #C7, (SP)          ;; STRIP THE ASCII JUNK
033146 062601          ADD     (SP)+, R1          ;; ADD IN THIS DIGIT
033150 000764          BR      2$              ;; LOOP
033152 005726          3$:      TST     (SP)+          ;; CLEAN TERMINATOR FROM STACK
033154 010166 000012  MOV     R1, 12(SP)          ;; SAVE THE RESULT
033160 010237 033174  MOV     R2, $HIOCT
033164 012602          MOV     (SP)+, R2          ;; POP STACK INTO R2
033166 012601          MOV     (SP)+, R1          ;; POP STACK INTO R1
033170 012600          MOV     (SP)+, R0          ;; POP STACK INTO R0
033172 000002          RTI
033174 000000          $HIOCT: WORD 0          ;; HIGH ORDER BITS GO HERE

```

4614

```

.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.

```

```

033176 010046          $TRAP:  MOV     R0, -(SP)          ;; SAVE R0
033200 016600 000002  MOV     2(SP), R0          ;; GET TRAP ADDRESS
033204 005740          TST     -(R0)            ;; BACKUP BY 2
033206 111000          MOVVB   (R0), R0          ;; GET RIGHT BYTE OF TRAP
033210 006300          ASL     R0              ;; POSITION FOR INDEXING
033212 016000 033232  MOV     $TRPAD(R0), R0    ;; INDEX TO TABLE
033216 000200          RTS     R0              ;; GO TO ROUTINE

```

```

033220 011646          ;; THIS IS USE TO HANDLE THE GETPRI" MACRO
033222 016666 000004 000002 $TRAP2: MOV     (SP), -(SP)      ;; MOVE THE PC DOWN
033230 000002          MOV     4(SP), 2(SP)     ;; MOVE THE PSW DOWN
                                RTI                          ;; RESTOR THE PSW

```

```

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

```

```

033232 033220          $TRPAD: .WORD  $TRAP2
033234 031302          $TYPE   ;; CALL=TYPE      TRAP+1(104401) TTY TYPEOUT ROUTINE
033236 031662          $TYPOC  ;; CALL=TYPOC     TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
033240 031636          $TYPOS  ;; CALL=TYPOS     TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
033242 031676          $TYPON  ;; CALL=TYPON     TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
033244 032064          $TYPDS  ;; CALL=TYPDS     TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
033246 032360          $GTSWR  ;; CALL=GTSWR     TRAP+6(104406) GET SOFT-SWR SETTING
033250 032310          $CKSWR  ;; CALL=CKSWR     TRAP+7(104407) TEST FOR CHANGE IN SOFT SWR
033252 032572          $RDCHR  ;; CALL=RDCHR     TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
033254 032722          $RDLIN  ;; CALL=RDLIN     TRAP+11(104411) TTY TYPEIN STRING ROUTINE
033256 033074          $RDOCT  ;; CALL=RDOCT     TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY

```


POWER DOWN AND UP ROUTINES

4615

SBTTL POWER DOWN AND UP ROUTINES

```

*****
:POWER DOWN ROUTINE
033260 012737 033420 000024 ;PWRDN: MOV    @ILLUP,@PWRVEC ;;SET FOR FAST UP
033266 012737 000340 000026      MOV    @340,@PWRVEC+2 ;;PRIO:7
033274 010046      MOV    R0,-(SP)      ;;PUSH R0 ON STACK
033276 010146      MOV    R1,-(SP)      ;;PUSH R1 ON STACK
033300 010246      MOV    R2,-(SP)      ;;PUSH R2 ON STACK
033302 010346      MOV    R3,-(SP)      ;;PUSH R3 ON STACK
033304 010446      MOV    R4,-(SP)      ;;PUSH R4 ON STACK
033306 010546      MOV    R5,-(SP)      ;;PUSH R5 ON STACK
033310 017746      MOV    @SWR,-(SP)    ;;PUSH @SWR ON STACK
033314 010637 033424      MOV    SP,$SAVR6    ;;SAVE SP
033320 012737 033332 000024      MOV    @PWRUP,@PWRVEC ;;SET UP VECTOR
033326 000000      HALT
033330 000776      BR     .-2          ;;HANG UP
*****
:POWER UP ROUTINE
033332 012737 033420 000024 ;PWRUP: MOV    @ILLUP,@PWRVEC ;;SET FOR FAST DOWN
033340 013706 033424      MOV    $SAVR6,SP    ;;GET SP
033344 005037 033424      CLR    $SAVR6      ;;WAIT LOOP FOR THE TTY
033350 005237 033424      1$:   INC    $SAVR6  ;;WAIT FOR THE INC
033354 001375      BNE    1$          ;;OF WORD
033356 012677 145556      MOV    (SP)+,@SWR   ;;POP STACK INTO @SWR
033362 012605      MOV    (SP)+,R5    ;;POP STACK INTO R5
033364 012604      MOV    (SP)+,R4    ;;POP STACK INTO R4
033366 012603      MOV    (SP)+,R3    ;;POP STACK INTO R3
033370 012602      MOV    (SP)+,R2    ;;POP STACK INTO R2
033372 012601      MOV    (SP)+,R1    ;;POP STACK INTO R1
033374 012600      MOV    (SP)+,R0    ;;POP STACK INTO R0
033376 012737 033260 000024      MOV    @PWRDN,@PWRVEC ;;SET UP THE POWER DOWN VECTOR
033404 012737 000340 000026      MOV    @340,@PWRVEC+2 ;;PRIO:7
033412 104401      TYPE   $POWER      ;;REPORT THE POWER FAILURE
033414 033426      $PWRMG: .WORD $POWER ;;POWER FAIL MESSAGE POINTER
033416 000002      RTI
033420 000000      $ILLUP: HALT      ;;THE POWER UP SEQUENCE WAS STARTED
033422 000776      BR     .-2        ;;BEFORE THE POWER DOWN WAS COMPLETE
033424 000000      $SAVR6: 0          ;;PUT THE SP HERE
033426      015      012      120
033431      117      127      105
033434      122      000

.EVEN
.END
4617      000001
    
```

Symbol table

ABASE = 000000	APRTOR- 000000	CURDAT 004042	EM102 020127	EM73 017626
ABORT 030674	APTC SU- 000040	DATA 002304	EM103 020211	EM74 017652
ABORTC 030726	APTENV= 000001	DATA2 002332	EM104 020264	EM75 017703
ABORTE 030772	APTSIZ= 000200	DATVER 026740	EM105 020334	EM76 017734
ABORTZ 031016	APTSPO- 000100	DATVFR 026722	EM106 020407	EM77 020004
ACDW1 = 000000	ASWREG= 000000	DAT1 026752	EM107 020465	ERR 026720
ACDW2 = 000000	ATESTN= 000000	DCOUNT 004024	EM110 020521	ERRCNT 003020
ACPUOP- 000000	AUNIT = 000000	DDISP = 177570	EM111 020563	ERRFP 026716
ACTCHS 002342	AUSWR = 000000	DELAY 026772	EM112 020641	ERROR = 104000
AC0 = *000000	AVECT1= 000000	DETFPA 027726	EM113 020724	ERRVEC = 000004
AC1 = *000001	AVECT2= 000000	DH1 023514	EM114 021010	ERTYPE 025230
AC2 = *000002	BA 002300	DH105 024455	EM123 021031	EXPDAT 004026
AC3 = *000003	BA2 002326	DH115 024507	EM124 021063	FLAG 003026
AC4 = *000004	BCR = 177524	DH134 024564	EM125 021155	FLO 003062
AC5 = *000005	BCSR = 177520	DH24 024000	EM126 021214	FLOAT 003052
AC6 = *000006	BDR = 177524	DH27 024044	EM127 021251	FPVEC = 000244
AC7 = *000007	BIT0 = 000001	DH4 023541	EM130 021313	FSTADD 004044
ADDTRP 026560	BIT00 = 000001	DH41 024107	EM131 021340	GOODAD 003016
ADDW0 = 000000	BIT01 = 000002	DH43 024160	EM132 021370	GTSWR = 104406
ADDW1 = 000000	BIT02 = 000004	DH47 024260	EM135 021426	HITMIS = 177752
ADDW10 = 000000	BIT03 = 000010	DH5 023626	EM136 021464	HT = 000011
ADDW11 = 000000	BIT04 = 000020	DH65 024345	EM137 021526	INIMEM 027174
ADDW12 = 000000	BIT05 = 000040	DH7 023725	EM140 021573	INITMM 025446
ADDW13 = 000000	BIT06 = 000100	DH72 024411	EM141 021664	INQ22 015714
ADDW14 = 000000	BIT07 = 000200	DISPLA 001142	EM142 021747	IOTVEC = 000020
ADDW15 = 000000	BIT08 = 000400	DISPRE 000174	EM143 022040	J11FLT 027734
ADDW2 = 000000	BIT09 = 001000	DMARD 026312	EM144 022113	KDPAR0 = 172360
ADDW3 = 000000	BIT1 = 000002	DMATRN 026230	EM145 022170	KDPAR1 = 172362
ADDW4 = 000000	BIT10 = 002000	DSWR = 177570	EM146 022236	KDPAR2 = 172364
ADDW5 = 000000	BIT11 = 004000	DT1 024702	EM147 022301	KDPAR3 = 172366
ADDW6 = 000000	BIT12 = 010000	DT105 025160	EM15 J 022356	KDPAR4 = 172370
ADDW7 = 000000	BIT13 = 020000	DT115 025170	EM151 022437	KDPAR5 = 172372
ADDW8 = 000000	BIT14 = 040000	DT130 025204	EM152 022505	KDPAR6 = 172374
ADDW9 = 000000	BIT15 = 100000	DT134 025212	EM153 022567	KDPAR7 = 172376
ADEVCT = 000000	BIT2 = 000004	DT14 024750	EM154 022647	KDPDR0 = 172320
ADEVN = 000000	BIT3 = 000010	DT17 024762	EM155 022734	KDPDR1 = 172322
ADLSB 004034	BIT4 = 000020	DT24 024774	EM156 023007	KDPDR2 = 172324
AENV = 000000	BIT5 = 000040	DT27 025004	EM157 023061	KDPDR3 = 172326
AENVN = 000000	BIT6 = 000100	DT35 025016	EM160 023127	KDPDR4 = 172330
AFATAL = 000000	BIT7 = 000200	DT4 024710	EM161 023151	KDPDR5 = 172332
ALLCTR 003152	BIT8 = 000400	DT41 025030	EM162 023211	KDPDR6 = 172334
AMADR1 = 000000	BIT9 = 001000	DT43 025040	EM163 023254	KDPDR7 = 172336
AMADR2 = 000000	BPTVEC = 000014	DT47 025054	EM164 023320	KIPAR0 = 172340
AMADR3 = 000000	BTEXP 003100	DT5 024722	EM165 023377	KIPAR1 = 172342
AMADR4 = 000000	BTRES 003110	DT50 025066	EM166 023446	KIPAR2 = 172344
AMAMS1 = 000000	CCHPAS 003030	DT51 025100	EM2 017205	KIPAR3 = 172346
AMAMS2 = 000000	CCR = 177746	DT52 025112	EM3 017217	KIPAR4 = 172350
AMAMS3 = 000000	CKSWR = 104407	DT64 025124	EM51 017231	KIPAR5 = 172352
AMAMS4 = 000000	COUNT 003120	DT65 025136	EM54 017231	KIPAR6 = 172354
AMSGAD = 000000	CPEREG = 177766	DT7 024736	EM56 017267	KIPAR7 = 172356
AMSGLG = 000000	CR = 000015	DT75 025146	EM57 017313	KIPDR0 = 172300
AMSGTY = 000000	CRLF = 000200	ECPAS 003032	EM61 017345	KIPDR1 = 172302
AMTYP1 = 000000	CSR1 002274	EMTSAV 004054	EM63 017374	KIPDR2 = 172304
AMTYP2 = 000000	CSR12 002322	EMTVEC = 000030	EM64 017425	KIPDR3 = 172306
AMTYP3 = 000000	CSR2 002276	EM1 017151	EM65 017464	KIPDR4 = 172310
AMTYP4 = 000000	CSR22 002324	EM100 020036	EM71 017534	KIPDR5 = 172312
APASS = 000000	CURADD 004050	EM101 020074	EM72 017570	KIPDR6 = 172314

Symbol table

KIPDR7-	172316	PR6	- 000300	SEQ	003072	SW6	- 000100	TPVEC	= 000064
KMCR	177734	PR7	= 000340	SETMMU	027060	SW7	000200	TRAPVE	= 000034
LDPARS	025610	PS	= 177776	SIMGOA	002312	SW8	000400	TRPFLG	020714
LDPORS	025640	PSW	= 177776	SIPARO	= 172240	SW9	- 001000	TRTVEC	= 000014
LEDCNT	002314	PWDSEQ	004032	SIPAP1-	172242	TAB1	003234	TSTADD	003022
LF	= 000012	PWRVEC	= 000024	SIPAR2	172244	TAB10	003564	TSTLOC	003162
LKS	= 177546	Q22EN	003002	SIPAR3	= 172246	TAB11	003374	TST1	005060
LKSFL	002340	Q22INT	026210	SIPAR4	= 172250	TAB11A	003404	TST10	010042
LKSINT	025722	Q22SIZ	025730	SIPAR5-	172252	TAB12	003414	TST11	011036
LOOP	005060	RAMPAR	025664	SIPAR6	= 172254	TAB13	003424	TST12	011230
LOOPIN	003154	RBUF	= 177562	SIPAR7	= 172256	TAB13B	003434	TST13	011534
LOWADD	003014	RBUF1	= 176502	SIPDR0-	172200	TAB14	003444	TST14	011722
LSTADD	004046	RCOUNT	002316	SIPDR1	= 172202	TAB15	003454	TST15	012052
MAIREG	= 177750	RCSR	= 177560	SIPDR2-	172204	TAB16	003464	TST16	012244
MASK	003160	RCSR1	= 176500	SIPDR3-	172206	TAB17	003474	TST17	012436
MER	= 172100	RDCHR	- 104410	SIPDR4-	172210	TAB18	003504	TST2	005332
MERTAG	003050	RDLIN	= 104411	SIPDR5	= 172212	TAB2	003244	TST20	012772
MMRC	- 177572	RDOCT	= 104412	SIPDR6	172214	TAB21	003514	TST21	013144
MMR1	= 177574	RECDAT	004030	SIPDR7	= 172216	TAB22	003524	TST22	013724
MMR2	177576	RECDST	003142	SLEND	015262	TAB23	003534	TST23	014452
MMR3	= 172516	RECFEC	003122	SLOC00	003004	TAB24	003544	TST24	014766
MMJ	026412	RECST	003132	SLOC01	003006	TAB25	003554	TST25	015266
MMUTRP	026564	RESVEC	= 000010	SPS	003074	TAB26	003564	TST26	015502
MMVEC	= 000250	RITEDA	004036	SPSJ	003076	TAB27	003574	TST27	015772
MSER	= 177744	RRMEM	027532	SR0	= 177572	TAB28	003604	TST3	005520
NATREG	= 177520	RWTMEM	027360	SR1	= 177574	TAB29	003614	TST30	016244
NEWADD	003024	R6	= 000006	SR2	= 177576	TAB29A	003624	TST4	005742
NEWDAT	004040	R7	= 000007	SR3	= 172516	TAB3	003254	TST5	006156
NOABRT	031032	SAVBR	002346	STACK	= 001100	TAB30	003634	TST6	006776
NOSLU	015264	SAVMRO	003040	START	004054	TAB31	003644	TST7	007522
NOTOK	026642	SAVMR1	003042	STBOT	= 001000	TAB32	003654	TYPDS	- 104405
NULL	= 000000	SAVMR2	003044	STKLMT-	177774	TAB33	003664	TYPE	= 104401
NXTST	011036	SAVPCR	002344	SWR	001140	TAB34	003674	TYPQC	= 104402
OK	026620	SAVPOS	003156	SWREG	000176	TAB4	003264	TYPON	= 104404
OK1	026646	SAVSUP	003034	SWTSEL	016302	TAB40	003704	TYPOS	= 104403
ONOQ22	026154	SAVSWR	003046	SW0	= 000001	TAB41	003714	UDPAR0	= 177660
PAR	026534	SAVTIM	003012	SW00	= 000001	TAB42	003724	UDPAR1	= 177662
PARABT	004052	SAVUSE	003036	SW01	= 000002	TAB43	003734	UDPAR2	= 177664
PAREND	006776	SAV30	004142	SW02	= 000004	TAB45	003744	UDPAR3	= 177666
PARINT	006770	SAV32	004144	SW03	= 000010	TAB46	003754	UDPAR4	= 177670
PARPAT	003010	SCOPE	= 000004	SW04	= 000020	TAB47	003764	UDPAR5	= 177672
PAR1	026536	SDPAR0	= 172260	SW05	= 000040	TAB47A	003774	UDPAR6	= 177674
PATT	005724	SDPAR1	= 172262	SW06	= 000100	TAB48	004004	UDPAR7	= 177676
PCR	= 177522	SDPAR2	= 172264	SW07	= 000200	TAB49	004014	UDPDR0	= 177620
PDR	026512	SDPAR3	= 172266	SW08	= 000400	TAB5	003274	UDPDR1	= 177622
PDR1	026514	SDPAR4	= 172270	SW09	= 001000	TAB5A	003304	UDPDR2	= 177624
PIP	= 177772	SDPAR5	= 172272	SW1	= 000002	TAB6	003314	UDPDR3	= 177626
PIPG	= 177772	SDPAR6	= 172274	SW10	= 002000	TAB6A	003324	UDPDR4	= 177630
PIRQT	016242	SDPAR7	= 172276	SW11	= 004000	TAB7	003334	UDPDR5	= 177632
PIRQVE	= 000240	SDPDR0	= 172220	SW12	= 010000	TAB8	003344	UDPDR6	= 177634
POLY	= 120001	SDPDR1	= 172222	SW13	= 020000	TAB9	003354	UDPDR7	= 177636
PPG	= 000000	SDPDR2	= 172224	SW14	= 040000	TBITVE	= 000014	UFDLFG	004146
PR1	= 000040	SDPDR3	= 172226	SW15	= 100000	TCOUNT	002320	UFDSET	= 000001
PR2	= 000100	SDPDR4	= 172230	SW2	= 000004	TEMP	002740	UIPAR0	= 177640
PR3	= 000140	SDPDR5	= 172232	SW3	= 000010	TIMOUT	003000	UIPAR1	= 177642
PR4	= 000200	SDPDR6	= 172234	SW4	= 000020	TKVEC	= 000060	UIPAR2	= 177644
PR5	= 000240	SDPDR7	172236	SW5	= 000040	TOUT	026402	UIPAR3	= 177646

Symbol table

JIPAR4	177650	\$CHARC	031632	\$EOPCT	030054	\$MAMS4	001244	\$SWREG	001222
JIPAR5	177652	\$CKSWR	032310	\$ERFLG	001103	\$MBADR	000234	\$SWRMK	= 000300
UIPAR6	177654	\$CMTAG	001100	\$ERMAX	001115	\$MFLG	031276	\$TESTN	001204
UIPAR7	177656	\$CM3	= 000000	\$ERROR	030452	\$MNEW	033063	\$TIMES	001164
JIPDR0	177600	\$CM4	= 000002	\$ERRPC	001116	\$MSGAD	001214	\$TKB	001146
JIPDR1	177602	\$CNTLG	033045	\$ERRTB	001324	\$MSGLG	001216	\$TKS	001144
UIPDR2	177604	\$CNTLU	033040	\$ERTTL	001112	\$MSGTY	001200	\$TMP0	001160
UIPDR3	177606	\$CPUOP	001226	\$ESCAP	001166	\$MSWR	033052	\$TMP1	001162
UIPDR4	177610	\$CRLF	001175	\$ETABL	001220	\$MTYP1	001231	\$TN	= 000031
UIPDR5	177612	\$DBLK	032300	\$ETEND	001324	\$MTYP2	001235	\$TPB	001152
UIPDR6	177614	\$DDW0	001254	\$FATAL	001202	\$MTYP3	001241	\$TPFLG	001157
JIPDR7	177616	\$DDW1	001266	\$FFLG	031300	\$MTYP4	001245	\$TPS	001150
UQUIET	004150	\$DDW10	001310	\$FILLC	001156	\$MXCNT	030450	\$TRAP	033176
VIREOP	016256	\$DDW11	001312	\$FILLS	001155	\$NULL	001154	\$TRAP2	033220
VMKOR	004152	\$DDW12	001314	\$GDADR	001120	\$NWTST	= 000000	\$TRP	= 000013
VQBE1	002306	\$DDW13	001316	\$GDDAT	001124	\$OCNT	032060	\$TRPAD	033232
VQBE2	002334	\$DDW14	001320	\$GET42	030104	\$OMODE	032062	\$TSTM	000236
VQPR1	002310	\$DDW15	001322	\$GTSWR	032360	\$OVER	030426	\$TSTMN	001102
VQPR2	002336	\$DDW2	001270	\$HD	= 000001	\$PASS	001206	\$TTYIN	033030
WC	002302	\$DDW3	001272	\$HIBTS	000232	\$PASTM	000240	\$TYPDS	032064
WC2	002330	\$DDW4	001274	\$HIOCT	033174	\$POWER	033426	\$TYPE	031302
WLDTRP	026704	\$DDW5	001276	\$ICNT	001104	\$PWRDN	033260	\$TYPEC	031514
YBUF	177566	\$DDW6	001300	\$ILLUP	033420	\$PWRMG	033414	\$TYPEX	031634
XBUF1	176506	\$DDW7	001302	\$INTAG	001135	\$PWRUP	033332	\$TYPC	031662
XCSR	177564	\$DDW8	001304	\$ITEMB	001114	\$QUES	001174	\$TYPON	031676
XCSR1	176504	\$DDW9	001306	\$LF	001176	\$RDCHR	032572	\$TYPOS	031636
\$APTHD	000232	\$DEVCT	001210	\$LFLG	031277	\$RDLIN	032722	\$UNIT	001212
\$ATYC	031060	\$DEVM	001256	\$LPADR	001106	\$RDOCT	033074	\$UNITM	000242
\$ATY1	031034	\$DOAGN	030124	\$LPERR	001110	\$RDSZ	= 000010	\$USWR	001224
\$ATY3	031042	\$DTBL	032270	\$MADR1	001232	\$RTNAD	030126	\$VECT1	001250
\$ATY4	031052	\$ENDAD	030114	\$MADR2	001236	\$SAVR6	033424	\$VECT2	001252
\$AUTOB	001134	\$ENCCT	030062	\$MADR3	001242	\$SCOPE	030150	\$XOFF	= 000023
\$BASE	001254	\$ENDMG	030133	\$MADR4	001246	\$SETUP	= 000137	\$XON	= 000021
\$BDADR	001122	\$ENULL	030130	\$MAIL	001200	\$STUP	= 177777	\$XTSTR	030170
\$BDDAT	001126	\$ENV	001220	\$MAMS1	001230	\$SVLAD	030372	\$\$GET4	= 000000
\$BELL	001170	\$ENVM	001221	\$MAMS2	001234	\$SVPC	= 000232	\$OFILL	032061
\$CDW1	001260	\$EOP	030014	\$MAMS3	001240	\$SWR	= 167400	.\$X	= 000232
\$CDW2	001262								

. ABS. 033436 000 (RW,I,GBL,ABS,OVR)
 000000 001 (RW,I,LCL,REL,CON)

Errors detected: 0

*** Assembler statistics

Work file reads: 403
 Work file writes: 308
 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: 00:04:18.08
 COKDDAO,COKDDAO/-SP=ORION.MLB/ML,COKDDAO.MAC/DS:GBL