```
                                                          .REM &
                                        IDENTIFICATION
                                              --

                                PRODUCT CODE:  AC FG67B-MC

                                PRODUCT NAME:  COKDDBO  KDJ11 DA CLUSTER DIAG.

                                PRODUCT DATE:   APRIL, 1986

                                MAINTAINER:    DIAGNOSTIC ENGINEERING
```

Table of contents

TABLE OF CONTENTS

1.0 PROGRAM ABSTRACT

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

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 conta'n the self-test and the boot codes.

The Serial Line Unit is implemented thru 2 Dlart 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

2.0 PROGRAMMING CONVENTIONS

2.1 Implementation Language

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

2.2 Program Generation

This Diagnostic is developed and assembled on a VAX us ng the PDP/11 mode "MCR MAC" and the "ORION.MLB' l'braryas the following:

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

Finally NAME.OBJ would be transfered to an XXDP+ med a 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 arb'trary place after HALT on Break the following registers should be set up:
17777572=0 to disable memmory 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 Opt'ons

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

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 it, it takes about 2 minutes.

2.8 Error Information

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

2.9 Examples

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

```
* KDJ11-DA CPU DIAGNOSTIC - COKDDA0 *

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 carrige return or
2) just type in carriage return in which case the software register will remain unchanged.

Example of error printout:

```
        ERROR DOING Q22BE INTERRUPTS
```

```
                                TEST   ERROR   ADDRESS ADDRESS
                                 #      PC     <21-16> <15-0>
                                 27    105620  66600   000000
                             Note: this may not correspond to the actual Program Counter.
```

```
246
247                             3.0     PROGRAM DESCRIPTION
248                                     - - - -
249
250                                     3.1  J11 CODE
251
252                                     This portion of the code tests out the J11 chip set. It is broken
253                                     into 3 pieces: CPU tests, which verify different instructions in
254                                     different modes and different trap conditions; MMU tests, which
255                                     verify different functions of MMU; and FFP tests, which do
256                                     different floating point instructions.
257
258                                     This portion of the code have been written in close relationship
259                                     with the J11 microcode. Therefore, even though not all possible
260                                     instructions in all possible addressing mode have been tested, an
261                                     attempt has been made to exercise all of the microcode.
262
263                                     Most of the CPU diagnostic tests have been taken from the KDJ11 B
264                                     Cluster diagnostic and has not been rewritten due to similarities.
265
266
267
268
269
270                                     3.2  NATIVE REGISTER
271
272
273
274                             TEST - NATIVE REGISTER
275
276                             NATIVE REGISTER TEST
277
278                              This test checks out the native register's existance and it's
279                              various bits.
280
281                              BGNTST
282                              ) Set up timeout vector PC to TIMOUT
283                              ) Set up timeout vector PSW to 7
284                              ) Read the Native register
285                              ) Check out the bootstrap switch as read only
286                              ) Check out the module functional revision as read only
287                              ) Check out that the self-test enable bit works
288                              ) Check out he indicators (i.e LEDS)
289                              ENDTST
290
291                              TIMOUT: Clean stack
292                                      Error NATIVE register timed out
293
294
295
```

```
297
298                                     3.3  MAINTENANCE REGISTER
299
300
301
302                                 TEST    MAINTENANCE REGISTER TEST
303
304                                 MAINTENANCE REGISTER TEST
305                                 THIS TEST WILL ADDRESS MAINTENANCE REGISTER AND CHECK BITS
306                                 7 4 TO BE 0010, 2 1 TO BE 10, AND READ BITS 10 08, 03, 00
307                                 FOR FUTURE USE.  THOSE BITS REPRESENT THE FOLLOWING SIGNALS:
308                                 MULTIPROCESSOR SLAVE, UNIBUS SYSTEM, FPA AVAILABLE, HALT/TRAP
309                                 OPTION, AND AC POWER OKAY.
310                                 ROUTINE TEST
311                                 .       IF MAINT. REG. BITS <7 4> NE 0010 OR <2 1> NE 10 THEN
312                                 .           .       ERROR
313                                 .       ENDIF
314                                 .       READ MAINT.REG. BITS <10 08,03,00>
315                                 ENDROUTINE
316
317
318
```

```
320
321                          3.4  KDJ11 DA ON BOARD MEMORY
322
323                          The KDJ11 DA has on board memory of 512 Kbytes   block mode, w th
324                          parity, f xed start at 0
325
326                          The KDJ11 DA board uses 256K x 1 chips which allows us to use two
327                          patterns per RAM only.
328
329                          Our memory diagnostic starts w th testing the data path using this
330                          pattern: 0, 177777, 177400, 170360, 007417, 052525, 125252. The 2nd
331                          test is check ng the accessib lity of each address and an error will
332                          flag if memory traps. The th rd test is to check the memory registers
333                          specific to the KDJ11-DA. Fourth, s the short and stuck on bits test
334                          which will test every s ngle bit 0/1 for all addresses. The KNAIZUK
335                          HARTMANN algorithm wh ch d vides the memory into sections of three is
336                          used in test f ve as a qu ck verify of all stuck at faults in the data
337                          and addressing. Test s x checks the parity detect logic of all memory
338                          locations on the KDJ11 DA.
339                          You can refer to this section for more details.
340
341
342
343                         TEST   KDJ11 DA DATA PATHS
344
345                         DATA PATH TEST
346
347                         This test checks out the data and address paths on the KDJ11-D
348                         Board. The patterns used will be:
349                                                          0
350                                                          177777
351                                                          177400
352                                                          170360
353                                                          007417
354                                                          052525
355                                                          125252
356
357                         BGNTST
358                         ) Set Timeout trap to TIMOUT
359                         ) Set timeout priority to 7
360                         ) Read location 0
361                         ) FOR pattern := first to last
362                         )   Write pattern
363                         )   Read pattern
364                         )   IF Pattern read <> Pattern Written THEN
365                         )    ERROR
366                         ) ENDFOR
367                         ENDTST
368
369
370
371                        TEST - MEMORY ACCESSABILITY
372
373                        ACCESSIBILITY TEST
374
375                         This test will check the accessibility of each address of memory
376                         on the KDJ11-D Board. IF a memory address traps out then an error
```

```
377                             will be flagged. A side effect of this test should be that all memory
378                             is cleared
379
380                             BGNTST
381                             )  Set timeout trap to TIMOUT
382                             )  Set timeout priority to 7
383                             )  For MSB_Address := 200000 to 1777777 DO
384                             )      Contents of address := 0
385                             )  Go to the next test
386                             )
387                             )TIMOUT:
388                             )   Error Address should not have timed out
389                             ENDTST
390
391
392
393                           TEST   MEMORY ERROR REGISTER
394
395                           MEMORY ERROR REGISTER
396
397                             This test will check the MEMORY ERROR REGISTER on the KDJ11-D
398                             Board.
399
400                             BGNTST
401                             )  Setup timeout VECTOR PC to TIMOUT
402                             )  Setup timeout VECTOR Priority to 7
403                             )  Setup Parity abort VECTOR to PARINT
404                             )  Setup Parity abort VECTOR Priority to 7
405                             )  Do a bus reset
406                             )  Read the Memory Error Register (17772100)
407                             )  Make sure that the register bits are in the right state
408                             )  Check all R/W bits
409                             ENDTST
410
411
412
413                           TEST - DATA SHORTS AND STUCK AT BITS
414
415                           DATA SHORTS AND STUCK AT BITS TEST
416
417                             This test will check the DATA Rams for Data shorts and stuck at bits.
418                             Testing occurs as below:
419
420                                    1. A memory location is checked to be set to 0
421                                    2. IF NOT 0 error
422                                    3. The location is complemented
423                                    4. IF contents NOT -1 error
424                                    5. This is repeated for all addresses
425                                    6. Steps 1-5 are done from location 1777777 to 0
426
427                             Note: The KDJ11-DA board uses 256k X 1 chips,This allows us to
428                                    use only 2 patterns per RAM. IF one bits is shorted to
429                                    another we will detect this when we read for the initial state.
430                                    If it has changed from the expected state then chances are that
431                                    the bits are shorted together.
432
433                             A side effect of this test should be that memory is cleared.
```

```
434
435                             BGNTST
436                             )  FOR MSB Address :- 0 to 1777776 DO BY 2
437                             )       Clear Address
438                             )       If Contents <> 0 THEN
439                             )          ERROR in memory
440                             )       Complement the Contents of Address
441                             )       IF contents <>  1 THEN
442                             )          ERROR in memory
443                             )  ENDFOR
444                             )  FOR MSB Address := 1777776 DOWNTO 0 DO BY 2
445                             )       IF contents <>  1 THEN
446                             )          ERROR in memory
447                             )       Complement the Contents of Address
448                             )       IF contents <> 0 THEN
449                             )          ERROR in memory
450                             )  ENDFOR
451                             ENDTST
452
453
454
455                             TEST   QUICK VERIFY DATA AND ADRESSING TEST
456
457                             UNIQUE ADDRESSING TEST
458
459                              This test will check the data and addressing of the memories.
460                             It uses the KNAIZUK HARTMANN QUICK VERIFY TRAM TEST ALGORITHM.
461                             This algorithm will test memory for all stuck at faults in the
462                             data and addressing
463
464                              Memory is split up into sections of 3.
465                             I.E. 0,1,2 - 3,4,5 - 6,7,10 ...
466
467                              Testing works as follows.
468
469                              1. Write a 0 into the 2nd and 3rd address of all groups
470                                 write a -1 into the 1st address of all groups
471                              2. make sure that the 2nd address of all groups contain 0
472                              3. Write a -1 into the 2nd address of all groups
473                              4. Make sure that the 3rd address of all groups contain 0
474                              5. Make sure that the 1st and 2nd address of all groups
475                                 contain -1
476                              6. Write a 0 into the 1st address of all groups
477                              7. Make sure that the 1st address of all groups contain 0
478                              8. Write a  1 into the 3rd address of all groups
479                              9. Make sure that the 3rd address of all groups contain -1
480
481
482
483
484                             TEST - CHECK PARITY DETECT LOGIC AND RAMS
485
486                             CHECK PARITY DETECT LOGIC
487
488                              This test will check out the parity detection logic on the KDJ11 DA
489                             board. We will write wrong parity to all locations and then we will
490                             Expect the a parity trap on a read.
```

```
491
492                             BGNTST
493                             )   Set up Parity Vector (114) To PARINT
494                             )   Enable Write Bad parity (Set bit 2 in MER (17772100))
495                             )   Clear all of memory
496                             )   Read first location written to
497                             )   IF Parity Abort Occurs THEN
498                             )      Error There should be no Parity Aborts when disabled
499                             )   Enable Parity Error  (Set Bit 0 in MER (17772100)
500                             )   FOR First Address to Last address DO
501                             )   BEGIN
502                             )   )    READ Address
503                             )   )    NOP
504                             )   )    IF No Interrupt THEN
505                             )   )       Error Didn't Detect Bad parity
506                             )   )    ELSE
507                             )   )       Clear the interrupt flag
508                             )   )     Read the MER and make sure the obtained address is the correct one
509                             )   ENDFOR
510                             )
511                             )  PARINT:
512                             )     Flag that an interrupt occured
513                             )     RTI
514                             ENDTST
515
516
517
518
519
520
521
522
523                                     3.5  ON-BOARD ROM CODE
524
525                                     The KDJ11-DA Native Firmware resides in a 2-16Kx8 EPROM's which
526                                     are phisically located on the KDJ11-DA module. The Firmware has
527                                     a monitor to accept user commands, auto sequence boot, console
528                                     boot, power up self tests, extended self tests invoked via user
529                                     intervention, and support a text of basic error messages 'n all
530                                     supported eleven foreign languages including extended error
531                                     messages in English.
532
533                                     In this test we perform a sum test for the on Board ROM's which
534                                     resides at address 17400000 - 17677777.
535
536
537
538                             TEST  - 16 BIT ROM CHECKSUM TEST
539
540                             ROM'S CHECKSUMS
541
542                             16 BIT ROM TEST
543
544                             BGNTST
545                             )   INIT MMU REGISTERS
546                             )   POINT PAGE 2 TO SELF TEST
547                             )   ENABLE SELF TEST BIT
```

```
548                             )   READ FIRST 2 WORDS OF ROM
549                             )   GET EXPECTED CHECKSUM
550                             )
551                             )   DO CALCULATE CHECKSUM
552                             )   )   ADD WORD TO REGISTER
553                             )   )   DO UNTIL ALL WORDS ADDED
554                             )   END CALCULATION
555                             )
556                             )   NEGATE THE SUM OF WORDS
557                             )   IF NEG. SUM < > EXPECTED CHECK_UM
558                             )   ERROR
559                            ENDTST
560
561
562
563
564
565
566
567                                    3.6  LINE TIME CLOCKS CODE
568
569                                    The line time clock control and status register ias accessed at
570                                    address 17777546. The EVENT interrupt thru vector 100 on interrupt
571                                    request level 6 is received  from the Q-bus signal BEVENT.
572
573                                    Note: in UFD mode only functions corresponding to Boot Rom selection
574                                    are checked.
575
576                                    In this diagnostic we start with test01: existence of the clock,
577                                    test02: interrupt priority max 5, test03: resetting LKS.
578
579                            TEST - LTC BIT 7
580
581                            LTC BIT 7 TEST
582
583                             This test check for the existance of the LTC register and it
584                             makes sure that the clock is ticking.
585
586                             BGNTST
587                              Set up timeout vector PC to TIMOUT
588                              Set up timeout vector PSW to 7
589                              Read the LTC CSR
590                              IF not timed OUT THEN
591                              : FOR a set amount of time
592                              :      Check bit7
593                              :      IF BIT7 is set THEN
594                              :         Increment BIT7 set flag
595                              :      ELSE
596                              :         INCREMENT BIT7 Clear Flag
597                              : ENDFOR
598                              : IF either flag has not been set at all THEN
599                              :    Error Clock is not ticking
600                             ENDTST
601
602                             TIMOUT: Clean stack
603                                      Error LTC register timed out
604
```

```
605
606
607                             TEST    LKS INT RRUPT PRIORITY
608
609                             CHECK THAT LKS INTERRUPTS HAPPEN AT PRIORITY 5 CLEARING LKS<07>
610                             AND DON'T HAPPEN AT PRIORITY 6.
611                             ROUTINE TEST
612                             IF UFD AND LKS IS DISABLED THEN
613                             .       EXIT TEST
614                             ENDIF
615                             .       SET PRIORITY TO 5
616                             .       CLEAR INTERRUPT_FLAG
617                             .       LET LKS<06>=#1 (ENABLE INTERRUPTS)
618                             .       SET COUNTER TO WAIT FOR 3 INTERRUPTS
619                             .       REPEAT
620                             .       .       DECREMENT COUNTER
621                             .       UNTIL INTERRUPT_FLAG EQ #3 OR COUNTER EQ #0
622                             .       CLEAR LKS<06>
623                             .       IF LKS<07> EQ #1 THEN
624                             .       .       ERROR (WAS NOT CLEARED ON INTERRUPT)
625                             .       ENDIF
626                             .       IF COUNTER LT TIME_REQUIRED_FOR_3_INTERRUPTS_FOR 800HZ
627                             .       .       ERROR (INTERRUPTS NEVER GO LOW)
628                             .       ENDIF
629                             .       IF INTERRUPT_FLAG LT #3 THEN
630                             .       .       ERROR (INTERRUPTS DON'T HAPPEN)
631                             .       ENDIF
632                             .       CLEAR INTERUPT_FLAG
633                             .       WAIT FOR LKS<7>=1
634                             .       LET LKS<7>=0
635                             .       IF LKS<7> NE #0 THEN
636                             .       .       ERROR (LKS<7> NOT CLEARED
637                             .       ENDIF
638                             .       SET PRIORITY TO 6
639                             .       SET COUNTER TO 1 SLOW CLOCK INTERRUPT
640                             .       SET LKS<06>
641                             .       REPEAT
642                             .       .       DECREMENT COUNTER
643                             .       UNTIL COUNTER EQ #0 OR INTERRUPT_FLAG NE #0
644                             .       IF INTERRUPT_FLAG NE #0 THEN
645                             .       .       ERROR (INTERRUPT WAS AT WRONG PRIORITY)
646                             .       ENDIF
647                             .       RESTORE ORIGINAL PRIORITY
648                             ENDROUTINE
649
650                             ROUTINE LINE_CLOCK_INTERUPT
651                             .       INCREMENT INTERRUPT_FLAG
652                             ENDROUTINE
653
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                             .       EXIT TEST
663                             ENDIF
664                             .       POINT LKS VECTOR 100 TO ERROR_LKS_ILLEGAL INTERRUPT
665                             .       SYNCHRONIZE LKS BY WAITING FOR 3 PULSES
666                             .       LET LKS<06>=#1
667                             .       EXECUTE "RESET"
668                             .       IF LKS<6> NE #0 THEN
669                             .               ERROR
670                             .       ENDIF
671                             .       IF ILLEGAL_LINE CLOCK_INTERRUPT NE 0 THEN
672                             .               ERROR
673                             .       ENDIF
674                             ENDROUTINE
675
676                             ROUTINE ERROR_LKS_ILLEGAL_INTERRUPT
677                             .       FLAG ILLEGAL_LINE_CLOCK_INTERRUPT
678                             RETURN
679
680
681
682
683
684
685
686
687                                     3.7  SERIAL LINE UNIT CODE
688
689                                     The KDJ11-da board has two Serial Line Unit that have to be tested.
690                                     When the proceesor halts it enters uODT and communicates over the
691                                     SLU0 at addresses 17777560 thru 17777566.They considered the console
692                                     addresses by the CPU. SLU1 is the General  purpose Serial I/O and
693                                     is addressed at 17776500 thru 17776506 and SLU1 does not have the halt
694                                     on breake option.
695
696                                     In this diagnostic we verify the functionality of the SLU chip
697                                     utilizing the maintenance mode of the chip. All serial line unit
698                                     tests: interrupt level, loop back capability, and transmitter
699                                     receiver bits registers, are performed on SLU0 and SLU1 except the
700                                     BREAK CONDITION and the OVERRUN CONDITION tested on SLU1 only, because
701                                     SLU0 is used as the console .
702
703
704
705                             TEST  - SERIAL LINE UNIT REGISTERS
706
707                             SERIAL LINE UNIT TEST(*)
708                             BCR<2-0> WILL BE READ TO FIND OUT BAUD RATE. SLU WILL BE PROG
709                             RAMMED TO CHECK THE INTERRUPT LEVELS BY SETTING BIT<06> IN
710                             RCSR AND XMIT. LOOP BACK CAPABILITIES WILL BE TESTED BY SETTING
711                             TO 1 XCSR<02>.   THE LINE CLOCK INTERRUPT SUBROUTINE WILL BE
712                             USED TO RETURN TO THE EXECUTION OF THE DIAGNOSTICS, IF THE
713                             PROGRAM HANGS IN THE LOOP BACK MODE.
714                             ROUTINE TEST
715                             IF UFD AND CONSOLE NOT PRESENT
716                             .       GO TO TEST_22
717                             ENDIF
718                             .       IF BCR<07> EQ #0 THEN
```

```
                        .       .       READ BCR<2 0> TO GET BAUD RATE
                        .       ENDIF
                        .       LET 4=ADDRESS_OF TIMEOUT_ROUTINE
                        .       DO FOR RCSR,XCSR,RBUF,XBUF
                        .       .       READ XRCSR,XCSR,RBUF,XBUF
                        .       .       IF TIMEOUT_FLAG NE #0 THEN
                        .       .       .       ERROR
                        .       .       ENDIF
                        .       ENDDO
                        ENDROUTINE

                        ROUTINE TIMEOUT
                        .       LET TIMEOUT FLAG=#1
                        ENDROUTINE


                        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


                        TEST  - RCSR BIT 7 AND XCSR BIT 2

                        CHECK THAT RCSR<07> CAN BE 0 AND 1 AND THAT XCSR<02> WORKS PROPERLY.

                        RCSR    <07>    RECEIVER DONE
                        RCSR    <02>    MAINTENANCE

                        ROUTINE TEST
                        .(CHECK RCSR<07> AND XCSR<07>)
                        .       WAIT FOR XCSR<07>=#1
                        .       LET XCSR<02>=#1 (LOOP BACK MODE)
                        .       LET XBUF=#125
                        .       WAIT FOR RCSR<07>=#1 NO MORE THAN 200MSEC
                        .       IF RCSR<07> NE #1 THEN
                        .       .       ERROR (RCSR<07> DOES NOT BECOME 1 OR XCSR<02>DOES NOT
                        .       .               WORK)
                        .       ENDIF
                        .       IF RBUF NE #125 THEN
                        .       .       ERROR
```

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

```
                        CHECK THAT INTERRUPTS HAPPEN AT PRIORITY 3 AND THAT THEY CLEAR
                        RCSR<06> AND XCSR<06>.

                        ROUTINE TEST
                        .       LET 60=#ADDRESS OF LEGAL RINTERRUPT
                        .       LET 64=#ADDRESS OF LEGAL XINTERRUPT
                        .       LET XCSR<02>=#1
                        .       SET PRIORITY TO #3
                        .       WAIT FOR XINTERRUPT=#3
                        .       IF XCSR<07> EQ #1 THEN
                        .               ERROR
                        .       ENDIF
                        .       WAIT FOR RINTERRUPT=#3
                        .       IF RCSR<07> EQ #0 THEN
                        .               ERROR
                        .       ENDIF
                        .       LET XCSR<02>=#0
                        .       SET PRIORITY TO NORMAL
                        ENDROUTINE

                        ROUTINE LEGAL_XINTERRUPT
                        .       LET XBUF=#CHARACTER
                        .       INCREMENT XINTERRUPT
                        ENDROUTINE

                        ROUTINE LEGAL_RINTERRUPT
                        .       READ RCSR
                        .       INCREMENT RINTERRUPT
                        ENDROUTINE



                        TEST    BREAK CONDITION

                        SLU #1 IS TESTED SINCE SLU #0 IS BEING USED AS THE CONSOLE

                        CHECK THAT SENDING BREAK CAUSES FRAMING ERROR.

                        RCSR    <15>    ERROR
                                <13>    FRAMING ERROR
                                <11>    RECEIVED BREAK

                        XCSR    <00>    TRANSMIT BREAK

                        ROUTINE TEST
                        .       LET XCSR<02>=#1
                        .       LET XCSR<00>=#1
                        .       WAIT FOR RCSR<07>=#1
                        .       IF RBUF<15!13!11> NE #1 THEN
                        .               ERROR (ERROR, FRAMING ERROR, RECEIVE BREAK NE 1)
                        .       ENDIF
                        .       LET XCSR<00>=#0
                        .       IF XCSR<00> NE #0 THEN
                        .               ERROR (XCSR<00> DOES NOT GO LOW)
                        .       ENDIF
                        .       WAIT FOR XCSR<07>=#1
                        .       LET XBUF=#NULL (SEND NULL CHARACTER TO SEE ERROR CLEARED)
```

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

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

```
                                3.9  Q22BE CODE

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

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



                        TEST  - DIFFERENT LEVELS OF INTERRUPTS

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

                        CHECK DIFFERENT LEVELS OF INTERRUPTS.
                        ROUTINE TEST
                        .       SET VECTOR TO INTERRUPT_DMA
                        .       FOR INTERRUPTS FROM 4 TO 7 DO
                        .       .       ENABLE INTERRUPTS
                        .       .       SET PRIORITY=INTERUPT
                        .       .       IF INTERRUPT_FLAG SET THEN
                        .       .       .       ERROR
                        .       .       ENDIF
                        .       .       ENABLE INTERRUPTS
                        .       .       SET PRIORITY=INTERRUPT-1
                        .       .       IF INTERRUPT_FLAG NOTSET THEN
                        .       .       .       ERROR
                        .       .       ENDIF
                        .       .       LET INTERRUPT_DMA=0
                        .       ENDDO
                        ENDROUTINE

                        ROUTINE INTERUPT_DMA
                        .       LET INTERRUPT_FLAG=1
                        RETURN
                        ENDROUTINE



                        TEST  - ARBITRATION BETWEEN PIRQ'S AND INTERRUPTS

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

```
1027                            .       ENDDO
1028                        ENDROUTINE
1029
1030
1031
1032
1033
1034
1035
1036                                3.10  LIST OF SUBTESTS PERFORMED
1037
1038                                The following list represents the sequential order of subtests
1039                                performed in COKDDAO..... subtests which are subject to the APT
1040                                qualifications of section 3.2 are indicated by a Cache APT label.
1041
1042
1043                        TEST NO.        TEST NAME/FUNCTION
1044
1045                        test 1          Base instruction set tests
1046                        test 32         KDJ11 DA on Board Memory
1047                        test 33         KDJ11-DA Data Path
1048                        test 34         Memory accessability
1049                        test 35         Memory Error Register
1050                        test 36         Data shport and stuck at bits
1051                        test 37         Quick verify Data and Addressing
1052                        test 38         Check Parity detect Logic and RAM's
1053                        test 39         Native Register
1054                        test 40         On Board ROM code
1055                        test 41         LTC bit 7
1056                        test 42         LKS interrupt Priority
1057                        test 43         Resetting LKS
1058                        test 44         Maintenance Register
1059                        test 45         Serial LIne Unit Register
1060                        test 46         XCSR bit 7
1061                        test 47         RCSR bit 7 and XCSR bit 2
1062                        test 48         Reset and XCSR<2!0>
1063                        test 49         Reset and interrupt enable bit
1064                        test 50         Interrupt priority for SLU
1065                        test 51         Break Condition
1066                        test 52         Overrun Condition
1067                        test 53         LED's On
1068                        test 54         Different Levels of interrupt
1069                        test 55         Arbitration between PIRQ's and interrupts
1070
1071
1072
```

3.11 PROGRAM UPDATES AND MODIFICATIONS

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

4.0 BIBLIOGRAPHY

N/A

5.0 GLOSSARY

N/A

APPENDICES

N/A

&

M2


```
1115    167400                  $SWR=167400
1116    000300                  $SWRMK=300
1117                    .TITLE  COKDDB0 KDJ11 DA CLUSTER DIAG.
                        ;*COPYRIGHT (C) APR 86
                        ;*DIGITAL EQUIPMENT CORP.
                        ;*MAYNARD, MASS. 01754
                        ;*
                        ;*PROGRAM BY DIAG. ENG.
                        ;*
                        ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP 11 MAINDEC SYSMAC
                        ;*PACKAGE (MAINDEC-11 DZQAC-C8), OCT, 1982.
                        ;*
        000001          $TN=1
1118                    .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>
1120                    ;*          7             DO EXTENSIVE DATA RAM TEST
1121                    ;*          6             DO EXTENSIVE TAG RAM TEST
1122                    .SBTTL  MEMORY MANAGEMENT DEFINITIONS
                        ;*KT11 VECTOR ADDRESS
        000250          MMVEC=  250
                        ;*KT11 STATUS REGISTER ADDRESSES
        177572          SR0     177572
        177574          SR1     177574
        177576          SR2=    177576
        172516          SR3=    172516
                        ;*USER "I" PAGE DESCRIPTOR REGISTERS
        177600          UIPDR0= 177600
        177602          UIPDR1= 177602
        177604          UIPDR2= 177604
        177606          UIPDR3= 177606
        177610          UIPDR4= 177610
        177612          UIPDR5= 177612
        177614          UIPDR6= 177614
        177616          UIPDR7= 177616
                        ;*USER "D" PAGE DESCRIPTOR REGISTORS
        177620          UDPDR0= 177620
        177622          UDPDR1= 177622
        177624          UDPDR2= 177624
        177626          UDPDR3= 177626
        177630          UDPDR4= 177630
        177632          UDPDR5= 177632
        177634          UDPDR6= 177634
        177636          UDPDR7= 177636
                        ;*USER "I" PAGE ADDRESS REGISTERS
        177640          UIPAR0= 177640
        177642          UIPAR1= 177642
        177644          UIPAR2= 177644
        177646          UIPAR3= 177646
```

MEMORY MANAGEMENT DEFINITIONS

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

MEMORY MANAGEMENT DEFINITIONS

```
                172316                  KIPDR7= 172316
                                        ;*KERNEL "D" PAGE DESCRIPTOR REGISTERS
                172320                  KDPDR0= 172320
                172322                  KDPDR1= 172322
                172324                  KDPDR2= 172324
                172326                  KDPDR3= 172326
                172330                  KDPDR4= 172330
                172332                  KDPDR5= 172332
                172334                  KDPDR6= 172334
                172336                  KDPDR7= 172336
                                        ;*KERNEL "I" PAGE ADDRESS REGISTERS
                172340                  KIPAR0= 172340
                172342                  KIPAR1= 172342
                172344                  KIPAR2= 172344
                172346                  KIPAR3= 172346
                172350                  KIPAR4= 172350
                172352                  KIPAR5= 172352
                172354                  KIPAR6= 172354
                172356                  KIPAR7= 172356
                                        ;*KERNEL "D" PAGE ADDRESS REGISTERS
                172360                  KDPAR0= 172360
                172362                  KDPAR1= 172362
                172364                  KDPAR2= 172364
                172366                  KDPAR3= 172366
                172370                  KDPAR4= 172370
                172372                  KDPAR5= 172372
                172374                  KDPAR6= 172374
                172376                  KDPAR7= 172376
  1123                                  .SBTTL  BASIC DEFINITIONS
                                        ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
                001100                  STACK=  1100
                104000                  ERROR=  EMT             ;;BASIC DEFINITION OF ERROR CALL
                000004                  SCOPE=  IOT             ;;BASIC DEFINITION OF SCOPE CALL
                                        ;*MISCELLANEOUS DEFINITIONS
                000011                  HT=     11              ;;CODE FOR HORIZONTAL TAB
                000012                  LF=     12              ;;CODE FOR LINE FEED
                000015                  CR=     15              ;;CODE FOR CARRIAGE RETURN
                000200                  CRLF=   200             ;;CODE FOR CARRIAGE RETURN-LINE FEED
                177776                  PS=     177776          ;;PROCESSOR STATUS WORD
                177776                  PSW=    PS
                177774                  STKLMT= 177774          ;;STACK LIMIT REGISTER
                177772                  PIRQ=   177772          ;;PROGRAM INTERRUPT REQUEST REGISTER
                177570                  DSWR=   177570          ;;HARDWARE SWITCH REGISTER
                177570                  DDISP=  177570          ;;HARDWARE DISPLAY REGISTER
                                        ;*GENERAL PURPOSE REGISTER DEFINITIONS
                000000                  R0=     %0              ;;GENERAL REGISTER
                000001                  R1=     %1              ;;GENERAL REGISTER
                000002                  R2=     %2              ;;GENERAL REGISTER
                000003                  R3=     %3              ;;GENERAL REGISTER
                000004                  R4=     %4              ;;GENERAL REGISTER
                000005                  R5=     %5              ;;GENERAL REGISTER
                000006                  R6=     %6              ;;GENERAL REGISTER
                000007                  R7=     %7              ;;GENERAL REGISTER
                000006                  SP=     %6              ;;STACK POINTER
                000007                  PC=     %7              ;;PROGRAM COUNTER
                                        ;*PRIORITY LEVEL DEFINITIONS
                000000                  PR0=    0               ;;PRIORITY LEVEL 0
```

BASIC DEFINITIONS

```
000040                  PR1=    40              ;;PRIORITY LEVEL 1
000100                  PR2=    100             ;;PRIORITY LEVEL 2
000140                  PR3=    140             ;;PRIORITY LEVEL 3
000200                  PR4=    200             ;;PRIORITY LEVEL 4
000240                  PR5=    240             ;;PRIORITY LEVEL 5
000300                  PR6=    300             ;;PRIORITY LEVEL 6
000340                  PR7=    340             ;;PRIORITY LEVEL 7
                        ;*"SWITCH REGISTER" SWITCH DEFINITIONS
100000                  SW15=   100000
040000                  SW14=   40000
020000                  SW13=   20000
010000                  SW12=   10000
004000                  SW11=   4000
002000                  SW10=   2000
001000                  SW09=   1000
000400                  SW08=   400
000200                  SW07=   200
000100                  SW06    100
000040                  SW05=   40
000020                  SW04=   20
000010                  SW03=   10
000004                  SW02=   4
000002                  SW01=   2
000001                  SW00=   1
001000                  SW9=    SW09
000400                  SW8=    SW08
000200                  SW7=    SW07
000100                  SW6=    SW06
000040                  SW5=    SW05
000020                  SW4=    SW04
000010                  SW3=    SW03
000004                  SW2=    SW02
000002                  SW1=    SW01
000001                  SW0=    SW00
                        ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
100000                  BIT15=  100000
040000                  BIT14=  40000
020000                  BIT13=  20000
010000                  BIT12=  10000
004000                  BIT11=  4000
002000                  BIT10=  2000
001000                  BIT09=  1000
000400                  BIT08=  400
000200                  BIT07=  200
000100                  BIT06=  100
000040                  BIT05=  40
000020                  BIT04=  20
000010                  BIT03=  10
000004                  BIT02=  4
000002                  BIT01=  2
000001                  BIT00=  1
001000                  BIT9=   BIT09
000400                  BIT8=   BIT08
000200                  BIT7=   BIT07
000100                  BIT6=   BIT06
000040                  BIT5=   BIT05
000020                  BIT4=   BIT04
```

```
                    000010                  BIT3=   BIT03
                    000004                  BIT2=   BIT02
                    000002                  BIT1=   BIT01
                    000001                  BIT0=   BIT00
                                            ;*BASIC "CPU" TRAP VECTOR ADDRESSES
                    000004                  ERRVEC= 4               ;;TIME OUT AND OTHER ERRORS
                    000010                  RESVEC= 10              ;;RESERVED AND ILLEGAL INSTRUCTIONS
                    000014                  TBITVEC=14              ;;"T" BIT
                    000014                  TRTVEC= 14              ;;TRACE TRAP
                    000014                  BPTVEC= 14              ;;BREAKPOINT TRAP (BPT)
                    000020                  IOTVEC= 20              ;;INPUT/OUTPUT TRAP (IOT) **SCOPE**
                    000024                  PWRVEC= 24              ;;POWER FAIL
                    000030                  EMTVEC= 30              ;;EMULATOR TRAP (EMT) **ERROR**
                    000034                  TRAPVEC=34              ;;"TRAP" TRAP
                    000060                  TKVEC=  60              ;;TTY KEYBOARD VECTOR
                    000064                  TPVEC-  64              ;;TTY PRINTER VECTOR
                    000240                  PIRQVEC=240             ;;PROGRAM INTERRUPT REQUEST VECTOR
1124                000001                          UFDSET=         1               ;FLAG FOR UFD
1125                                        .SBTTL  TRAP CATCHER
                    000000                          .=0
                                            ;*ALL UNUSED LOCATIONS FROM 4   776 CONTAIN A ".+2,HALT"
                                            ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
                                            ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
                    000174                          .=174
     000174         000000                 DISPREG: .WORD  0               ;;SOFTWARE DISPLAY REGISTER
     000176         000000                 SWREG:   .WORD  0               ;;SOFTWARE SWITCH REGISTER
1126                000200                          .=200
1127 000200         005037  001160                  CLR     $TMP0
1128 000204         000137  004060                  JMP     @#START
1129                000220                          .=220
1130 000220         012737  000777  001160          MOV     #777,$TMP0
1131 000226         000137  004060                  JMP     @#START
1132
1133                                        .SBTTL  ACT11 HOOKS
                                            ;;*****************************************************************
                                            ;HOOKS REQUIRED BY ACT11
                    000232                          $SVPC=.                 ;SAVE PC
                    000046                          .=46
     000046         031732                          $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
                    000052                          .=52
     000052         000000                          .WORD   0               ;;2)SET LOC.52 TO ZERO
                    000232                          .=$SVPC                 ;; RESTORE PC
1134                                        .SBTTL  APT PARAMETER BLOCK
                                            ;;*****************************************************************
                                            ;SET LOCATIONS 24 AND 44 AS REQUIRED FOR APT
                                            ;;*****************************************************************
                    000232                          .$X=.   ;;SAVE CURRENT LOCATION
                    000024                          .=24    ;;SET POWER FAIL TO POINT TO START OF PROGRAM
     000024         000200                          200     ;;FOR APT START UP
                    000044                          .=44    ;;POINT TO APT INDIRECT ADDRESS PNTR.
     000044         000232                          $APTHDR ;;POINT TO APT HEADER BLOCK
                    000232                          .=.$X   ;;RESET LOCATION COUNTER
                                            ;;*****************************************************************
                                            ;SETUP APT PARAMETER BLOCK AS DEFINED IN THE APT-PDP11 DIAGNOSTIC
                                            ;INTERFACE SPEC.
     000232                                 $APTHD:
     000232         000000                 $HIBTS: .WORD   0       ;;TWO HIGH BITS OF 18 BIT MAILBOX ADDR.
```

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)
```

```
1135                                    .SBTTL  COMMON TAGS
                                        ;;*****************************************************************
                                        ;*THIS TABLE CONTAINS VARIOUS COMMON STORAGE LOCATIONS
                                        ;*USED IN THE PROGRAM.
            001100                              .=1100
001100                                  $CMTAG:                         ;;START OF COMMON TAGS
001100  000000                                  .WORD   0
001102     000                          $TSTNM: .BYTE   0               ;;CONTAINS THE TEST NUMBER
001103     000                          $ERFLG: .BYTE   0               ;;CONTAINS ERROR FLAG
001104  000000                          $ICNT:  .WORD   0               ;;CONTAINS SUBTEST ITERATION COUNT
001106  000000                          $LPADR: .WORD   0               ;;CONTAINS SCOPE LOOP ADDRESS
001110  000000                          $LPERR: .WORD   0               ;;CONTAINS SCOPE RETURN FOR ERRORS
001112  000000                          $ERTTL: .WORD   0               ;;CONTAINS TOTAL ERRORS DETECTED
001114     000                          $ITEMB: .BYTE   0               ;;CONTAINS ITEM CONTROL BYTE
001115     001                          $ERMAX: .BYTE   1               ;;CONTAINS MAX. ERRORS PER TEST
001116  000000                          $ERRPC: .WORD   0               ;;CONTAINS PC OF LAST ERROR INSTRUCTION
001120  000000                          $GDADR: .WORD   0               ;;CONTAINS ADDRESS OF 'GOOD' DATA
001122  000000                          $BDADR: .WORD   0               ;;CONTAINS ADDRESS OF  BAD' DATA
001124  000000                          $GDDAT: .WORD   0               ;;CONTAINS 'GOOD' DATA
001126  000000                          $BDDAT: .WORD   0               ;;CONTAINS 'BAD' DATA
001130  000000                                  .WORD   0               ;;RESERVED--NOT TO BE USED
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)
            000002                      .REPT   2
001160  000000                          $TMP0:  .WORD   0               ;;USER DEFINED
001162  000000                          $TMP1:  .WORD   0               ;;USER DEFINED
001164  000000                          $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
001166  000000                          $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
001170     207      377      377        $BELL:  .ASCIZ  <207><377><377> ;;CODE FOR BELL
001173     000
001174     077                          $QUES:  .ASCII  /?/             ;;QUESTION MARK
001175     015                          $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001176     012      000                 $LF:    .ASCIZ  <12>            ;;LINE FEED
                                        ;;*****************************************************************
                                        .SBTTL  APT MAILBOX-ETABLE
                                        ;;*****************************************************************
                                        .EVEN
001200                                  $MAIL:                  ;;APT MAILBOX
001200  000000                          $MSGTY: .WORD   AMSGTY   ;;MESSAGE TYPE CODE
001202  000000                          $FATAL: .WORD   AFATAL   ;;FATAL ERROR NUMBER
001204  000000                          $TESTN: .WORD   ATESTN   ;;TEST NUMBER
001206  000000                          $PASS:  .WORD   APASS    ;;PASS COUNT
001210  000000                          $DEVCT: .WORD   ADEVCT   ;;DEVICE COUNT
001212  000000                          $UNIT:  .WORD   AUNIT    ;;I/O UNIT NUMBER
001214  000000                          $MSGAD: .WORD   AMSGAD   ;;MESSAGE ADDRESS
```

APT MAILBOX-ETABLE

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

ERROR POINTER TABLE

```
                                .SBTTL  ERROR POINTER TABLE
                                ;*THIS TABLE CONTAINS THE INFORMATION FOR EACH ERROR THAT CAN OCCUR.
                                ;*THE INFORMATION IS OBTAINED BY USING THE INDEX NUMBER FOUND IN
                                ;*LOCATION $ITEMB. THIS NUMBER INDICATES WHICH ITEM IN THE TABLE IS PERTINENT.
                                ;*NOTE1:        IF $ITEMB IS 0 THE ONLY PERTINENT DATA IS ($ERRPC).
                                ;*NOTE2:        EACH ITEM IN THE TABLE CONTAINS 4 POINTERS EXPLAINED AS FOLLOWS:
                                ;*      EM              ;;POINTS TO THE ERROR MESSAGE
                                ;*      DH              ;;POINTS TO THE DATA HEADER
                                ;*      DT              ;;POINTS TO THE DATA
                                ;*      DF              ;;POINTS TO THE DATA FORMAT
     001324                     $ERRTB:
  1136
  1137
  1138                          .SBTTL  ERROR DEFINITIONS
  1139                          ;ITEM 1
  1140 001324  020617                   EM1             ;CPU ERROR
  1141 001326  025314                   DH1             ;TEST #, ERROR PC
  1142 001330  026502                   DT1             ;$TMP1,$ERRPC
  1143 001332  000000                   0
  1144                          ;ITEM 2
  1145 001334  020653                   EM2             ;MMU ERROR
  1146 001336  025314                   DH1             ;TEST #, ERROR PC
  1147 001340  026502                   DT1             ;$TMP1,$ERRPC
  1148 001342  000000                   0
  1149                          ;ITEM 3
  1150 001344  020665                   EM3             ;FPP ERROR
  1151 001346  025314                   DH1             ;TEST #, ERROR PC
  1152 001350  026502                   DT1             ;$TMP1,$ERRPC
  1153 001352  000000                   0
  1154                          ;ITEM 4
  1155 001354  020677                   EM54            ;CHECKSUM ERROR IN 16 BIT ROM
  1156 001356  025314                   DH1             ;TEST #, PC
  1157 001360  026502                   DT1             ;$TMP1,$ERRPC
  1158 001362  000000                   0
  1159                          ;ITEM 5
  1160 001364  020735                   EM56            ;TIMEOUT READING LKS
  1161 001366  025314                   DH1             ;TEST #, PC
  1162 001370  026502                   DT1             ;$TMP1,$ERRPC
  1163 001372  000000                   0
  1164                          ;ITEM 6
  1165 001374  020761                   EM57            ;LKS<07> DOES NOT BECOME 1
  1166 001376  025314                   DH1             ;TEST #, PC
  1167 001400  026502                   DT1             ;$TMP1,$ERRPC
  1168 001402  000000                   0
  1169                          ;ITEM 7
  1170 001404  021073                   EM64            ;WRONG NUMBER OF LKS INTERRUPTS
  1171 001406  025314                   DH1             ;TEST #, PC
  1172 001410  026502                   DT1             ;$TMP1,$ERRPC
  1173 001412  000000                   0
  1174                          ;ITEM 10
  1175 001414  021132                   EM65            ;LKS INTERRUPTS HAPPEN AT WRONG PRIORITY
  1176 001416  026145                   DH65            ;TEST #, PC, PRIORITY
  1177 001420  026736                   DT65            ;$TMP1,$ERRPC,$GDDAT
  1178 001422  000000                   0
  1179                          ;ITEM 11
  1180 001424  021202                   EM71            ;RESET DOESN'T CLEAR LKS<06>
  1181 001426  025314                   DH1             ;TEST #, PC
```

ERROR DEFINITIONS

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

ERROR DEFINITIONS

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

ERROR DEFINITIONS

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

ERROR DEFINITIONS

```
1353 002052  000000                    0
1354                          ;ITEM 54
1355 002054  023636                    EM145           ;TIMEOUT IN ACCESSING THE LKS REGISTER
1356 002056  025314                    DH1             ;TEST #, PC
1357 002060  026502                    DT1             ;$TMP1,$ERRPC
1358 002062  000000                    0
1359                          ;ITEM 55
1360 002064  023704                    EM146           ;ERROR IN STUCK AT ZERO BITS 0,1 MER
1361 002066  025314                    DH1             ;TEST #, PC
1362 002070  026502                    DT1             ;$TMP1,$ERRPC
1363 002072  000000                    0
1364                          ;ITEM 56
1365 002074  023747                    EM147           ;COULD NOT SET ONE OF THE R/W BITS ON THE MER
1366 002076  025341                    DH4             ;TEST #, PC, EXPECTED DATA, RECEIVED DATA
1367 002100  026746                    DT75            ;$TMP1,$ERRPC,$GDDAT,$BDDAT
1368 002102  000000                    0
1369                          ;ITEM 57
1370 002104  024024                    EM150           ;BITS 0,2,14,15 ON THE MER DID NOT CLEAR ON RESET
1371 002106  025314                    DH1             ;TEST #, PC
1372 002110  026502                    DT1             ;$TMP1,$ERRPC
1373 002112  000000                    0
1374                          ;ITEM 60
1375 002114  024105                    EM151           ;TIMEOUT IN ACCESSING THE MER REGISTER
1376 002116  025314                    DH1             ;TEST #, PC
1377 002120  026502                    DT1             ;$TMP1,$ERRPC
1378 002122  000000                    0
1379                          ;ITEM 61
1380 002124  024153                    EM152           ;ERROR IN THE DATA SHORTS AND STUCK AT MEMORY TEST
1381 002126  026364                    DH134           ;TEST #, PC,PATTERN,DATA,PAR,VIRTUAL ADDRESS
1382 002130  027012                    DT134           ;$TMP1,$ERRPC,$GDDAT,$BDDAT,KIPAR2,$BDADR
1383 002132  000000                    0
1384                          ;ITEM 62
1385 002134  024235                    EM153           ;PARITY ABORT OCCURED WITH PARITY ERROR DISABLED
1386 002136  025314                    DH1             ;TEST #, PC
1387 002140  026502                    DT1             ;$TMP1,$ERRPC
1388 002142  000000                    0
1389                          ;ITEM 63
1390 002144  024315                    EM154           ;PARITY ABORT DID NOT OCCUR WITH PARITY ERROR ENABLED
1391 002146  025314                    DH1             ;TEST #, PC
1392 002150  026502                    DT1             ;$TMP1,$ERRPC
1393 002152  000000                    0
1394                          ;ITEM 64
1395 002154  024402                    EM155           ;MER DIDN'T HAVE CORRECT ADDRESS BITS 11 17
1396 002156  025341                    DH4             ;TEST #, PC, EXPECTED DATA, RECEIVED DATA
1397 002160  026746                    DT75            ;$TMP1,$ERRPC,$GDDAT,$BDDAT
1398 002162  000000                    0
1399                          ;ITEM 65
1400 002164  024455                    EM156           ;MER READ EXTENDED ADDRESS BITS HAS FAILED
1401 002166  025341                    DH4             ;TEST #, PC, EXPECTED DATA, RECEIVED DATA
1402 002170  026746                    DT75            ;$TMP1,$ERRPC,$GDDAT,$BDDAT
1403 002172  000000                    0
1404                          ;ITEM 66
1405 002174  024527                    EM157           ;ERROR IN THE MEMORY
1406 002176  026364                    DH134           ;TEST #, PC,PATTERN,DATA,PAR,VIRTUAL ADDRESS
1407 002200  027012                    DT134           ;$TMP1,$ERRPC,$GDDAT,$BDDAT,KIPAR2,$BDADR
1408 002202  000000                    0
1409                          ;ITEM 67
```

ERROR DEFINITIONS

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

```
                                .SBTTL  GLOBAL VARIABLES AND REGISTER NAMES

                                ;REGISTERS FOR THE FIRST Q22BE
002334  000000                  CSR1:   .WORD   0                       ;CONTROL REGISTER 1 FOR Q22BE
002336  000000                  CSR2:   .WORD   0                       ;CONTROL/STATUS REGISTER 2
002340  000000                  BA:     .WORD   0                       ;DMA ADDRESS FOR Q22BE
002342  000000                  WC:     .WORD   0                       ;WORD COUNT REGISTER
002344  000000                  DATA:   .WORD   0                       ;DMA DATA FOR Q22BE
002346  000000                  VQBE1:  .WORD   0                       ;ADDRESS OF VECTOR FOR Q22BE
002350  000000                  VQPR1:  .WORD   0                       ;PRIORITY
002352  000000                  SIMGOA: .WORD   0                       ;SIMULTANEOUS GO ADDRESS REGISTER

002354  000000                  ledcnt: .word   0                       ;location for the led count     ;mc
002356  000000                  RCOUNT: .WORD   0                       ;RECEIVER INTERRUPT COUNT       ;MC
002360  000000                  TCOUNT: .WORD   0                       ;TRANSMITTER INTERRUPT COUNT    ;MC
002362  000000                  lopbak: .word   0                       ;flag for the loop back connector;mc

                                ;REGISTERS FOR THE SECOND Q22BE
002364  000000                  CSR12:  .WORD   0                       ;CONTROL REGISTER 1 FOR Q22BE
002366  000000                  CSR22:  .WORD   0                       ;CONTROL/STATUS REGISTER 2
002370  000000                  BA2:    .WORD   0                       ;DMA ADDRESS FOR Q22BE
002372  000000                  WC2:    .WORD   0                       ;WORD COUNT REGISTER
002374  000000                  DATA2:  .WORD   0                       ;DMA DATA FOR Q22BE
002376  000000                  VQBE2:  .WORD   0                       ;ADDRESS OF VECTOR FOR Q22BE
002400  000000                  VQPR2:  .WORD   0                       ;PRIORITY

002402  000000                  LKSFL:  .WORD   0
002404  000000                  lkcnt:  .word   0
002406  000000                  savloc: .word   0
002410  000000                  ACTCHS: .WORD   0                       ;ACTUAL CHECKSUM
002412  000000                  SAVPCR: .WORD   0
002414  000000                  SAVBR:  .WORD   0
        002740                  .=2740
002740  000000                  TEMP:   .WORD   0
002742                          .BLKW   15.                             ;RESERVED FOR BLOCK MODE TRANSFER
003000  000000                  TIMOUT: .WORD   0
003002  000002                  Q22EN:  .WORD   2               ;PRIORITY 7-4 FOR Q22BE

003004    377   021   377       tchar:  .byte   -1,21,-1,0      ;string for the loop back connector test
003007    000

          177524                        BCR=            177524          ;BOOT/DIAGNOSTICS CONFIGURATION
          177524                        BDR=            177524          ;BOOT/DIAGNOSTICS DISPLAY
          177520                        BCSR=           177520          ;BOOT/DIAGNOSTICS STATUS
          177746                        CCR=            177746          ;CACHE CONTROL REGISTER
          177752                        HITMIS=         177752          ;HIT OR MISS REGISTER
          177734                        KMCR=           177734          ;UNIBUS CONFIGURATION REGISTER
          177546                        LKS=            177546          ;CLOCK STATUS REGISTER
          177750                        MAIREG=         177750          ;MAINTENANCE REGISTER
          177520                        NATREG=         177520          ;NATIVE REGISTER
          172100                        MER=            172100          ;MEMORY ERROR REGISTER
          177744                        MSER=           177744          ;MEMORY SYSTEM ERROR
          177522                        PCR=            177522          ;PAGE CONTROL REGISTER
          177772                        PIR=            177772          ;PROGRAM INTERRUPT REQUEST
          177560                        RCSR=           177560          ;RECEIVER STATUS REGISTER
```

GLOBAL VARIABLES AND REGISTER NAMES

```
1523            177562                  RBUF=           177562          ;RECEIVER DATA BUFFER
1524            177564                  XCSR=           177564          ;TRANSMITTER STATUS REGISTER
1525            177566                  XBUF=           177566          ;TRANSMITTER DATA BUFFER
1526            176500                  RCSR1=          176500          ;RECEIVER STATUS REGISTER
1527            176502                  RBUF1=          176502          ;RECEIVER DATA BUFFER
1528            176504                  XCSR1=          176504          ;TRANSMITTER STATUS REGISTER
1529            176506                  XBUF1=          176506          ;TRANSMITTER DATA BUFFER
1530
1531            177766                  CPEREG=         177766          ;CPU ERROR REGISTER
1532
1533            177572                  MMR0=SR0                        ;MEMORY MANAGEMENT REG. 0
1534            177574                  MMR1=SR1                        ;MEMORY MANAGEMENT REG. 1
1535            177576                  MMR2=SR2                        ;MEMORY MANAGEMENT REG. 2
1536            172516                  MMR3=SR3                        ;MEMORY MANAGEMENT REG. 3
1537            120001                  POLY= 120001
1538            000000                  NULL=           0
1539            100000                  begrom  =       100000
1540
1541                                .SBTTL  GLOBAL DATA SECTION
1542
1543                                ;++
1544                                ; THE GLOBAL DATA SECTION CONTAINS DATA THAT ARE USED
1545                                ; IN MORE THAN ONE TEST.
1546                                ;
1547
1548                                ;THESE LOCATIONS ARE USED IN MORE THAN ONE TEST TO STORE VECTOR DATA
1549                                ;WHEN THE TEST NEEDS TO HAVE AN ERROR CONDITION RESPOND DIFFERENTLY
1550                                ;FROM THE DEFAULT RESPONCE.
1551 003010  000000                 SLOC00: .WORD   0
1552 003012  000000                 SLOC01: .WORD   0
1553
1554                                ;THESE LOCATIONS ARE USED IN MORE THAN ONE TEST TO STORE WORKING DATA.
1555 003014  000000                 PARPAT: .WORD   0               ;LOCATION TO SAVE PATTERN USED IN PARITY TEST
1556 003016  000000                 SAVTIM: .WORD   0               ;LOCATION TO THE SAVE THE TIMEOUT TRAP
1557 003020  000000                 LOWADD: .WORD   0               ;STORES LOW ADDRESS FOR RAM TESTS
1558 003022  000000                 GOODAD: .WORD   0               ;STORES GOOD ADDRESS FOR RAM TESTS
1559 003024  000000                 ERRCNT: .WORD   0               ; CONTAINS TOTAL NO. OF EEROM ERRORS
1560 003026  000000                 TSTADD: .WORD   0               ;ADDRESS STORE FOR RAM TESTS
1561 003030  000000                 NEWADD: .WORD   0               ;ADDRESS STORE FOR RAM TEST
1562 003032  000000                 FLAG:   .WORD   0               ;USED TO STORE "FLAG" CONDITIONS
1563 003034  000000                 CCHPAS: .WORD   0               ; flag-counter for control of Cache subtests
1564 003036  000000                 EEPAS:  .WORD   0               ; flag-counter for control of EEROM subtest
1565 003040  000000                 SAVSUP: .WORD   0               ;USED TO STORE SUPERVISOR STACK VALUE
1566 003042  000000                 SAVUSE: .WORD   0               ;USED TO STORE USER STACK VALUE
1567 003044  000000                 SAVMR0: .WORD   0               ;USED TO STORE MMU STATUS REGISTER 0 DATA
1568 003046  000000                 SAVMR1: .WORD   0               ;USED TO STORE MMU STATUS REGISTER 1 DATA
1569 003050  000000                 SAVMR2: .WORD   0               ;USED TO STORE MMU STATUS REGISTER 2 DATA
1570 003052  000000                 SAVSWR: .WORD   0               ; SAVE SFTWRE SWTCH REG DURING EEROM TEST
1571 003054  000000                 MERTAG: .WORD   0               ;
1572 003056                         FLOAT:  .BLKW   4               ;USED TO STORE VALUES FOR MMU TESTS
1573 003066                         FLO:    .BLKW   4               ;USED TO STORE VALUES FOR MMU TESTS
1574 003076  000000                 SEQ:    .WORD   0               ;STORES SEQUENCE NUMBER FOR JUMP TESTS
1575 003100  000000                 SPS:    .WORD   0               ;STORES STACK POINTER FOR JUMP TESTS
1576 003102  000000                 SPSJ:   .WORD   0               ;STORES STACK POINTER FOR JUMP TESTS
1577 003104                         BTEXP:  .BLKW   4               ;STORES EXPONENT DURING BIT TESTS
1578 003114                         BTRES:  .BLKW   4               ;STORES RECIEVED DATA FOR BIT TESTS
1579 003124  000000                 COUNT:  .WORD   0               ;ERROR INDICATOR FOR FLOATING POINT TESTS
```

GLOBAL DATA SECTION

```
1580 003126                          RECFEC: .BLKW   4               ;RECIEVED FLOATING POINT EXCEPTION CODE
1581 003136                          RECST:  .BLKW   4               ;RECIEVED FLOATING POINT STATUS
1582 003146                          RECDST: .BLKW   4               ;DESTINATION ADDRESS FOR FLOATING POINT TESTS
1583
1584                                 ;THESE LOCATIONS ARE USED BY MORE THAN ONE TEST AS LOOP COUNTERS
1585 003156  000000                  ALLCTR: .WORD   0
1586 003160  000000                  LOOPIN: .WORD   0
1587
1588                                 ;SOME MORE TEMPORARY STORAGE FOR RAM TESTS
1589 003162  000000                  SAVPOS: .WORD   0               ;STORES TEMPORARY BIT POSITIONS FOR RAM TESTS
1590 003164  000000                  MASK:   .WORD   0               ;STORES BIT MASK FOR ERROR ISOLATION
1591
1592                                 ;!!!!!THIS IS IT. THE PROGRAM TEST LOCATION!!!!!!!!!!!!!!!!!!
1593 003166  000000                  TSTLOC: .WORD   0
1594 003170                                  .BLKW   20.
1595                                         ;FPP REGISTER DEFINITIONS
1596          000000                         AC0= %0
1597          000001                         AC1= %1
1598          000002                         AC2= %2
1599          000003                         AC3= %3
1600          000004                         AC4= %4
1601          000005                         AC5= %5
1602          000006                         AC6= %6
1603          000007                         AC7- %7
1604
1605                                         ;FPP INTERRUPT VECTOR
1606
1607          000244                         FPVEC=244
1608
1609
1610          001000                         STBOT=  1000
1611
1612
1614 003240  123456                  TAB1:   .WORD   123456
1615 003242  000000                          .WORD   000000
1616 003244  000000                          .WORD   0
1617 003246  000001                          .WORD   1
1618 003250  055555                  TAB2:   .WORD   055555
1619 003252  177777                          .WORD   177777
1620 003254  145671                          .WORD   145671
1621 003256  100000                          .WORD   100000
1622 003260  003000                  TAB3:   .WORD   003000
1623 003262  123456                          .WORD   123456
1624 003264  000000                          .WORD   0
1625 003266  000000                          .WORD   0
1626 003270  055555                  TAB4:   .WORD   55555
1627 003272  177777                          .WORD   -1
1628 003274  000000                          .WORD   0
1629 003276  000000                          .WORD   0
1630 003300  043243                  TAB5:   .WORD   43243
1631 003302  000000                          .WORD   0
1632 003304  000000                          .WORD   0
1633 003306  000000                          .WORD   0
1634 003310  162400                  TAB5A:  .WORD   162400
1635 003312  000000                          .WORD   0
1636 003314  000000                          .WORD   0
1637 003316  000000                          .WORD   0
```

CLOBAL DATA SECTION

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

GLOBAL DATA SECTION

```
1690 003530 100400 000000 000000 TAB22:  .WORD   100400,0,0,0
     003536 000000
1691 003540 000200 000000 000000 TAB23:  .WORD   200,0,0,1
     003546 000001
1692 003550 062400 000000 000000 TAB24:  .WORD   62400,0,0,0
     003556 000000
1693 003560 001100 000000 000000 TAB25:  .WORD   1100,0,0,0
     003566 000000
1694 003570 100600 000000 000000 TAB26:  .WORD   100600,0,0,0
     003576 000000
1695 003600 001000 000000 000000 TAB27:  .WORD   1000,0,0,0
     003606 000000
1696 003610 000600 000000 000000 TAB28:  .WORD   600,0,0,0
     003616 000000
1697 003620 010100 000000 000000 TAB29:  .WORD   10100,0,0,0
     003626 000000
1698 003630 010100 000000 002000 TAB29A: .WORD   10100,0,2000,0
     003636 000000
1699
1700 003640 000500 000000 000000 TAB30:  .WORD   500,0,0,0
     003646 000000
1701 003650 100400 000000 000000 TAB31:  .WORD   100400,0,0,0
     003656 000000
1702 003660 016000 000000 000000 TAB32:  .WORD   16000,0,0,0
     003666 000000
1703 003670 011600 000000 000000 TAB33:  .WORD   11600,0,0,0
     003676 000000
1704 003700 000640 000000 000000 TAB34:  .WORD   640,0,0,0
     003706 000000
1705 003710 077600 000000 000000 TAB40:  .WORD   77600,0,0,0
     003716 000000
1706 003720 100200 000000 000000 TAB41:  .WORD   100200,0,0,1
     003726 000001
1707 003730 000340 000000 000000 TAB42:  .WORD   340,0,0,0
     003736 000000
1708 003740 000077 177777 177777 TAB43:  .WORD   77,177777,177777,177776
     003746 177776
1709 003750 000577 177777 177777 TAB45:  .WORD   577,-1,-1,-1
     003756 177777
1710 003760 000577 177777 000000 TAB46:  .WORD   577,-1,0,0
     003766 000000
1711 003770 173737 124242 052525 TAB47:  .WORD   173737,124242,052525,12346
     003776 012346
1712 004000 000000 000000 052525 TAB47A: .WORD   0,0,052525,12346
     004006 012346
1713 004010 173737 124242 000000 TAB48:  .WORD   173737,124242,0,0
     004016 000000
1714 004020 000600 000000 000000 TAB49:  .WORD   600,0,0,0
     004026 0000C0
1715
1716 004030 000000               DCOUNT  :       .WORD   0
1717 004032 000000               EXPDAT  :       .WORD   0
1718 004034 000000               RECDAT  :       .WORD   0
1719 004036 000000               PWDSEQ  :       .WORD   0
1720 004040 000000               ADLSB   :       .WORD   0
1721 004042 000000               RITEDA  :       .WORD   0
1722 004044 000000               NEWDAT  :       .WORD   0
```

GLOBAL DATA SECTION

```
1723 004046  000000                CURDAT  :       .WORD   0
1724 004050  000000                FSTADD  :       .WORD   0
1725 004052  000000                LSTADD  :       .WORD   0
1726 004054  000000                CURADD  :       .WORD   0
1727 004056  000000                PARABT  :       .WORD   0       ;PARIY ABORT FLAG
1728
1729
1730 004060                        START:
                                   ;; LCP/ORION ROUTINE TO SAVE EMTULATOR AND PRIORITY
     004060  005737  004146        EMTSAV: TST     SAV30                   ;; FIRST TIME THROUGH ?
     004064  001034                        BNE     VMKOR                   ;; BRANCH IF BEEN HERE ALREADY
     004066  032737  000040  000052        BIT     #BIT5,@#52              ;; ARE WE IN UFD MODE ?
     004074  001430                        BEQ     VMKOR                   ;; LEAVE IF NOT
     004076  012737  177777  004152        MOV     # 1,UFDFLG             ;; SET UFD FLAG
     004104  032737  000100  000052        BIT     #BIT6,@#52              ;; ARE WE IN QUIET MODE ?
     004112  001403                        BEQ     1$                      ;; BR IF NOT
     004114  012737  177777  004154        MOV     # 1,UQUIET             ;; SET QUIET MODE
     004122  104042                1$:     EMT     42                      ;; GET ADDRESS OF XXDP DCA TABLE
     004124  005060  000042                CLR     42(R0)                  ;; CLR XXDP+ "DRSERR"
     004130  013737  000030  004146        MOV     30,SAV30                ;; SAVE EMULATOR ADDRESS
     004136  013737  000032  004150        MOV     32,SAV32                ;; SAVE EMULATOR PRIORITY LEVEL
     004144  000404                        BR      VMKOR                   ;; GET AROUND TAG AREA
     004146  000000                SAV30:  .WORD   0                       ;; PUT EMULATOR INFO HERE
     004150  000000                SAV32:  .WORD   0                       ;; PUT PRIORITY LOCATION        HERE
     004152  000000                UFDFLG: .WORD   0                       ;; USER FRIENDLY MODE FLAG
     004154  000000                UQUIET: .WORD   0                       ;; UFD QUIET MODE FLAG
     004156                        VMKOR:
                                   ;*************************************************************************
1731 004156                        1$:
                                   .SBTTL  INITIALIZE THE COMMON TAGS
                                   ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
     004156  012706  001100                MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
     004162  005026                        CLR     (R6)+           ;;CLEAR MEMORY LOCATION
     004164  022706  001140                CMP     #SWR,R6 ;;DONE?
     004170  001374                        BNE     .-6             ;;LOOP BACK IF NO
     004172  012706  001100                MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                                   ;;INITIALIZE A FEW VECTORS
     004176  012737  031766  000020        MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
     004204  012737  000340  000022        MOV     #340,@#IOTVEC+2 ;;LEVEL 7
     004212  012737  032270  000030        MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
     004220  012737  000340  000032        MOV     #340,@#EMTVEC+2 ;;LEVEL 7
     004226  012737  035014  000034        MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
     004234  012737  000340  000036        MOV     #340,@#TRAPVEC+2;LEVEL 7
     004242  012737  035076  000024        MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
     004250  012737  000340  000026        MOV     #340,@#PWRVEC+2 ;;LEVEL 7
     004256  013737  031700  031672        MOV     $ENDCT,$EOPCT   ;;SETUP END-OF-PROGRAM COUNTER
     004264  005037  001164                CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
     004270  005037  001166                CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
     004274  112737  000001  001115        MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
     004302  012737  004302  001106        MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
     004310  012737  004310  001110        MOV     #.,$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.
     004316  013746  000004                MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
     004322  012737  004356  000004        MOV     #30000$,@#ERRVEC        ;;SET UP ERROR VECTOR
     004330  012737  177570  001140        MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
     004336  012737  177570  001142        MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
```

INITIALIZE THE COMMON TAGS

```
        004344  022777  177777  174566          CMP     @ 1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
        004352  001012                          BNE     30002$          ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                        ;;AND  THE HARDWARE SWR IS NOT = 1
        004354  000403                          BR      30001$          ;;BRANCH IF NO TIMEOUT
        004356  012716  004364          30000$: MOV     @30001$,(SP)    ;;SET UP FOR TRAP RETURN
        004362  000002                          RTI
        004364  012737  000176  001140  30001$: MOV     @SWREG,SWR      ;;POINT TO SOFTWARE SWR
        004372  012737  000174  001142          MOV     @DISPREG,DISPLAY
        004400  012637  000004          30002$: MOV     (SP)+,@@ERRVEC  ;;RESTORE ERROR VECTOR
        004404  005037  001206                  CLR     $PASS           ;;CLEAR PASS COUNT
        004410  132737  000200  001221          BITB    @APTSIZE,$ENVM  ;;TEST USER SIZE UNDER APT
        004416  001403                          BEQ     30003$          ;;YES,USE NON-APT SWITCH
        004420  012737  001222  001140          MOV     @$SWREG,SWR     ;;NO,USE APT SWITCH REGISTER
        004426                  30003$:
1732    004426  013737  004152  004154          MOV     UFDFLG,UQUIET   ;ABORT IN UFD ON ERROR
1733    004434  012737  030220  000004          MOV     @TOUT,@@ERRVEC  ;POINT TO TIMEOUT ROUTINE
1734    004442  012737  000340  000006          MOV     @340,@@ERRVEC+2 ;AT PRIORITY 7
1735    004450  012737  027464  000114          MOV     @RAMPAR,@@114   ;POINT PARITY ABORT
1736    004456  012737  000340  000116          MOV     @340,@@116      ;AT PRIORITY7
1737    004464  012737  030402  000250          MOV     @MMUTRP,@@250   ;POINT MMU TRAP VECTOR
1738    004472  012737  000340  000252          MOV     @340,@@252      ;
1739    004500  005037  177766                  CLR     @@177766        ;CLEAR CPU ERROR REGISTER
1740    004504  032737  000100  000052          BIT     @BIT06,@@52     ;IN UFD QUIET MODE ?
1741    004512  001164                          BNE     LOOP            ;IF SO, SKIP PRINTOUT
1742    004514  104401  017750                  TYPE    ,SWTSEL         ;
1743                            .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
        004520  005737  000042                  TST     @@42            ;;ARE WE RUNNING UNDER XXDP/ACT?
        004524  001012                          BNE     30004$          ;;BRANCH IF YES
        004526  123727  001220  000001          CMPB    $ENV,@1         ;;ARE WE RUNNING UNDER APT?
        004534  001406                          BEQ     30004$          ;;BRANCH IF YES
        004536  023727  001140  000176          CMP     SWR,@SWREG      ;;SOFTWARE SWITCH REG SELECTED?
        004544  001005                          BNE     30005$          ;;BRANCH IF NO
        004546  104406                          GTSWR                   ;;GET SOFT-SWR SETTINGS
        004550  000403                          BR      30005$
        004552  112737  000001  001134  30004$: MOVB    @1,$AUTOB       ;;SET AUTO-MODE INDICATOR
        004560                  30005$:
1744    004560  000240                          NOP                     ; debug aid
1745    004562  123727  001220  000001          CMPB    $ENV,@1         ; running under APT?
1746    004570  001020                          BNE     20$             ; default no-APT initialization
1747    004572  013700  001224                  MOV     $USWR,R0        ; work copy pass calculation
1748    004576  005700                          TST     R0              ; if = 0 default value
1749    004600  001420                          BEQ     25$             ; setup default value
1750
1751    004602  042700  000017                  BIC     @17,R0          ; clear low order nibble
1752    004606  000241                          CLC                     ; assure no unknowns
1753    004610  006000                          ROR     R0              ; 4 rotates = divide
1754    004612  006000                          ROR     R0              ; by 16 (=pass time)
1755    004614  006000                          ROR     R0              ; this area subroutined
1756    004616  006000                          ROR     R0              ; with general purpose
1757    004620  005700                          TST     R0              ; divide, this test to
1758    004622  001413                          BEQ     30$             ; determine skip altogether
1759
1760    004624  010037  003034                  MOV     R0,CCHPAS       ;residue = no. of desired passes
1761    004630  000413                          BR      35$             ; continue on
1762    004632  012737  177777  003034  20$:    MOV     @ 1,CCHPAS      ; largest number no apt mode??
1763    004640  000407                          BR      35$
1764    004642  012737  000001  003034  25$:    MOV     @1,CCHPAS       ; normal default = 1
```

GET VALUE FOR SOFTWARE SWITCH REGISTER

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

```
                                .SBTTL  TEST - NATIVE REGISTER
                                ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                                ; NATIVE REGISTER TEST
                                ;
                                ; This test checks out the native register's existance and it's
                                ; various bits.
                                ;
                                ; BGNTST
                                ;  Set up timeout vector PC to TIMOUT
                                ;  Set up t meout vector PSW to 7
                                ;  Read the Native register
                                ;  Check out the bootstrap switch as read only
                                ;  Check out the module funct onal revision as read only
                                ;  Check out that the self test enable bit works
                                ;  Check out he indicators (i.e LEDS)
                                ; ENDTST
                                ;
                                ; TIMOUT: Clean stack
                                ;         Error NATIVE register timed out
                                ;
                                ;-- --    ---- --------  -      ---   ---- ----   ---  ----   --- --


                                                                ;NATIVE REGISTER TEST
                                ;;*************************************************************
     005220  000004             TST1:   SCOPE
1838 005222  032777  010000  173710        BIT     #BIT12,@SWR     ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
1839 005230  001424                        BEQ     1000$           ;    THEN TYPE TEST TRACE
1840 005232  104401  005240                TYPE    ,30013$         ;;TYPE ASCIZ STRING
     005236  000421                        BR      30012$          ;;GET OVER THE ASCIZ
                                ;;30013$:       .ASCIZ  <15><12>/TEST 1  - NATIVE REGISTER TEST/
     005302                     30012$:
1841 005302                     1000$:
1842 005302  013737  000004  003016        MOV     @#4,SAVTIM              ;  SAVE UNEXPECTED TIMEOUT TRAP
1843 005310  012737  005464  000004        MOV     #100$,@#4               ;  SET UP TIMEOUT TRAP
1844 005316  012737  000340  000006        MOV     #340,@#6                ;
1845 005324  013700  177520                MOV     NATREG,R0               ;  READ THE NATIVE REGISTER
1846
1847                                ;
1848                                ; CHECK BITS 15-13 READ ONLY
1849                                ;
1850 005330  052737  160000  177520        BIS     #160000,NATREG          ;  TRY TO WRITE TO THE FUN REV BITS
1851 005336  013701  177520                MOV     NATREG,R1               ;  READ THE NATIVE REGISTER AGAIN
1852 005342  020001                        CMP     R0,R1                   ;  IF IT CHANGED THEN
1853 005344  001402                        BEQ     10$                     ;     ERROR IN NATIVE REGISTER
1854 005346  104047                        ERROR   +47                     ;
1855 005350  000447                        BR      110$                    ;
1856
1857                                ; CHECK THE INDICATOR BITS
1858                                ;
1859 005352  042737  000177  177520  10$:    BIC     #177,NATREG             ;  CLEAR THE INDICATOR BITS
1860 005360  032737  000177  177520        BIT     #177,NATREG             ;  IF THE BITS DIDN'T CLEAR
1861 005366  001403                        BEQ     20$                     ;  : THEN
1862 005370  104050                        ERROR   +50                     ;  :   ERROR IN THE NATIVE REGISTER
1863 005372  005037  172100                CLR     MER                     ;  CLEAR MER AFTER REPORTING THE ERROR ;MC
1864 005376  012701  000001        20$:    MOV     #1,R1                   ;  START PATTERN IN FIRST BIT
1865 005402  012702  000007                MOV     #7,R2                   ;  SET LOOP COUNT TO 7
```

TEST NATIVE REGISTER

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

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

TEST 16 BIT ROM CHECKSUM TEST

```
1945
1946 005740  104004                        error   +4                      ;nc, rom checksum error
1947 005742  005037  172100                clr     mer
1948 005746  013737  003016  000004        mov     savtim,@#4
1949 005754  005037  177572                clr     @#sr0
1950 005760  042737  000200  177520        bic     #bit07,@#natreg         ;disable the rom
1951 005766  000445                        Br      TST3            ;;go to the next test
1952
1953
1954
1955
1956 005770                          40$:
1957
1958
1959 005770  005037  172100                clr     mer                                     ;end of test
1960 005774  013737  003016  000004        mov     savtim,@#4
1961 006002  005037  177572                clr     @#sr0
1962 006006  042737  000200  177520        bic     #bit07,@#natreg         ;disable the rom
1963 006014  000432                        Br      TST3            ;;go to the next test
1964
1965
1966
1967                                 ;handle trap to 250
1968
1969 006016  042737  000200  177520  mmuerr: bic     #bit07,@#natreg         ;disable the rom
1970 006024  005037  177572                clr     @#sr0
1971 006030  104002                        error   +2                      ;mmu error
1972 006032  022626                        cmp     (sp)+,(sp)+
1973 006034  005037  172100                clr     mer
1974 006040  013737  003016  000004        mov     savtim,@#4
1975 006046  000415                        Br      TST3            ;;go to the next test
1976
1977
1978                                 ;handle trap to 4
1979
1980 006050                          timerr:
1981 006050  104041                        error   +41
1982 006052  005037  172100                clr     mer
1983 006056  022626                        cmp     (sp)+,(sp)+             ;clear stack
1984 006060  042737  000200  177520        bic     #bit07,@#natreg         ;disable the rom
1985 006066  005037  177572                clr     @#sr0
1986 006072  013737  003016  000004        mov     savtim,@#4
1987 006100  000400                        Br      TST3            ;;go to the next test
1988
1989
1990
1991
```

```
                                .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 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
                                ;
                                ;- -   --     - -- -- - --------  --- ------ --   --   -  -   -    -

                                                                                ;DATA PATH TESTS
                                ;;*****************************************************************
 006102  000004                 TST3:   SCOPE
 006104  032777  010000  173026         BIT     #BIT12,@SWR     ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
 006112  001425                         BEQ     1000$           ;    THEN TYPE TEST TRACE
 006114  104401  006122                 TYPE    ,30017$         ;;TYPE ASCIZ STRING
 006120  000422                         BR      30016$          ;;GET OVER THE ASCIZ
                                ;;30017$:       .ASCIZ  <15><12>/TEST 3    KDJ11-D DATA PATH TEST/
 006166                         30016$:
 006166                         1000$:
 006166  013737  000004  003016         MOV     @#4,SAVTIM              ;  SAVE UNEXPECTED TIMEOUT TRAP ;MC
 006174  012737  006276  000004         MOV     #100$,@#4               ;  SET UP THE TIMEOUT TRAP
 006202  012737  000340  000006         MOV     #340,@#6                ;
 006210  005737  000000                 TST     @#0                     ;  READ LOCATION 0
 006214  010701                         MOV     PC,R1                   ;  GET A POINTER TO THE PATTERNS
 006216  062701  000106                 ADD     #PATT-.,R1              ;
 006222  012702  000007                 MOV     #7,R2                   ;  SET THE LOOP COUNT TO 7
 006226                         5$:                                     ;  FOR ALL OF THE PATTERNS
 006226  011137  000000                 MOV     (R1),@#0                ;  : WRITE THE PATTERN TO 0
 006232  023721  000000                 CMP     @#0,(R1)+               ;  : IF THE PATTERN IS NOT READ BACK
 006236  001412                         BEQ     10$                     ;  : : THEN
 006240  013737  000000  001126         MOV     @#0,$BDDAT              ;  : :   GET READ DATA
 006246  016137  177776  001124         MOV     -2(R1),$GDDAT           ;  : :   GET EXPECTED DATA
 006254  104044                         ERROR   +44                     ;  : :   ERROR IN THE DATA PATHS
 006256  005037  172100                 CLR     MER                     ; CLR MER AFTER ERROR REPORTING ;MC
 006262  000401                         BR      15$                     ;  : END IF
 006264  077220                 10$:    SOB     R2,5$                   ;  ENDFOR
                                                                        ; END TEST
 006266  013737  003016  000004 15$:    MOV     SAVTIM,@#4              ;  RESTORE UNEXPECTED TIMEOUT TRAP
```

TEST KDJ11-DA DATA PATHS

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

```
                                        .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 TIMOUT
                                        ;   Set timeout priority to 7
                                        ;   For MSB_Address := 200000 to 1777777 DO
                                        ;       Contents of address := 0
                                        ;   Go to the next test
                                        ;
                                        ; TIMOUT:
                                        ;    Error Address should not have timed out
                                        ; ENDTST
                                        ;
                                        ;--   ---  -- - - ---   -----  -----   -- -- --    ---    - - ---    -

                                        ;;*****************************************************************
     006342  000004                     TST4:   SCOPE
2094 006344  032777  010000  172566             BIT     #BIT12,@SWR     ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
2095 006352  001426                     BEQ     1000$           ;    THEN TYPE TEST TRACE
2096 006354  104401  006362             TYPE    ,30019$         ;;TYPE ASCIZ STRING
     006360  000423                     BR      30018$          ;;GET OVER THE ASCIZ
                                        ;;30019$:       .ASCIZ  <15><12>/TEST 4  - MEMORY ACCESSIBILITY TEST/
     006430                             30018$:
2097 006430                             1000$:
2098 006430  013737  000004  003016     MOV     @#4,SAVTIM              ;  SAVE TIMEOUT
2099 006436  012737  006534  000004     MOV     #100$,@#4               ;  SET UP THE TIMEOUT TRAP
2100 006444  012737  000340  000006     MOV     #340,@#6                ;
2101 006452  004737  030676             JSR     PC,SETMMU               ;  GO SET UP THE MMU REGISTERS
2102 006456  012701  040000             MOV     #40000,R1               ;  WE WANT TO MAP THRU PAGE 2
2103 006462  012737  001600  172344     MOV     #1600,@#KIPAR2          ;  SET UP KPAR 2
2104 006470                             1$:                             ;  FOR 160000 TO 2000000
2105 006470  142721  000377             BICB    #377,(R1)+              ;  : CLEAR OUT A BYTE
2106 006474  020127  060000             CMP     R1,#60000               ;  : IF WE HAVE PASSED THE PAGE BOUNDARY
2107 006500  103773                     BLO     1$                      ;  : : THEN
2108 006502  062737  000200  172344     ADD     #200,@#KIPAR2           ;  : :  POINT KPAR2 TO A NEW PAGE
2109 006510  012701  040000             MOV     #40000,R1               ;  : :  SET THE VIRTUAL ADDRESS TO PAGE 2
2110 006514  023727  172344  020000     CMP     @#KIPAR2,#20000         ;  : ENDIF
2111 006522  001362                     BNE     1$                      ;  :
2112
2113 006524  013737  003016  000004     10$:    MOV     SAVTIM,@#4              ;  :
2114 006532  000416                     BR      TST5            ;;GO TO THE NEXT TEST
2115 006534                             20$:
2116                                    ;
2117                                    ; TIMEOUT ROUTINE
2118                                    ;
2119
2120 006534  012737  006552  000004     100$:   MOV     #102$,@#4
2121 006542  005301                     DEC     R1                      ;
2122 006544  104046                     ERROR   +46                     ; REPORT ERROR
2123 006546  005037  172100             CLR     MER
```

C5

SEQ 0054

TEST MEMORY ACCESSABILITY

```
2124 006552  022626                  102$:   CMP     (SP)+,(SP)+             ; CLEAN STACK
2125 006554  005037  177572                  CLR     @#SR0                   ;
2126 006560  013737  003016  000004          MOV     SAVTIM,@#4              ;  RESTORE TIMEOUT
2127 006566  000400                          BR      TST5            ;;GO TO THE NEXT TEST
2128
```

```
                                        .SBTTL  TEST   MEMORY ERROR REGISTER
                                        ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                                        ;MEMORY ERROR REGISTER
                                        ;
                                        ; This test will check the MEMORY ERROR REGISTER on the KDJ11 D
                                        ; Board.
                                        ;
                                        ; BGNTST
                                        ;   Setup timeout VECTOR PC to TIMOUT
                                        ;   Setup timeout VECTOR Prior'ty to 7
                                        ;   Setup Par'ty abort VECTOR to PARINT
                                        ;   Setup Par'ty abort VECTOR Priority to 7
                                        ;   Do a bus reset
                                        ;   Read the Memory Error Register (17772100)
                                        ;   Make sure that the reg'ster bits are in the r'ght state
                                        ;   Check all R/W bits
                                        ; ENDTST
                                        ;
                                        ;--     -           -     -- ---    --   - -    - -   -
                                        ;;*****************************************************************
     006570  000004                     TST5:   SCOPE
2150 006572  032777  010000  172340             BIT     #BIT12,@SWR     ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
2151 006600  001427                             BEQ     1000$           ;    THEN TYPE TEST TRACE
2152 006602  104401  006610                     TYPE    ,30021$         ;;TYPE ASCIZ STRING
     006606  000424                             BR      30020$          ;;GET OVER THE ASCIZ
     006660                             ;;30021$:       .ASCIZ  <15><12>/TEST 5  - MEMORY ERROR REGISTER TEST/
     006660                             30020$:
2153 006660                             1000$:
2154 006660  005037  004056                     clr     parabt                  ;init parity abort........mc
2155 006664  012737  000340  177776             mov     #340,@#177776           ;set the psw to 7 ........mc
2156 006672  122737  000001  001220             CMPB    #APTENV,$ENV            ;ARE WE IN APT MODE?
2157 006700  001005                             BNE     1001$                   ;IF NOT: DO THIS TEST
2158 006702  005737  001206                     TST     $PASS                   ;FIRST PASS?
2159 006706  001402                             BEQ     1001$                   ; YES THEN PROCEED
2160 006710  000137  007452                     JMP     PAREND                  ; NO THEN GO TO THE NEXT TEST
2161
2162 006714  01~737  000004  003016  1001$:  MOV     @#4,SAVTIM              ;  SAVE UNEXPECTED TIMEOUT TRAP
2163 006722  012737  007374  000004             MOV     #100$,@#4               ;  SET UP THE TIMEOUT TRAP
2164 006730  012737  000340  000006             MOV     #340,@#6                ;
2165 006736  013746  000114                     MOV     @#114,-(SP)             ;  SAVE VECTOR
2166 006742  013746  000116                     MOV     @#116,-(SP)             ;  SAVE PRIORITY
2167 006746  012737  007444  000114             MOV     #PARINT,@#114           ;
2168 006754  012737  000340  000116             MOV     #340,@#116              ;
2169 006762  005037  172100                     CLR     @#MER                   ;  READ THE MER
2170 006766  013700  172100                     MOV     @#MER,R0                ;
2171                                    ;
2172                                    ; CHECK OUT THE MER "ALWAYS READ AS 0" BITS
2173                                    ;
2174 006772  052737  030032  172100             BIS     #30032,MER              ;  TRY TO WRITE THE STUCK AT 0 BITS
2175 007000  013701  172100                     MOV     MER,R1                  ;  READ THE MER
2176 007004  020001                             CMP     R0,R1                   ;  IF THE STUCK AT 0 BITS CHANGED
2177 007006  001403                             BEQ     10$                     ;  : THEN
2178 007010  104055                             ERROR   +55                     ;  :  ERROR IN THE MER
2179 007012  005037  172100                     CLR     MER                     ;  ENDIF
2180                                    ; CHECK THAT BITS 0,2,14,15 ARE CLEARED ON RESET
2181                                    ;
2182 007016  052737  140005  172100  10$:    BIS     #140005,MER             ;  SET ALL THE R/W BITS ON THE BOARD
```

TEST MEMORY ERROR REGISTER

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

```
2240                                                                    ;     GET
2241                                                                    ;      A
2242                                                                    ;       PARITY ABORT
2243                                                                    ;        SOON !!!
2244                                                                    ;        .
2245                                                                    ;        .
2246                                                                    ;        .
2247                                                                    ;
2248 007214  042737  000001  172100          BIC     #BIT0,@#MER       ;  DISABLE PARITY ABORTS
2249 007222  022737  000001  004056          CMP     #1,PARABT         ;  IF WE DIDN'T GET A PARITY ABORT
2250 007230  001405                          BEQ     30$               ;  : THEN
2251 007232  005037  004056                  CLR     PARABT            ;  : CLEAR PARITY ABORT
2252 007236  104063                          ERROR   +63               ;  :   ERROR DID NOT GET A PARITY ABORT
2253 007240  005037  172100                  CLR     @#MER             ;  :  CLEAR MEMORY ERROR REGISTER ;MC
2254
2255                                 ;
2256                                 ; CHECK OUT THE ADDRESS AND EXTENDED MER READ ENABLED
2257                                 ;
2258 007244  005037  004056          30$:    CLR     PARABT            ;  :  CLEAR THE ABORT FLAG
2259 007250  042737  100000  172100          BIC     #BIT15,@#MER      ;  :  CLEAR THE PARITY ERROR FLAG
2260 007256  013700  172100                  MOV     @#MER,R0          ;  :  GET THE MEMORY ERROR REGISTER
2261 007262  042700  170037                  BIC     #170037,R0        ;  :  CLEAR THE UNNEEDED BITS
2262 007266  020027  007740                  CMP     R0,#7740          ;  :  IF NOT CORRECT BITS SET THEN
2263 007272  001410                          BEQ     35$               ;  :     ERROR IN ADDRESS 11 17
2264 007274  010037  001126                  MOV     R0,$BDDAT         ;  :     GET RECIEVED DATA
2265 007300  012737  007740  001124          MOV     #7740,$GDDAT      ;  :     GET EXPECTED DATA
2266 007306  104064                          ERROR   +64               ;  :
2267 007310  005037  172100                  CLR     @#MER             ;  :  CLEAR MEMORY ERROR REGISTER ;MC
2268
2269 007314  052737  040000  172100  35$:    BIS     #BIT14,@#MER      ;  :  SET ENABLE READ EXTENDED BIT
2270 007322  013702  172100                  MOV     @#MER,R2          ;  :  READ THE MER AGAIN
2271 007326  042702  177037                  BIC     #177037,R2        ;  :  CLEAR THE UNNEEDED BITS
2272 007332  020227  000040                  CMP     R2,#40            ;  :  IF NOT CORRECT BITS SET THEN
2273 007336  001410                          BEQ     40$               ;  :     ERROR IN ADDR 18 21 OR READ EXT BIT
2274 007340  010237  001126                  MOV     R2,$BDDAT         ;  :     GET RECIEVED DATA
2275 007344  012737  000040  001124          MOV     #40,$GDDAT        ;  :     GET EXPECTED DATA
2276 007352  104065                          ERROR   +65               ;  ENDIF
2277 007354  005037  172100                  CLR     @#MER             ;  :  CLEAR MEMORY ERROR REGISTER ;MC
2278
2279 007360  005011                  40$:    CLR     (R1)              ;
2280 007362  005037  177572                  CLR     @#SR0             ;  DISABLE MMU
2281 007366  005037  172100                  CLR     @#MER             ;
2282 007372  000407                          BR      101$              ;  ENDTST
2283                                 ;
2284                                 ; TIMEOUT ROUTINE
2285                                 ;
2286
2287 007374  104060                  100$:   ERROR   +60               ;  REPORT ERROR
2288 007376  005037  172100                  CLR     @#MER             ;  :  CLEAR MEMORY ERROR REGISTER ;MC
2289
2290 007402  022626                          CMP     (SP)+,(SP)+       ;  CLEAN STACK
2291 007404  005011                          CLR     (R1)              ;
2292 007406  005037  177572                  CLR     @#SR0             ;  DISABLE MMU
2293
2294 007412  012737  007424  000004  101$:   MOV     #102$,@#4                         ;MC
2295 007420  005037  172100                  CLR     @#MER             ;  :  CLEAR MEMORY ERROR REGISTER ;MC
2296 007424                          102$:
```

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

```
 2310
 2311
 2312
 2313                                   .SBTTL  TEST   DATA SHORTS AND STUCK AT BITS
 2314                                   ;*********************************************************************
 2315                                   ;DATA SHORTS AND STUCK AT BITS TEST
 2316                                   ;
 2317                                   ; This test will check the DATA Rams for Data shorts and stuck at bits.
 2318                                   ; Testing occurs as below:
 2319                                   ;
 2320                                   ;       1. A memory location is checked to be set to 0
 2321                                   ;       2. IF NOT 0 error
 2322                                   ;       3. The location is complemented
 2323                                   ;       4. IF contents NOT -1 error
 2324                                   ;       5. This is repeated for all addresses
 2325                                   ;       6. Steps 1 5 are done from location 1777777 to 0
 2326                                   ;
 2327                                   ; Note: The KDJ11 DA board uses 256k X 1 chips,This allows us to
 2328                                   ;       use only 2 patterns per RAM. IF one b'ts is shorted to
 2329                                   ;       another we will detect this when we read for the initial state.
 2330                                   ;       If it has changed from the expected state then chances are that
 2331                                   ;       the bits are shorted together.
 2332                                   ;
 2333                                   ; A side effect of this test should be that memory is cleared.
 2334                                   ;
 2335                                   ; BGNTST
 2336                                   ;   ENABLE PARITY
 2337                                   ;   FOR MSB Address := 0 to 1.77776 DO BY 2
 2338                                   ;       Clear Address
 2339                                   ;       If Contents <> 0 THEN
 2340                                   ;          ERROR in memory
 2341                                   ;       Complement the Contents of Address
 2342                                   ;       IF contents <> -1 THEN
 2343                                   ;          ERROR in memory
 2344                                   ;   ENDFOR
 2345                                   ;   FOR MSB_Address := 1777776 DOWNTO 0 DO BY 2
 2346                                   ;       IF contents <> -1 THEN
 2347                                   ;          ERROR in memory
 2348                                   ;       Complement the Contents of Address
 2349                                   ;       IF contents <> 0 THEN
 2350                                   ;          ERROR in memory
 2351                                   ;   ENDFOR
 2352                                   ; ENDTST
 2353                                   ;
 2354                                   ;   -------------- - ---  ---  -   -   -    ----- -    -    -- --  --
 2355                                   ;;*********************************************************************
      007452  000004                   TST6:   SCOPE
 2356 007454  032777  010000  171456           BIT     #BIT12,@SWR     ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
 2357 007462  001433                           BEQ     1000$           ;    THEN TYPE TEST TRACE
 2358 007464  104401  007472                   TYPE    ,30023$         ;;TYPE ASCIZ STRING
      007470  000430                           BR      30022$          ;;GET OVER THE ASCIZ
                                       ;;30023$:       .ASCIZ  <15><12>/TEST 6    DATA SHORTS AND STUCK AT BITS TEST/
      007552                           30022$:
 2359 007552                           1000$:
 2360 007552  005037  177572                   CLR     @#SR0                   ;
 2361 007556  013737  000004  003016           MOV     @#4,SAVTIM              ;STORE UNEXPECTED TIMEOUT
 2362 007564  012737  010130  000004           MOV     #100$,@#4               ;  SET UP THE TIMEOUT TRAP
```

TEST - DATA SHORTS AND STUCK AT BITS

```
 2363 007572  012737  000340  000006          MOV     #340,@#6
 2364 007600  004737  030676                  JSR     PC,SETMMU       ;  GO SET UP THE MMU REGISTERS
 2365 007604  012701  040000                  MOV     #40000,R1       ;  WE WANT TO MAP THRU PAGE 2
 2366 007610  012737  001600  172344          MOV     #1600,@#KIPAR2  ;  SET UP KPAR 2
 2367 007616  013746  000114                  MOV     @#114,-(SP)     ;  SAVE VECTOR  ;MC
 2368 007622  013746  000116                  MOV     @#116,-(SP)     ;  SAVE PRIORITY ;MC
 2369 007626  012737  010116  000114          MOV     #50$,@#114      ;  set up trap                  ;mc
 2370 007634  012737  000340  000116          MOV     #340,@#116      ;   "  "   "                    ;mc
 2371 007642  052737  000001  172100          BIS     #BIT0,@#MER     ;  ENABLE PARITY ABORTS         ;mc
 2372
 2373 007650                          10$:                            ;  FOR ADDRESS 160000 TO 2000000 DO
 2374 007650  011137  001126                  MOV     (R1),$BDDAT     ;  : READ MEMORY
 2375 007654  001406                          BEQ     15$             ;  : IF IT IS NOT ZERO THEN
 2376 007656  005037  001124                  CLR     $GDDAT          ;  :    GET THE EXPECTED DATA
 2377 007662  010137  001122                  MOV     R1,$BDADR       ;  :    GET THE VIRTUAL ADDRESS
 2378 007666  104061                          ERROR   +61             ;  :    ERROR IN THE MEMORY
 2379 007670  000531                          BR      101$            ;  RESTORE AND TO THE NEXT TEST
 2380
 2381 007672  005111                  15$:    COM     (R1)            ;  : COMPLEMENT MEMORY
 2382 007674  011137  001126                  MOV     (R1),$BDDAT     ;  : SAVE THE RECIEVED DATA
 2383 007700  022127  177777                  CMP     (R1)+,#177777   ;  : IF ITS NOT = -1 THEN
 2384 007704  001412                          BEQ     20$             ;  :    ERROR IN THE MEMORY
 2385 007706  012737  177777  001124          MOV     #-1,$GDDAT      ;  :    GET EXPECTED DATA
 2386 007714  010137  001122                  MOV     R1,$BDADR       ;  :    GET THE VIRTUAL ADDRESS
 2387 007720  162737  000002  001122          SUB     #2,$BDADR       ;  :
 2388 007726  104061                          ERROR   +61             ;  :
 2389 007730  000511                          BR      101$            ;  RESTORE AND TO THE NEXT TEST
 2390
 2391 007732  020127  060000          20$:    CMP     R1,#60000       ;  : IF WE HAVE PASSED THE PAGE BOUNDARY
 2392 007736  103744                          BLO     10$             ;  : : THEN
 2393 007740  062737  000200  172344          ADD     #200,@#KIPAR2   ;  : :  POINT KPAR2 TO A NEW PAGE
 2394 007746  012701  040000                  MOV     #40000,R1       ;  : :  SET THE VIRTUAL ADDRESS TO PAGE 2
 2395 007752  023727  172344  020000          CMP     @#KIPAR2,#20000 ;  : ENDIF
 2396 007760  001401                          BEQ     25$             ;  ENDFOR
 2397 007762  000732                          BR      10$             ;
 2398 007764                          25$:
 2399 007764  004737  030610                  jsr     pc,delay        ;       MOV     #200,R5                 ;  THIS LOOP
IS IN HERE TO TEST THE DATA RETENTION OF MEMORY.
 2400                                                                 ;26$:   MOV     #177777,R4              ;
 2401                                                                 ;27$:   SOB     R4,27$                  ;
 2402                                                                 ;       SOB     R5,26$                  ;
 2403 007770  012701  060000                  MOV     #60000,R1       ;  GET A POINTER TO THE TOP OF A PAGE
 2404 007774  162737  000200  172344          SUB     #200,@#KIPAR2   ;  SET KIPAR TO POINT TO THE TOP PAGE
 2405 010002                          30$:                            ;  FOR LAST ADDRESS TO FIRST BY 2
 2406 010002  014137  001126                  MOV     -(R1),$BDDAT    ;  : SAVE THE DATA
 2407 010006  023727  001126  177777          CMP     $BDDAT,#177777  ;  : READ BACK MEMORY MAKE SURE IT IS  1
 2408 010014  001407                          BEQ     35$             ;  : IF NOT = -1 THEN
 2409 010016  012737  177777  001124          MOV     #177777,$GDDAT  ;  :   GET THE EXPECTED DATA
 2410 010024  010137  001122                  MOV     R1,$BDADR       ;  :   GET THE VIRTUAL ADDRESS
 2411 010030  104061                          ERROR   +61             ;  :   ERROR IN MEMORY
 2412 010032  000450                          BR      101$            ;  RESTORE AND TO THE NEXT TEST
 2413
 2414 010034  005111                  35$:    COM     (R1)            ;  : COMPLEMENT MEMORY
 2415 010036  011137  001126                  MOV     (R1),$BDDAT     ;  : IF NOT = 0 THEN
 2416 010042  001406                          BEQ     40$             ;  :   ERROR IN MEMORY
 2417 010044  005037  001124                  CLR     $GDDAT          ;  :   GET EXPECTED DATA
 2418 010050  010137  001122                  MOV     R1,$BDADR       ;  :   GET THE VIRTUAL ADDRESS
 2419 010054  104061                          ERROR   +61             ;  .
```

TEST - DATA SHORTS AND STUCK AT BITS

```
2420 010056  000436                          BR      101$                    ;  RESTORE AND TO THE NEXT TEST
2421
2422 010060  020127  040000          40$:    CMP     R1,#40000               ;  : IF VIRTUAL ADDRESS ABOUT TO ENTER NEXT PAGE
2423 010064  101346                          BHI     30$                     ;  : : THEN
2424 010066  162737  000200  172344          SUB     #200,@#KIPAR2           ;  : :  ADJUST THE PAR
2425 010074  012701  060000                  MOV     #60000,R1               ;  : :  RESET THE VIRTUAL ADDRESS TO TOP OF PAGE
2426 010100  023727  172344  001400          CMP     @#KIPAR2,#1400          ;  : ENDIF
2427 010106  001335                          BNE     30$                     ;  ENDFOR
2428 010110  005037  172100                  CLR     MER                     ;  CLEAR THE MER
2429 010114  000417                          BR      101$                    ;  RESTORE AND TO THE NEXT TEST
2430
2431                                 ; PARITY ABORT ROUTINE
2432                                 ;
2433
2434 010116  104072                  50$:    ERROR   .72                     ; REPORT ERROR
2435 010120  005037  172100                  CLR     MER                     ; INIT THE MER
2436 010124  022626                          CMP     (SP)+,(SP)+             ; CLEAN STACK
2437
2438                                 ;VECTOR SAVE AND RESTORE
2439                                 ;UNEXPECTED PARITY ABORT ERROR ROUTINE
2440 010126  000412                          BR      101$
2441
2442                                 ;
2443                                 ; TIMEOUT ROUTINE
2444                                 ;
2445
2446 010130  012737  010160  000004  100$:    MOV     #102$,@#4
2447 010136  005301                          DEC     R1                      ;
2448 010140  104046                          ERROR   .46                     ; REPORT ERROR
2449 010142  022626                          CMP     (SP)+,(SP)+             ; CLEAN STACK
2450 010144  005037  172100                  CLR     MER                     ;
2451 010150  005037  177572                  CLR     @#SR0                   ;
2452
2453 010154  005037  172100          101$:   CLR     MER                     ; INIT THE MER
2454
2455 010160                          102$:
2456 010160  012637  000116                  MOV     (SP)+,@#116             ;  RESTORE PRIORITY
2457 010164  012637  000114                  MOV     (SP)+,@#114             ;  RESTORE VECTOR
2458 010170  013737  003016  000004          MOV     SAVTIM,@#4              ;  RESTORE UNEXPECTED TIMEOUT
2459 010176  000400                          BR      TST7            ;;GO TO THE NEXT TEST
2460
2461
```

```
                                        .SBTTL TEST - QUICK VERIFY DATA AND ADRESSING TEST
                                        ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
                                        ;UNIQUE ADDRESSING TEST
                                        ;
                                        ; This test will check the data and addressing of the memories.
                                        ; It uses the KNAIZUK HARTMANN QUICK VERIFY TRAM TEST ALGORITHM.
                                        ; Th's 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 conta'n 0
                                        ; 3. Write a  1 into the 2nd address of all groups
                                        ; 4. Make sure that the 3rd address of all groups conta'n 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
                                        ;
                                        ;
                                        ; ----   ----   --- -- ---- ----- -- - --- -       ---

                                        ;;*****************************************************************
     010200  000004                     TST7:   SCOPE
2493 010202  032777  010000  170730             BIT     #BIT12,@SWR     ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
2494 010210  001434                             BEQ     1000$           ;    THEN TYPE TEST TRACE
2495 010212  104401  010220                     TYPE    ,30025$         ;;TYPE ASCIZ STRING
     010216  000431                             BR      30024$          ;;GET OVER THE ASCIZ
                                        ;;30025$:       .ASCIZ  <15><12>/TEST 7  - QUICK VERIFY DATA AND ADDRESSING TEST/
     010302                             30024$:
2496 010302                             1000$:
2497 010302  004737  030676                     JSR     PC,SETMMU       ; SET UP THE MMU REGISTERS
2498 010306  013737  000004  003016             MOV     @#4,SAVTIM      ; STORE UNEXPECTED TIMEOUT
2499 010314  013737  010476  000004             MOV     100$,@#4        ; TIMEOUT
2500 010322  004737  031012                     CALL    INIMEM          ; INITIALIZE MEMORY
2501 010326  012704  000001                     MOV     #1,R4           ; SET ADDRESS INDEX TO 1
2502 010332  005005                             CLR     R5              ; SET EXPECTED PATTERN TO 0
2503 010334  005003                             CLR     R3              ; SET OPERATION TO READ
2504 010336  004737  031176                     CALL    RWTMEM          ; GO READ AND CHECK MEMORY
2505 010342  012704  000001                     MOV     #1,R4           ; SET ADDRESS INDEX TO 1
2506 010346  012705  177777                     MOV     #-1,R5          ; SET WRITE PATTERN TO -1
2507 010352  012703  000001                     MOV     #1,R3           ; SET OPERATION TO WRITE
2508 010356  004737  031176                     CALL    RWTMEM          ; GO WRITE THE MEMORY
2509 010362  012704  000002                     MOV     #2,R4           ; SET ADDRESS INDEX TO 2
2510 010366  005005                             CLR     R5              ; SET EXPECTED PATTERN TO 0
2511 010370  005003                             CLR     R3              ; SET OPERATION TO READ
2512 010372  004737  031176                     CALL    RWTMEM          ; GO READ AND CHECK MEMORY
2513 010376  004737  031350                     CALL    RRMEM           ; READ AND CHECK I AND I+1 INDEXES
2514 010402  005004                             CLR     R4              ; SET ADDRESS INDEX TO 0
2515 010404  005005                             CLR     R5              ; SET PATTERN TO 0
```

TEST - QUICK VERIFY DATA AND ADRESSING TEST

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

```
                                        .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 occured
                                        ;     RTI
                                        ;ENDTST
                                        ;   --    --  ---- ---- ----  ------- -  ----------    --  --   -
                                        ;;*****************************************************************
    010532  000004                      TST10:  SCOPE
2582 010534  122737  0000C   001220             CMPB    @APTENV,$ENV            ;ARE WE IN APT MODE?
2583 010542  001005                      BNE     2000$                   ;IF NOT: DO THIS TEST
2584 010544  005737  0012C               TST     $PASS                   ;FIRST PASS??
2585 010550  001402                      BEQ     2000$                   ;IF YES, DO IT
2586 010552  000137  011552              JMP     NXTST                   ;ELSE GO TO THE NEXT TEST
2587 010556  032777  000200   170354  2000$:  BIT     @BIT7,@SWR      ; IF BIT7 SET IN SWR THEN SKIP THIS TEST
2588 010564  001402                      BEQ     100$                    ;
2589 010566  000137  011552              JMP     NXTST                   ;
2590 010572  032777  010000   170340  100$:   BIT     #BIT12,@SWR     ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
2591 010600  001426                      BEQ     1000$                   ;    THEN TYPE TEST TRACE
2592 010602  104401  010610              TYPE    ,30027$                 ;;TYPE ASCIZ STRING
     010606  000423                      BR      30026$                  ;;GET OVER THE ASCIZ
                                        ;;30027$:       .ASCIZ  <15><12>/TEST 10   CHECK PARITY DETECT TEST/
     010656                             30026$:
2593 010656                             1000$:
2594 010656  005037  172100              CLR     @#MER                   ; INIT THE MEMORY ERROR REG.
2595 010662  005037  004056              CLR     PARABT                  ;
2596 010666  013746  000114              MOV     @#114, (SP)             ; SAVE VECTORE
2597 010672  013746  000116              MOV     @#116, (SP)             ; SAVE PRIORITY
2598 010676  012737  007444  000114      MOV     #PARINT,@#114           ;
2599 010704  012737  000340  000116      MOV     #340,@#116              ;
```

```
2600                                    ;
2601                                    ; CLEAR ALL OF MEMORY WITH WRITE WRONG PARITY SET
2602                                    ;
2603 010712  004737  030676             JSR     PC,SETMMU               ;  GO SET UP THE MMU REGISTERS
2604 010716  105037  003014             CLRB    PARPAT                  ;  INITIALIZE FIRST PATTERN TO ZERO
2605 010722  012701  040000             MOV     #40000,R1               ;  WE WANT TO MAP THRU PAGE 2
2606 010726  012737  001600  172344     MOV     #1600,@#KIPAR2          ;  SET UP KPAR 2
2607 010734  012703  000002             MOV     #2,R3                   ;  LOOP FOR ODD PARITY AND EVEN PARITY
2608
2609
2610 010740                             1$:
2611                                                                    ;  BGNDO
2612 010740  012737  000004  172100             MOV     #4,@#MER        ;  : ENABLE WRITE WRONG PARITY  ;MC
2613 010746  113721  003014             MOVB    PARPAT,(R1)+            ;  : CLEAR OUT A BYTE
2614 010752  005037  172100             CLR     @#MER                   ;  : DISABLE WRITE WRONG PARITY ;MC
2615                                                                    ;  : FOR 160000 TO 2000000
2616 010756  020127  060000             CMP     R1,#60000               ;  : : IF WE HAVE PASSED THE PAGE BOUNDARY
2617 010762  103766                     BLO     1$                      ;  : : : THEN
2618 010764  062737  000200  172344     ADD     #200,@#KIPAR2           ;  : : :  POINT KPAR2 TO A NEW PAGE
2619 010772  012701  040000             MOV     #40000,R1               ;  : : :  SET THE VIRTUAL ADDRESS TO PAGE 2
2620 010776  023727  172344  020000     CMP     @#KIPAR2,#20000         ;  : : ENDIF
2621 011004  001355                     BNE     1$                      ;  : ENDFOR
2622                                    ;
2623                                    ; LET'S NOW MAKE SURE THAT THE ABORTS OCCUR
2624                                    ;
2625 011006                             20$:
2626 011006  012737  001600  172344             MOV     #1600,@#KIPAR2  ;  : SET UP PAR 2
2627 011014  012701  040000             MOV     #40000,R1               ;  : SET VIRTUAL ADDRESS TO PAGE #2
2628 011020                             22$:                            ;  : BGNDO
2629 011020  042737  000004  172100             BIC     #BIT2,@#MER     ;  : : DISABLE WRITE WRONG PARITY
2630 011026  052737  000001  172100             BIS     #BIT0,@#MER     ;  : : ENABLE PARITY ERRORS
2631 011034  105711                     TSTB    (R1)                    ;  : : READ THE BYTE
2632 011036  005737  003054             TST     MERTAG                  ;  : :
2633 011042  000240                     NOP                             ;  : :
2634                                                                    ;  : :PARITY
2635                                                                    ;  : : ABORT
2636                                    ;                               ;  : :  TRAP
2637                                    ;                               ;  : :   SHOULD
2638                                    ;                               ;  : :    HAPPEN
2639                                    ;                               ;  : :     SOON !!!
2640                                    ;                               ;  : :
2641                                    ;                               ;  : :
2642 011044  023727  004056  000001             CMP     PARABT,#1       ;  : : IF A PARITY ABORT DIDN'T OCCUR
2643 011052  001410                     BEQ     25$                     ;  : : : THEN
2644 011054  005037  004056             CLR     PARABT                  ;  : : .   (EITHER 0 OR >1 ABORTS HAVE OCCURED)
2645 011060  005037  172100             CLR     MER                     ;  : : :   CLEAR OUT THE MER
2646 011064  104063                     ERROR   +63                     ;  : : :   ERROR DID NOT GET A PARITY ABORT
2647 011066  005037  172100             clr     mer
2648 011072  000465                     BR      40$                     ;  : : : ELSE
2649                                    ;
2650                                    ; CHECK OUT THE ADDRESS AND EXTENDED MER READ ENABLED
2651                                    ;
2652 011074                             25$:
2653 011074  032737  100001  172100             BIT     #100001,@#MER   ;  : : : :  ARE BITS 0 AND 15 ARE SET ;MC
2654 011102  001003                     BNE     30$                     ;  : : : :  YES THEY ARE              ;MC
2655 011104  104073                     ERROR   +73                     ;  : : : :  NO THEY'RE NOT SET        ;MC
2656 011106  005037  172100             clr     mer
```

TEST CHECK PARITY DETECT LOGIC AND RAMS

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

```
2714 011416  000762                          BR      60$                     ;
2715                                 ;
2716                                 ; LET'S NOW MAKE SURE THAT THE ABORTS DON T OCCUR
2717                                 ;
2718 011420                          70$:
2719 011420  012737  001600  172344          MOV     #1600,@#KIPAR2          ;  SET UP PAR 2
2720 011426  012701  040000                  MOV     #40000,R1               ;  SET VIRTUAL ADDRESS TO PAGE #2
2721 011432                          75$:                                    ;  BGNDO
2722 011432  052737  000001  172100          BIS     #BIT0,@#MER             ;  : ENABLE PARITY ERRORS
2723 011440  105711                          TSTB    (R1)                    ;  : READ THE BYTE
2724 011442  012737  177777  003054          MOV     # 1,MERTAG               ;
2725                                                                         ;  : A
2726                                                                         ;  :  PARITY
2727                                                                         ;  :   ABORT
2728                                                                         ;  :    TRAP
2729                                                                         ;  :     SHOULD NOT
2730                                                                         ;  :      HAPPEN !!!
2731                                                                         ;  :
2732                                                                         ;  :
2733                                                                         ;  :
2734 011450  005737  004056                  TST     PARABT                  ;  : IF A PARITY ABORT HAS OCCURED
2735 011454  001407                          BEQ     80$                     ;  : : THEN
2736 011456  005037  004056                  CLR     PARABT                  ;  : : :
2737 011462  005037  172100                  CLR     MER                     ;  : :   INITIALIZE THE MER
2738 011466  104063                          ERROR   +63                     ;  : :   ERROR A PARITY ABORT HAS OCCURED
2739 011470  005037  172100                  clr     mer
2740 011474  005201                  80$:    INC     R1                      ;  : BUMP UP THE ADDRESS
2741 011476  020127  060000                  CMP     R1,#60000               ;  : IF WE HAVE PASSED THE PAGE BOUNDARY
2742 011502  103753                          BLO     75$                     ;  : : THEN
2743 011504  062737  000200  172344          ADD     #200,@#KIPAR2           ;  : :  POINT KPAR2 TO A NEW PAGE
2744 011512  012701  040000                  MOV     #40000,R1               ;  : :  SET THE VIRTUAL ADDRESS TO PAGE 2
2745 011516  023727  172344  020000          CMP     @#KIPAR2,#20000         ;  : ENDIF
2746 011524  001401                          BEQ     90$                     ;  DOUNTIL WE HAVE READ ALL ADDRESSES
2747 011526  000741                          BR      75$                     ;
2748 011530  005037  177572          90$:    CLR     @#SR0                   ;  DISABLE THE MMU
2749 011534  005037  172100                  CLR     @#MER                   ;
2750 011540  012637  000116                  MOV     (SP)+,@#116             ;RESTORE PRIORITY
2751 011544  012637  000114                  MOV     (SP)+,@#114             ;RESTORE VECTOR
2752 011550  000400                          BR      TST11           ;;GO TO THE NEXT TEST
2753 011552                          NXTST:
```

```
                                .SBTTL  TEST - LTC BIT 7
                                ;*****************************************************************
                                ; LTC BIT 7 TEST
                                ;
                                ; This test check for the existance of the LTC register and it
                                ; makes sure that the clock is ticking.
                                ;
                                ; BGNTST
                                ;  Set up timeout vector PC to TIMOUT
                                ;  Set up timeout vector PSW to 7
                                ;  Read the LTC CSR
                                ;  IF not timed OUT THEN
                                ;  : FOR a set amount of t'me
                                ;  :     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 tick'ng
                                ; ENDTST
                                ;
                                ; TIMOUT: Clean stack
                                ;         Error LTC register timed out
                                ;
                                ;    ---   -      - -  ----  ----   ----  ---   --  -         --
                                
                                                                        ;LTC BIT7 TEST
                                ;;*************************************************************
     011552  000004             TST11:  SCOPE
2785 011554  032777  010000  167356        BIT     #BIT12,@SWR     ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
2786 011562  001420                        BEQ     1000$           ;    THEN TYPE TEST TRACE
2787 011564  104401  011572                TYPE    ,30029$         ;;TYPE ASCIZ STRING
     011570  000415                        BR      30028$          ;;GET OVER THE ASCIZ
                                ;;30029$:       .ASCIZ  <15><12>/TEST 11 - LTC BIT7 TEST/
     011624                     30028$:
2788 011624                     1000$:
2789 011624  013737  000004  003016        MOV     @#4,SAVTIM              ; SAVE UNEXPECTED TIMEOUT
2790 011632  012737  011736  000004        MOV     #100$,@#4               ;  SET UP THE TIMEOUT TRAP
2791 011640  012737  000340  000006        MOV     #340,@#6                ;
2792 011646  005737  177546                TST     LKS                     ;  READ THE LKS REGISTER
2793 011652  005000                        CLR     R0                      ;  CLEAR THE HIGH COUNTER
2794 011654  005001                        CLR     R1                      ;  CLEAR THE LOW COUNTER
2795 011656  012703  000001                MOV     #1,R3                   ;
2796 011662  012702  177777                MOV     #177777,R2              ;  SET UP A COUNT
2797 011666  032737  000200  177546  1$:   BIT     #BIT7,LKS               ;  FOR AN AMOUNT OF TIME
2798 011674  001402                        BEQ     5$                      ;  : CHECK BIT 7 OF THE LKS
2799 011676  005200                        INC     R0                      ;  : IF IT IS A "1' THEN
2800 011700  000401                        BR      10$                     ;  :    BUMP UP R0
2801 011702  005201                5$:     INC     R1                      ;  : ELSE
2802 011704  077210                10$:    SOB     R2,1$                   ;  :    BUMP UP R1
2803 011706  005700                        TST     R0                      ;  ENDFOR
2804 011710  001402                        BEQ     20$                     ;  IF R0 = 0 OR
2805 011712  005701                        TST     R1                      ;     R1 = 0    THEN
2806 011714  001004                        BNE     25$                     ;     ERROR WITH BIT7 OF LTC
2807 011716  077315                20$:    SOB     R3,1$                   ;  TOLERATE ONE ERROR ONLY
```

```
011720  104053                                    ERROR   +53                     ; ENDTST
011722  005037  172100                            clr     mer
011726  013737  003016  000004  25$:              MOV     SAVTIM,@#4              ;  RESTORE UNEXPECTED TIMEOUT
011734  000412                                    BR      TST12             ;;GO TO THE NEXT TEST


                                ;
                                ; TIMEOUT ROUTINE
                                ;

011736  012737  011752  000004  100$:             mov     #102$,@#4
011744  104054                                    ERROR   +54                     ; REPORT ERROR
011746  005037  172100                            clr     mer
011752  022626                          102$:     CMP     (SP)+,(SP)+             ; CLEAN STACK
011754  013737  003016  000004                    MOV     SAVTIM,@#4              ;  RESTORE UNEXPECTED TIMEOUT



```

```
                                .SBTTL  TEST    LKS INTERRUPT PRIORITY
                                ;CHECK THAT LKS INTERRUPTS HAPPEN AT PRIORITY 5 CLEARING LKS<07>
                                ;AND DON'T HAPPEN AT PRIORITY 6.
                                ;ROUTINE TEST
                                ;IF UFD AND LKS IS DISABLED THEN
                                ;.      EXIT TEST
                                ;ENDIF
                                ;.      SET PRIORITY TO 5
                                ;.      CLEAR INTERRUPT_FLAG
                                ;.      LET LKS<06>=#1 (ENABLE INTERRUPTS)
                                ;.      SET COUNTER TO WAIT FOR 3 INTERRUPTS
                                ;.      REPEAT
                                ;.      .       DECREMENT COUNTER
                                ;.      UNTIL INTERRUPT FLAG EQ #3 OR COUNTER EQ #0
                                ;.      CLEAR LKS<06>
                                ;.      IF LKS<07> EQ #1 THEN
                                ;.      .       ERROR (WAS NOT CLEARED ON INTERRUPT)
                                ;.      ENDIF
                                ;.      IF COUNTER LT TIME_REQUIRED_FOR_3_INTERRUPTS FOR 800HZ
                                ;.      .       ERROR (INTERRUPTS NEVER GO LOW)
                                ;.      ENDIF
                                ;.      IF INTERRUPT_FLAG LT #3 THEN
                                ;.      .       ERROR (INTERRUPTS DON'T HAPPEN)
                                ;.      ENDIF
                                ;.      CLEAR INTERUPT_FLAG
                                ;.      WAIT FOR LKS<7>=1
                                ;.      LET LKS<7>=0
                                ;.      IF LKS<7> NE #0 THEN
                                ;.      .       ERROR (LKS<7> NOT CLEARED
                                ;.      ENDIF
                                ;.      SET PRIORITY TO 6
                                ;.      SET COUNTER TO 1 SLOW CLOCK INTERRUPT
                                ;.      SET LKS<06>
                                ;.      REPEAT
                                ;.      .       DECREMENT COUNTER
                                ;.      UNTIL COUNTER EQ #0 OR INTERRUPT_FLAG NE #0
                                ;.      IF INTERRUPT_FLAG NE #0 THEN
                                ;.      .       ERROR (INTERRUPT WAS AT WRONG PRIORITY)
                                ;.      ENDIF
                                ;.      RESTORE ORIGINAL PRIORITY
                                ;ENDROUTINE
                                ;
                                ;ROUTINE LINE_CLOCK_INTERUPT
                                ;.      INCREMENT INTERRUPT_FLAG
                                ;ENDROUTINE

                                ;;*****************************************************************
011762  000004                  TST12:  SCOPE
011764  032777  010000  167146          BIT     #BIT12,@SWR     ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
011772  001425                        BEQ     1000$           ;    THEN TYPE TEST TRACE
011774  104401  012002                TYPE    ,30031$         ;;TYPE ASCIZ STRING
012000  000422                        BR      30030$          ;;GET OVER THE ASCIZ
                                ;;30031$:       .ASCIZ  <15><12>/TEST 12 - LKS INTERRUPT PRIORITY/
012046                          30030$:
012046                          1000$:
012046  032737  000100  000052          BIT     #BIT06,@#52                     ;UFD MODE?
012054  001400                        BEQ     1$                              ;IF NOT, GO DO TEST
```

G6


```
                                        ;
                                        ; WAIT FOR 3 INTERRUPTS AND CHECK LKS<7> TO BE 0 AFTER INTERRUPT
                                        ;
012056  042737  000100  177546  1$:     BIC     #BIT06,LKS              ; FROM END OF TEST 42?? PROBLEM
012064  005037  002402                  CLR     LKSFL                   ;CLEAR INTERRUPT FLAG
012070  005037  002404                  clr     lkcnt
012074  013737  000100  002406          mov     @#100,savloc            ;save wht was in 100
012102  012737  027530  000100          mov     #clkcnt,@#100           ;MOV    #LKSINT,100                     ;POI
NT VECTOR TO ROUTINE
012110  012701  077777                  MOV     #77777,R1               ;COUNTER FOR SLOW CLOCK
012114  052737  000100  177546          BIS     #BIT06,LKS              ;SET INTERRUPT ENABLE BIT
012122  032737  000100  177546          BIT     #BIT06,LKS              ;BIT SET OK?
012130  001003                          BNE     2$                      ;IF YES, BRANCH
012132  104042                          ERROR   +42                     ;ERROR WRITING 1 TO LKS<6>
012134  005037  172100                  clr     mer
012140  106427  000240          2$:     MTPS    #240                    ;SET PRIORITY TO 5
012144  004737  030610          3$:     jsr     pc,delay
012150  022737  000003  002404          cmp     #3,lkcnt                ;CMP    #3,LKSFL                        ;3 I
NTERRUPTS HAPPENED?
012156  001372                          bne     3$                      ;BEQ    4$                              ;IF
YES, BRANCH
                                        ;
                                        ; DISABLE INTERRUPTS AND CHECK THAT PROPER CONDITIONS ARE MET
                                        ;
012160  106427  000340          4$:     MTPS    #340                    ;RAISE PRIORITY
012164  042737  000100  177546          BIC     #BIT06,LKS              ;DISABLE INTERRUPTS
012172  032737  000100  177546  6$:     BIT     #BIT06,LKS              ;LKS<6>-0?
012200  001403                          BEQ     7$                      ;IF YES, BRANCH
012202  104042                          ERROR   +42                     ;ERROR WRITING 0 TO LKS<6>
012204  005037  172100                  clr     mer
012210                          7$:
012210  022737  000003  002404  8$:     CMP     #3,lkcnt                ;DID 3 INTERRUPTS HAPPEN?
012216  001406                          BEQ     9$                      ;IF YES, BRANCH
012220  012737  000003  001124          MOV     #3,$GDDAT               ;3 INTERRUPTS EXPECTED
012226  104007                          ERROR   +7                      ;INTERRUPTS DON'T HAPPEN
012230  005037  172100                  clr     mer
                                        ;
                                        ; CHECK WHETHER INTERRUPTS HAPPEN AT PRIORITY 6
                                        ;
012234  005037  002404          9$:     CLR     lkcnt                   ;CLEAR INTERRUPT FLAG
012240  106427  000300                  MTPS    #300                    ;RAISE PRIORITY TO 6
012244  012701  077777                  MOV     #77777,R1               ;COUNTER FOR SLOW CLOCK
012250  052737  000100  177546          BIS     #BIT06,LKS              ;SET INTERRUPT ENABLE BIT
012256                          10$:
012256  005737  002404                  TST     lkcnt                   ;ANY INTERRUPTS?
012262  001002                          bne     11$                     ;IF YES, EXIT LOOP
012264  004737  030610                  jsr     pc,delay                ;SOB    R1,10$                          ;CON
TINUE WITH COUNT
012270  005737  002404          11$:    TST     lkcnt                   ;ANY INTERRUPTS?
012274  001406                          BEQ     12$                     ;IF NO, BRANCH
012276  012737  000006  001124          MOV     #6,$GDDAT               ;STORE PRIORITY FOR TYPE OUT
012304  104010                          ERROR   +10                     ;INTERUPTS HAPPEN AT WRONG PRIORITY
012306  005037  172100                  clr     mer
012312  106427  000340          12$:    MTPS    #340                    ;RESTORE PRIORITY
012316  042737  000100  177546          BIC     #BIT6,LKS               ;DISABLE CLOC  INTERRUPTS
012324  013737  002406  000100          mov     @#savloc,@#100
```

```
                                        .SBTTL  TEST    RESETTING LKS
                                        ;RESETTING LKS(*)
                                        ;THIS TEST WILL PROVE THAT RESET INSTRUCTION CLEARS LKS<06>.
                                        ;ROUTINE TEST
                                        ;IF UFD AND LKS IS DISABLED THEN
                                        ;.      EXIT TEST
                                        ;ENDIF
                                        ;.      POINT LKS VECTOR 100 TO ERROR_LKS_ILLEGAL INTERRUPT
                                        ;.      SYNCHRONIZE LKS BY WAITING FOR 3 PULSES
                                        ;.      LET LKS<06>=#1
                                        ;.      EXECUTE "RESET"
                                        ;.      IF LKS<6> NE #0 THEN
                                        ;.              ERROR
                                        ;.      ENDIF
                                        ;.      IF ILLEGAL_LINE_CLOCK INTERRUPT NE 0 THEN
                                        ;.              ERROR
                                        ;.      ENDIF
                                        ;ENDROUTINE
                                        ;
                                        ;ROUTINE ERROR_LKS_ILLEGAL_INTERRUPT
                                        ;.      FLAG ILLEGAL LINE_CLOCK INTERRUPT
                                        ;RETURN

                                        ;;*****************************************************************
 012332  000004                         TST13:  SCOPE
 012334  032777  010000  166576                 BIT     #BIT12,@SWR     ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
 012342  001423                                 BEQ     1000$           ;   THEN TYPE TEST TRACE
 012344  104401  012352                         TYPE    ,30033$         ;;TYPE ASCIZ STRING
 012350  000420                                 BR      30032$          ;;GET OVER THE ASCIZ
                                        ;;30033$:       .ASCIZ  <15><12>/TEST 13 - RESETTING LKS TEST/
 012412                                 30032$:
 012412                                 1000$:
 012412  005737  001206                         TST     $PASS                           ;FIRST PASS?
 012416  001053                                 BNE     TST14           ;;IF NOT FIRST PASS, EXIT TEST
 012420  032737  000100  000052                 BIT     #BIT06,@#52                     ;UFD MODE?
 012426  001400                                 BEQ     1$                              ;IF NOT, BRANCH
                                        ;
                                        ; SYNCHRONISE WITH LINE TIME CLOCK BY WAITING FOR 3 INTERRUPTS
                                        ;
 012430                                 1$:
 012430  013737  000100  002406                 mov     @#100,savloc
 012436  005037  002404                         clr     lkcnt
 012442  012737  027530  000100                 mov     #clkcnt,@#100
                                                                        ;MOV     #LKSINT,@#100                   ;SET
 UP INTERRUPT VECTOR
 012450  012737  000340  000102                 MOV     #340,@#102              ;AT PROIRITY 7
 012456  052737  000100  177546                 BIS     #BIT06,LKS              ;SET INTERRUPT ENABLE BIT
 012464  005037  002402                         CLR     LKSFL                   ;CLEAR INTERRUPTS FLAG
 012470  012702  077777                         MOV     #77777,R2               ;COUNTER TO WAIT FOR INTERRUPTS
 012474  106427  000240                         MTPS    #240                    ;LOWER PRIORITY TO 5
 012500  004737  030610                 2$:     jsr     pc,delay
 012504  022737  000003  002404                 cmp     #3,lkcnt                ;CMP     #3,LKSFL               ;3 I
NTERRUPTS HAPPENED?
 012512  001372                                 bne     2$                      ;BEQ     3$                     ;EXI
T LOOP, IF SO
                                                                        ;SOB     R2,2$                          ;OTH
ERWISE, KEEP WAITING
 012514  106427  000340                 3$:     MTPS    #340                    ;RAISE PRIORITY TO 7
 012520  000005                                 RESET                           ;EXECUTE RESET

 012522  013737  002406  000100         4$:     mov     savloc,@#100
```

TEST RESETTING LKS

```
2988 012530  032737  000100  177546          BIT     #BIT06,LKS                      ;INTERRUPT ENABLE BIT CLEARED?
2989 012536  001403                           BEQ     TST14           ;;IF SO, EXIT TEST
2990 012540  104011                           ERROR   +11                             ;RESET DOESN'T CLEAR LKS
2991 012542  005037  172100                   clr     mer
2992
```

```
                                        .SBTTL  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

                                        ;;*****************************************************************
     012546  000004                     TST14:  SCOPE
3009 012550  032777  010000  166362             BIT     @#BIT12,@SWR    ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
3010 012556  001426                             BEQ     1000$           ;    THEN TYPE TEST TRACE
3011 012560  104401  012566                     TYPE    ,30035$         ;;TYPE ASCIZ STRING
     012564  000423                             BR      30034$          ;;GET OVER THE ASCIZ
                                        ;;30035$:       .ASCIZ  <15><12>/TEST 14   MAINTENANCE REGISTER TEST/
     012634                             30034$:
3012 012634                             1000$:
3013 012634  032737  177000  177750             BIT     #177000,MAIREG                  ;UNUSED BITS ALL ZEROS?  ;MC001
3014                                                                                    ;CHANGED FROM 174000 FOR THE DA
3015 012642  001403                             BEQ     1$                              ;IF OK, BRANCH
3016 012644  104043                             ERROR   +43                             ;MAINTENANCE REGISTER ERROR
3017 012646  005037  172100                     clr     mer
3018 012652  032737  000010  177750  1$:       BIT     @#BIT3,MAIREG                   ;IF JUMPER IS IN
3019 012660  001010                             BNE     2$                              ;; THEN
3020 012662  042737  000010  177750             BIC     @#BIT3,MAIREG                   ;; SET HALT TO ODT
3021 012670  022737  000105  177750             CMP     #105,MAIREG                     ;; MAINTENANCE REGISTER SHOULD BE SET UP TO
105
3022                                                                                    ;; CHANGED FROM <5,2> FOR THE DA
3023 012676  001401                             BEQ     2$                              ;IF SO, BRANCH
3024 012700  104043                             ERROR   +43                             ;MAINTENANCE REGISTER ERROR
3025 012702  005037  172100             2$:     clr     mer
3026
```

```
                                        .SBTTL  TEST    SERIAL LINE UNIT REGISTERS
                                        ;SERIAL LINE UNIT TEST(*)
                                        ;BCR<2 0> WILL BE READ TO FIND OUT BAUD RATE. SLU WILL BE PROG
                                        ;RAMMED TO CHECK THE INTERRUPT LEVELS BY SETTING BIT<06> IN
                                        ;RCSR AND XMIT. LOOP BACK CAPABILITIES WILL BE TESTED BY SETTING
                                        ;TO 1 XCSR<02>.   THE LINE CLOCK INTERRUPT SUBROUTINE WILL BE
                                        ;USED TO RETURN TO THE EXECUTION OF THE DIAGNOSTICS, IF THE
                                        ;PROGRAM HANGS IN THE LOOP BACK MODE.
                                        ;ROUTINE TEST
                                        ;IF UFD AND CONSOLE NOT PRESENT
                                        ;.      GO TO TEST 22
                                        ;ENDIF
                                        ;.      IF BCR<07> EQ #0 THEN
                                        ;.      .       READ BCR<2 0> TO GET BAUD RATE
                                        ;.      ENDIF
                                        ;.      LET 4=ADDRESS_OF_TIMEOUT_ROUTINE
                                        ;.      DO FOR RCSR,XCSR,RBUF,XBUF
                                        ;.      .       READ XRCSR,XCSR,RBUF,XBUF
                                        ;.      .       IF TIMEOUT_FLAG NE #0 THEN
                                        ;.      .       .       ERROR
                                        ;.      .       ENDIF
                                        ;.      ENDDO
                                        ;ENDROUTINE
                                        ;
                                        ;ROUTINE TIMEOUT
                                        ;.      LET TIMEOUT FLAG=#1
                                        ;ENDROUTINE

                                        ;;*****************************************************************
     012706  000004                     TST15:  SCOPE
3057 012710  032777  010000  166222             BIT     #BIT12,@SWR     ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
3058 012716  001426                             BEQ     1000$           ;    THEN TYPE TEST TRACE
3059 012720  104401  012726                     TYPE    ,30037$         ;;TYPE ASCIZ STRING
     012724  000423                             BR      30036$          ;;GET OVER THE ASCIZ
                                        ;;30037$:       .ASCIZ  <15><12>/TEST 15 - SLU REGISTER ACCESS TEST/
     012774                             30036$:
3060 012774                             1000$:
3061 012774  032737  000100  000052             BIT     #BIT06,@#52                     ;UFD MODE?
3062 013002  001402                             BEQ     1$                              ;IF NOT, GO DO THE TEST
3063 013004  000137  016714                     JMP     SLEND                           ;IF TRUE, SKIP ALL SLU TESTS
3064                                    ;
3065                                    ; TRY TO ACCESS SLU REGISTERS
3066                                    ;
3067 013010  013701  000004             1$:     MOV     ERRVEC,R1                       ;SAVE TIMEOUT VECTOR
3068 013014  012737  013050  000004             MOV     #3$,ERRVEC                      ;POINT NEW ONE TO PROGRAM AREA
3069 013022  012737  000340  000006             MOV     #340,ERRVEC+2                   ;AT PRIORITY 7
3070 013030  012702  177560                     MOV     #RCSR,R2                        ;START ACCESSING WITH RCSR
3071 013034  012703  177566                     MOV     #XBUF,R3                        ;
3072 013040  012704  000002                     MOV     #2,R4                           ;
3073 013044  005712                     2$:     TST     (R2)                            ;ACCESS SLU REGISTER
3074 013046  000405                             BR      4$                              ;IF NO TIMEOUT, CONTINUE
3075 013050  010237  001126             3$:     MOV     R2,$BDDAT                       ;STORE ADDRESS THAT TIMED OUT
3076 013054  104012                             ERROR   +12                             ;TIMEOUT ACCESSING SLU REGISTER
3077 013056  005037  172100                     clr     mer
3078 013062  020322                     4$:     CMP     R3,(R2)+                        ;LAST REGISTER ACCESSED?
3079 013064  103767                             BLO     2$                              ;IF NOT, BRANCH
3080 013066  012702  176500                     MOV     #RCSR1,R2                       ;GET POINTERS TO SLU 1
```

TEST SERIAL LINE UNIT REGISTERS

```
3081 013072  012703  176506              MOV     #XBUF1,R3               ;GET LAST ADDRESS ON SLU 1
3082 013076  077416                      SOB     R4,2$                   ;
3083 013100  010137  000004              MOV     R1,ERRVEC               ;RESTORE TIMEOUT VECTOR
3084
```

M6


```
                                        .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

                                        ;;*****************************************************************
013104  000004                          TST16:  SCOPE
013106  122737  000001  001220                  CMPB    #APTENV,$ENV            ;ARE WE IN APT MODE?
013114  001006                                  BNE     1002$                   ;IF NOT: DO THIS TEST
013116  005737  001206                          TST     $PASS                   ;FIRST PASS??
013122  001403                                  BEQ     1002$                   ;IF YES, DO IT
013124  012704  000002                          MOV     #2,R4                   ; LOOP COUNT OF 1
013130  000455                                  BR      4$                      ; GO SET UP POINTERS TO SLU 1
013132                                  1002$:
013132  032777  010000  166000                  BIT     #BIT12,@SWR     ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
013140  001423                                  BEQ     1000$           ;    THEN TYPE TEST TRACE
013142  104401  013150                          TYPE    ,30039$         ;;TYPE ASCIZ STRING
013146  000416                                  BR      30038$          ;;GET OVER THE ASCIZ
                                        ;;30039$:       .ASCIZ  <15><12>/TEST 16   XCSR BIT 7 TEST/
013204                                  30038$:
                                                                        ;MOV    #100000,R1
013204  004737  030610                          jsr     pc,delay                ;1001$: SOB     R1,1001$        ; WAIT FOR L
AST CHARACTER
013210                                  1000$:
                                                                        ;MOV    #1000,R1                        ;COU
NTER FOR ABOUT 200MICROSEC.
013210  012702  177564                          MOV     #XCSR,R2                ;
013214  012703  177566                          MOV     #XBUF,R3                ;
013220  012704  000002                          MOV     #2,R4                   ;
013224  105712                          1$:     TSTB    (R2)                    ;XCSR<7> READY 1?
013226  100402                                  BMI     2$                      ;IF SO, EXIT WAIT LOOP
                                                                        ;SOB    R1,1$                           ;IF
NOT 1, CONTINUE WAITING
013230  004737  030610                          jsr     pc,delay
                                        ;
013234  105712                          2$:     TSTB    (R2)                    ;XCSR<7>=1?
013236  100403                                  BMI     3$                      ;IF YES, BRANCH
013240  104013                                  ERROR   +13                     ;XCSR<7> DOES NOT BECOME 1
013242  005037  172100                          clr     mer
013246  012713  000000                  3$:     MOV     #NULL,(R3)              ;TRY TO TRANSMIT NULL CHARACTER
013252  105712                                  TSTB    (R2)                    ;XCSR<7>=0
013254  100003                                  BPL     4$                      ;
013256  104013                                  ERROR   +13                     ;XMIT READY DIDN'T GO LOW
013260  005037  172100                          clr     mer
013264  012702  176504                  4$:     MOV     #XCSR1,R2               ;
013270  012703  176506                          MOV     #XBUF1,R3               ;
013274  012701  001000                          MOV     #1000,R1                ;
```

N6

Q 0078

TEST - XCSR BIT 7

```
3139 013300  077427                    SOB     R4,1$                   ;
3140
```

```
                                        .SBTTL  TEST    RCSR BIT 7 AND XCSR BIT 2
                                        ;CHECK THAT RCSR<07> CAN BE 0 AND 1 AND THAT XCSR<02> WORKS PROPERLY.
                                        ;
                                        ;RCSR   <07>    RECEIVER DONE
                                        ;XCSR   <02>    MAINTENANCE
                                        ;
                                        ;ROUTINE TEST
                                        ;.(CHECK RCSR<07> AND XCSR<07>)
                                        ;.      WAIT FOR XCSR<07>=#1
                                        ;.      LET XCSR<02>=#1 (LOOP BACK MODE)
                                        ;.      LET XBUF=#125
                                        ;.      WAIT FOR RCSR<07>=#1 NO MORE THAN 200MSEC
                                        ;.      IF RCSR<07> NE #1 THEN
                                        ;.      .       ERROR (RCSR<07> DOES NOT BECOME 1 OR XCSR<02>DOES NOT
                                        ;.      .               WORK)
                                        ;.      ENDIF
                                        ;.      IF RBUF NE #125 THEN
                                        ;.      .       ERROR
                                        ;.      ENDIF
                                        ;.      IF RCSR<07> NE #0 THEN
                                        ;.      .       ERROR (RCSR<07>DOES NOT GO LOW)
                                        ;.      ENDIF
                                        ;.      LET XCSR<02>=#0
                                        ;ENDROUITNE

                                        ;;*****************************************************************
      013302  000004                    TST17:  SCOPE
3168  013304  122737  000001  001220            CMPB    #APTENV,$ENV            ;ARE WE IN APT MODE?
3169  013312  001006                    BNE     1002$                   ;IF NOT: DO THIS TEST
3170  013314  005737  001206            TST     $PASS                   ;FIRST PASS??
3171  013320  001403                    BEQ     1002$                   ;IF YES, DO IT
3172  013322  012703  000002            MOV     #2,R3                   ; LOOP COUNT OF 1
3173  013326  000555                    BR      12$                     ; GO SET UP POINTERS TO SLU 1
3174  013330  032777  010000  165602    1002$:  BIT     #BIT12,@SWR     ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
3175  013336  001431                    BEQ     1000$           ;       THEN TYPE TEST TRACE
3176  013340  104401  013346            TYPE    ,30041$         ;;TYPE ASCIZ STRING
      013344  000426                    BR      30040$          ;;GET OVER THE ASCIZ
                                        ;;30041$:       .ASCIZ  <15><12>/TEST 17   RCSR BIT 7 AND XCSR BIT 2 TEST/
      013422                            30040$:
3177
3178                                                                    ;MOV    #100000,R1
                                                                        ;1001$: SOB     R1,1001$        ; WAIT FOR L
AST CHARACTER
3179  013422                            1000$:
3180
3181  013422  004737  030610                    jsr     pc,delay
3182
3183  013426  012701  000013                    MOV     #13,R1                  ;COUNTER FOR ABOUT 200MICROSEC.
3184  013432  012702  177560                    MOV     #RCSR,R2                ;
3185  013436  012703  000002                    MOV     #2,R3
3186  013442  004737  030610            1$:     jsr     pc,delay
3187  013446  105762  000004                    TSTB    4(R2)                   ;XCSR<7> READY 1?
3188  013452  100410                            bmi     2$                      ;BPL    1$                      ;IF
NOT 1, CONTINUE WAITING
3189  013454  004737  030610                    jsr     pc,delay
3190  013460  105762  000004                    tstb    4(r2)
3191  013464  100403                            bmi     2$
3192  013466  104076                            error   +76
3193  013470  005037  172100                    clr     mer
3194  013474  052762  000004  000004  2$:       BIS     #BIT02,4(R2)            ;SET LOOP BACK MODE
```

TEST RCSR BIT 7 AND XCSR BIT 2

```
3195 013502  032762  000004  000004          BIT     #BIT02,4(R2)                    ;GOT SET OK?
3196 013510  001006                          BNE     3$                              ;IF YES, BRANCH
3197 013512  005062  000004                  CLR     4(R2)                           ;RESET TO PRINT ERROR
3198 013516  104034                          ERROR   +34                             ;XCSR<2> DOES NOT BECOME '
3199 013520  005037  172100                  clr     mer
3200 013524  000465                          BR      TST20           ;;EXIT TEST
3201                                 ;
3202                                 ; STALL FOR A WHILE IN CASE XCSR<2> CAUSES RCSR<7> TO BE 1
3203                                 ;
3204 013526                          3$:                                             ;MOV     #60000,R1                      ;STA
LL IN CASE XCSR<2> SETS READY
3205 013526  105712                  4$:     TSTB    (R2)                            ;IF RECEIVER READY SET?
3206 013530  100403                          BMI     5$                              ;IF SET, BRANCH
3207 013532  004737  030610                  jsr     pc,delay                        ;SOB     R1,4$                          ;OTH
ERWISE, STAY FOR A WHILE
3208
3209
3210 013536  000402                          BR      6$                              ;IF NOT READY, BRANCH
3211 013540  005762  000002          5$:     TST     2(R2)                           ;READ RBUF
3212                                 ;
3213                                 ; TRANSMIT XON AND CHECK RCSR<7>
3214                                 ;
3215 013544  012762  000021  000006  6$:     MOV     #21,6(R2)                       ;TRANSMIT A CHARACTER
3216                                                                                 ;MOV     #60000,R1                      ;COU
NTER TO WAIT
3217 013552  105712                  7$:     TSTB    (R2)                            ;RCSR<7> READY 1?
3218 013554  100402                          BMI     8$                              ;IF YES, EXIT WAIT LOOP
3219 013556  004737  030610                  jsr     pc,delay
3220                                                                                 ;SOB     R1,7$                          ;OTH
ERWISE, CONTINUE WAITING
3221
3222 013562  105712                  8$:     TSTB    (R2)                            ;RCSR<7>=1?
3223 013564  100405                          BMI     9$                              ;IF YES, BRANCH
3224 013566  005062  000004                  CLR     4(R2)                           ;RESET XCSR<2>
3225 013572  104014                          ERROR   +14                             ;RECEIVER READY DIDN'T COME UP
3226 013574  005037  172100                  clr     mer
3227 013600  016237  000002  001126  9$:     MOV     2(R2),$BDDAT                    ;STORE RECEIVED DATA
3228 013606  022737  000021  001126          CMP     #21,$BDDAT                      ;DATA RECEIVED OK?
3229 013614  001410                          BEQ     10$                             ;IF YES, BRANCH
3230 013616  012737  000021  001124          MOV     #21,$GDDAT
3231 013624  005062  000004                  CLR     4(R2)                           ;RESET TO ENABLE SLU
3232 013630  104015                          ERROR   +15                             ;WRONG CHARACTER RECEIVED
3233 013632  005037  172100                  clr     mer
3234 013636  105712                  10$:    TSTB    (R2)                            ;RCSR<7>=0?
3235 013640  100005                          BPL     11$                             ;IF ZERO, BRANCH
3236 013642  005062  000004                  CLR     4(R2)                           ;RESET TO ENABLE SLU
3237 013646  104016                          ERROR   +16                             ;RCSR<07><>0 AFTER READING RBUF
3238 013650  005037  172100                  clr     mer
3239 013654  042762  000004  000004  11$:    BIC     #BIT02,4(R2)                    ;DISABLE LOOP BACK MODE
3240 013662  012702  176500          12$:    MOV     #RCSR1,R2                       ;POINT TO SLU 1
3241 013666  012701  000013                  MOV     #13,R1                          ;SET UP COUNTER
3242                                                                                 ;SOB     R3,1$                                  ;
3243 013672  005303                          dec     r3
3244 013674  005703                          tst     r3
3245 013676  001261                          bne     1$
3246
```

```
                                        .SBTTL  TEST    RESET AND XCSR<2!0>
                                        ;CHECK THAT RESET CLEARS XCSR<0!2>.
                                        ;ROUTINE TEST
                                        ;.(CHECK RCSR<07> AND XCSR<07> AND RESET)
                                        ;.      LET XCSR<02,00>=#1 (LOOP BACK MODE)
                                        ;.      EXECUTE "RESET"
                                        ;.      IF XCSR<02!00> NE #0 THEN
                                        ;.              ERROR
                                        ;.      ENDIF
                                        ;.      LET XCSR<02>=#0
                                        ;ENDROUTINE
                                        ;;*****************************************************************
 013700  000004                         TST20:  SCOPE
013702  122737  000001  001220                  CMPB    #APTENV,$ENV            ;ARE WE IN APT MODE?
013710  001006                                  BNE     1001$                   ;IF NOT: DO THIS TEST
013712  005737  001206                          TST     $PASS                   ;FIRST PASS??
013716  001403                                  BEQ     1001$                   ;IF YES, DO IT
013720  012703  000002                          MOV     #2,R3                   ; LOOP COUNT OF 1
013724  000451                                  BR      10$                     ; GO SET UP POINTERS TO SLU 1
013726  032777  010000  165204  1001$:  BIT     #BIT12,@SWR     ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
013734  001426                                  BEQ     1000$           ;    THEN TYPE TEST TRACE
013736  104401  013744                          TYPE    ,30043$         ;;TYPE ASCIZ STRING
 013742  000423                                BR      30042$          ;;GET OVER THE ASCIZ
                                        ;;30043'        .ASCIZ  <15><12>/TEST 20 - RESET AND XCSR<0!2> TEST/
 014012                                 30042$:
014012                                  1000$:
014012  012702  177564                          MOV     #XCSR,R2                                ;
014016  012703  000002                          MOV     #2,R3                           ;
014022  052712  000005                  1$:     BIS     #BIT02!BIT00,(R2)               ;LOOP BACK MODE
                                        ;
                                        ; EXECUTE RESET AND VALIDATE THAT XCSR<7,2> BECOMES <1,0>
                                        ;
014026  000005                                  RESET                                   ;EXECUTE RESET
014030  032712  000005                          BIT     #BIT02!BIT00,(R2)               ;XCSR<2,0> CLEAR?
014034  001405                                  BEQ     10$
014036  042712  000005                          BIC     #BIT02!BIT00,(R2)               ;CLEAR THE BITS SO WE CAN REPORT
014042  104022                                  ERROR   +22                             ;XCSR<2,0> NOT CLEARED ON RESET
014044  005037  172100                          clr     mer
014050  012702  176504                  10$:    MOV     #XCSR1,R2                       ;
014054  077316                                  SOB     R3,1$                           ;
```

```
3287                                    .SBTTL  TEST    RESET AND INTERRUPT ENABLE BITS
3288                                    ;CHECK THAT INTERRUPTS DON T HAPPEN AT PRIORITY 4 AND THAT RESET
3289                                    ;CLEARS XCSR<06> AND RCSR<06>.
3290                                    ;
3291                                    ;RCSR   <06>    RECEIVER INTERRUPT ENABLE
3292                                    ;XCSR   <06>    TRANSMITTER INTERRUPT ENABLE
3293                                    ;
3294                                    ;ROUTINE TEST
3295                                    ;.      LET 60-@ADDRESS_OF ILLEGAL INTERRUPT_XRCSR
3296                                    ;.      LET 64=@ADDRESS OF_ILLEGAL_INTERRUPT XRCSR
3297                                    ;.      SET PRIORITY TO 4
3298                                    ;.      LET XCSR<02>=#1 (LOOPBACK MODE)
3299                                    ;.      LET XCSR<06>=#1 (ENABLE TRANSMIT INTERRUPTS)
3300                                    ;.      LET RCSR<06>=#1 (ENABLE RECEIVE INTERRUPTS)
3301                                    ;.      WAIT FOR XCSR<07>=#1 (READY TO TRANSMIT)
3302                                    ;.      LET XBUF=@NULL (SEND A CHARACTER)
3303                                    ;.      WAIT FOR ILLEGAL INTERRUPTS (ABOUT 200MSEC)
3304                                    ;.      EXECUTE "RESET"
3305                                    ;.      IF XCSR<06> NE #0 OR RCSR<06> NE #0 OR XRCSR NE #0 THEN
3306                                    ;.          ERROR
3307                                    ;.      ENDIF
3308                                    ;.      RESTORE PRIORITY TO NORMAL
3309                                    ;ENDROUTINE
3310                                    ;
3311                                    ;ROUTINE ILLEGAL_INTERRUPT XRCSR
3312                                    ;.      INCREMENT XRCSR
3313                                    ;ENDROUTINE
3314
3315                                    ;;*****************************************************************
     014056  000004                     TST21:  SCOPE
3316 014060  122737  000001  001220             CMPB    @APTENV,$ENV            ;ARE WE IN APT MODE?
3317 014066  001015                             BNE     1001$                   ;IF NOT: DO THIS TEST
3318 014070  005737  001206                     TST     $PASS                   ;FIRST PASS??
3319 014074  001412                             BEQ     1001$                   ;IF YES, DO IT
3320 014076  012705  000002                     MOV     #2,R5                   ; LOOP COUNT OF 1
3321 014102  013737  000004  003016             MOV     @#4,SAVTIM              ; STORE UNEXPECTED TIMEOUT
3322 014110  012737  015000  000004             MOV     #100$,@#4               ; SET UP TIMEOUT
3323 014116  000137  014642                     JMP     20$                     ; GO SET UP POINTERS TO SLU 1
3324 014122  032777  010000  165010  1001$:     BIT     #BIT12,@SWR     ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
3325 014130  001433                             BEQ     1000$           ;    THEN TYPE TEST TRACE
3326 014132  104401  014140                     TYPE    ,30045$         ;;TYPE ASCIZ STRING
     014136  000430                             BR      30044$          ;;GET OVER THE ASCIZ
                                        ;;30045$:       .ASCIZ  <15><12>/TEST 21 - SLU RESET AND INTERRUPT ENABLE TEST/
     014220                             30044$:
3327 014220                             1000$:
3328 014220  005037  002360                     clr     tcount
3329 014224  005037  002356                     clr     rcount
3330 014230  012737  015000  000004             MOV     #100$,@#4               ;
3331 014236  012701  000060                     MOV     #60,R1                  ; R1 POINTS TO SLU 0 INTERRUPT VECTOR
3332 014242  012704  177560                     MOV     #RCSR,R4                ; R4 POINTS TO SLU 0 REGISTERS
3333 014246  012705  000002                     MOV     #2,R5                   ; R5 IS THE LOOP COUNT
3334
3335                                    ; CHECK THAT INTERRUPTS ENABLE BITS FOR RECEIVER AND TRASMITTER OF SLU
3336                                    ; ARE CLEARED BY RESET
3337                                    ;
3338 014252  005037  002360             1$:     clr     tcount
3339 014256  005037  002356                     clr     rcount
```

F7

TEST RESET AND INTERRUPT ENABLE BITS

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

TEST RESET AND INTERRUPT ENABLE BITS

```
   3397 014532  012702  000340                    MOV     #340,R2             ;SET PRIORITY TO 7
   3398 014536  000402                            BR      13$                 ;GO WAIT IN CASE OF INTERRUPTS
   3399 014540  162702  000040          12$:      SUB     #40,R2              ;LOWER PRIORITY LEVEL
   3400 014544  052764  000004  000004  13$:      BIS     #BIT02,4(R4)        ;SET LOOP BACK MODE
   3401 014552  106402                            MTPS    R2                  ;SET PRIORITY
   3402 014554  004737  030610                    JSR     PC,DELAY            ;TIMER ROUTINE                  ;MC
   3403 014560  005737  002356                    tst     rcount              ;       MOV     #140,R3                 ;TIME DELAY
   3404 014564  001406                            beq     16$                 ;14$:   SOB     R3,14$                  ;WAIT FOR IN
TERRUPTS
   3405                                                                       ;BR     16$                     ;IF INTERRUPTS DIDN'
T HAPPENED, BRANCH
   3406 014566  042764  000004  000004  15$:      BIC     #BIT02,4(R4)        ;CLEAR LOOP BACK MODE
   3407 014574  104021                            ERROR   +21                 ;INTERRUPTS HAPPEN AT WRONG PRIORITY
   3408 014576  005037  172100                    clr     mer
   3409                                                                       ;CMP    (SP)+,(SP)+             ;CLEAN UP THE STACK
   3410 014602  022702  000200          16$:      CMP     #200,R2             ;AT PRIORITY 4?
   3411 014606  001354                            BNE     12$                 ;IF NOT LAST ONE, CONTINUE
   3412                                           ;
   3413                                           ; CLEAN UP BEFORE NEXT TEST
   3414                                           ;
   3415 014610  106427  000340                    MTPS    #340                ;RESTORE PRIORITY 7
   3416 014614  042714  000100                    BIC     #BIT06,(R4)         ;CLEAR INTERRUPT ENABLE BIT
   3417                                                                       ;MOV    #400,R2                 ;STALL DELAY
   3418 014620  105714                  17$:      TSTB    (R4)                ;RECEIVE READY?
   3419 014622  100402                            BMI     18$                 ;STOP WAITING, IF SO
   3420 014624  004737  030610                    jsr     pc,delay            ;SOB    R2,17$                  ;OTHERWISE, STAY IN
THE LOOP
   3421
   3422 014630  005764  000002          18$:      TST     2(R4)               ;READ CHARACTER TRANSMITTED
   3423 014634  042764  000004  000004            BIC     #BIT02,4(R4)        ;CLEAR LOOP BACK MODE
   3424 014642  012701  000300          20$:      MOV     #300,R1             ; POINT TO SLU #1 VECTOR
   3425 014646  012704  176500                    MOV     #RCSR1,R4           ; POINT TO SLU #1 REGISTER
   3426 014652  005305                            DEC     R5                  ;
   3427 014654  001402                            BEQ     19$                 ;
   3428 014656  000137  014252                    JMP     1$                  ;
   3429 014662                          19$:
   3430
   3431
   3432
   3433                                           ;
   3434                                           ; loop back connector test
   3435                                           ;
   3436
   3437
   3438 014662  012702  000002                    mov     #2,r2               ;counter
   3439
   3440 014666  032702  000001          110$:     bit     #1,r2               ; odd or even ??
   3441 014672  001005                            bne     112$                ;nope even
   3442 014674  022737  000001  002362            cmp     #1,lopbak           ;yep odd.  connector installed ???
   3443 014702  001031                            bne     120$                ;no it's not
   3444 014704  000403                            br      114$                ;
   3445
   3446 014706  052737  000004  176504  112$:     bis     #bit02,xcsr1        ;set the maintenance bit in the
   3447                                                                       ;csr to do the internal loop back
   3448
   3449 014714  012703  003004          114$:     mov     #tchar,r3           ;pointer of .byte : -1, 21, -1, 0
   3450 014720  004737  030610                    jsr     pc,delay            ;allow last character from terminal
   3451 014724  113700  176502                    movb    @#rbuf1,r0          ;suck any extra character
   3452
   3453 014730  112300                  116$:     movb    (r3)+,r0            ;put the first character in r0
```

TEST RESET AND INTERRUPT ENABLE BITS

```
 3454 014732  001415                        beq     120$                    ;done with this pass of th's test
 3455 014734  110037  176506                movb    r0,xbuf1                ;transmit
 3456 014740  004737  030610                jsr     pc,delay                ;allow some time
 3457 014744  105737  176500                tstb    @#rcsr1                 ;receiver ready ??
 3458 014750  100401                        bmi     118$
 3459 014752  104077                        error   +77                     ;no character rece'ved
 3460
 3461 014754  123737  176502  176506  118$: cmpb    @#rbuf1,@#xbuf1
 3462 014762  001762                        beq     116$
 3463 014764  104100                        error   +100                    ;wrong character received
 3464
 3465 014766                          120$:
 3466 014766  077241                        sob     r2,110$
 3467
 3468 014770                          500$:
 3469 014770  013737  003016  000004        MOV     SAVTIM,@#4              ;RESTORE UNEXPECTED TIMEOUT
 3470 014776  000440                        BR      TST22           ;;GO TO THE NEXT TEST
 3471
 3472                                 ;
 3473                                 ; DEBUG PURPOSES
 3474                                 ;
 3475
 3476 015000  106427  000340        100$:  MTPS    #340                    ;RESTORE PRIORITY 7
 3477 015004  042714  000100               BIC     #BIT06,(R4)             ;CLEAR INTERRUPT ENABLE BIT
 3478                                                                        ;MOV     #400,R2                ;STALL DELAY
 3479 015010  105714                 170$:  TSTB    (R4)                    ;RECEIVE READY?
 3480 015012  100403                        BMI     180$                    ;STOP WAITING, IF SO
 3481 015014  004737  030610                jsr     pc,delay                ;SOB     R2,170$                ;OTHERWISE, STAY IN
THE LOOP
 3482 015020  105714                        tstb    (r4)
 3483 015022  005764  000002        180$:  TST     2(R4)                   ;READ CHARACTER TRANSMITTED
 3484 015026  042764  000004  000004       BIC     #BIT02,4(R4)            ;CLEAR LOOP BACK MODE
 3485 015034  000000                        HALT
 3486
 3487
 3488                                 ;
 3489                                 ;INTERRUPT SERVICE ROUTINES
 3490                                 ;
 3491
 3492 015036  005737  002360        200$:  tst     TCOUNT                                  ;INCREMENT TRANS. COUNTER ;MC
 3493 015042  001003                        bne     61$
 3494 015044  042764  000100  000004       bic     #bit6,4(r4)
 3495 015052  005237  002360        61$:   inc     tcount
 3496 015056  000002                        RTI                                     ;RETURN FROM XMIT INTERRUPT ;MC
 3497
 3498 015060  005737  002356        202$:  tst     RCOUNT                          ;INCREMENT RECEIVER COUNTER;MC
 3499 015064  001002                        bne     81$
 3500 015066  042714  000100               bic     #bit6,(r4)
 3501 015072  005237  002356        81$:   inc     rcount
 3502 015076  000002                        RTI                                     ;RETURN FROM RECEIVER INTERRUP
 3503
 3504
 3505
```

```
                                .SBTTL  TEST    INTERRUPT PRIORITY FOR SLU
                                ;CHECK THAT INTERRUPTS HAPPEN AT PRIORITY 3 AND THAT THEY CLEAR
                                ;RCSR<06> AND XCSR<06>.
                                ;
                                ;ROUTINE TEST
                                ;.      LET 60=#ADDRESS OF LEGAL RINTERRUPT
                                ;.      LET 64=#ADDRESS OF LEGAL XINTERRUPT
                                ;.      LET XCSR<02>=#1
                                ;.      SET PRIORITY TO #3
                                ;.      WAIT FOR XINTERRUPT=#3
                                ;.      IF XCSR<07> EQ #1 THEN
                                ;.              ERROR
                                ;.      ENDIF
                                ;.      WAIT FOR RINTERRUPT=#3
                                ;.      IF RCSR<07> EQ #0 THEN
                                ;.              ERROR
                                ;.      ENDIF
                                ;.      LET XCSR<02>=#0
                                ;.      SET PRIORITY TO NORMAL
                                ;ENDROUTINE
                                ;
                                ;ROUTINE LEGAL_XINTERRUPT
                                ;.      LET XBUF=#CHARACTER
                                ;.      INCREMENT XINTERRUPT
                                ;ENDROUTINE
                                ;
                                ;ROUTINE LEGAL_RINTERRUPT
                                ;.      READ RCSR
                                ;.      INCREMENT RINTERRUPT
                                ;ENDROUTINE

                                ;;*****************************************************************
 015100  000004                 TST22:  SCOPE
 015102  122737  000001  001220         CMPB    #APTENV,$ENV            ;ARE WE IN APT MODE?
 015110  001007                         BNE     1002$                   ;IF NOT: DO THIS TEST
 015112  005737  001206                 TST     $PASS                   ;FIRST PASS??
 015116  001404                         BEQ     1002$                   ;IF YES, DO IT
 015120  012705  000002                 MOV     #2,R5                   ; LOOP COUNT OF 1
 015124  000137  015454                 JMP     20$                     ; GO SET UP POINTERS TO SLU 1
 015130  032777  010000  164002 1002$:  BIT     #BIT12,@SWR     ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
 015136  001431                         BEQ     1000$           ;    THEN TYPE TEST TRACE
 015140  104401  015146                 TYPE    ,30047$         ;;TYPE ASCIZ STRING
 015144  000424                         BR      30046$          ;;GET OVER THE ASCIZ
                                ;;30047$:       .ASCIZ  <15><12>/TEST 22   SLU INTERRUPT PRIORITY TEST/
 015216                         30046$:
 015216  004737  030610                 jsr     pc,delay                                        ;MOV    #100000,R1
                                                                                ;1001$: SOB     R1,1001$        ; WAIT FOR L
AST CHARACTER
 015222                         1000$:


                                ;
                                ; GET READY FOR INTERRUPTS
                                ;
 015222                         100$:
 015222  012701  000060                 MOV     #60,R1                  ;
 015226  012704  177560                 MOV     #RCSR,R4                ;
 015232  012705  000002                 MOV     #2,R5                   ;
```

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

TEST INTERRUPT PRIORITY FOR SLU

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

```
                                .SBTTL  TEST    BREAK CONDITION
                                ;
                                ; SLU #1 IS TESTED SINCE SLU #0 IS BEING USED AS THE CONSOLE
                                ;
                                ;CHECK THAT SENDING BREAK CAUSES FRAMING ERROR.
                                ;
                                ;RCSR    <15>    ERROR
                                ;        <13>    FRAMING ERROR
                                ;        <11>    RECEIVED BREAK
                                ;
                                ;XCSR    <00>    TRANSMIT BREAK
                                ;
                                ;ROUTINE TEST
                                ;.       LET XCSR<02>=#1
                                ;.       LET XCSR<00>=#1
                                ;.       WAIT FOR RCSR<07>=#1
                                ;.       IF RBUF<15!13!11> NE #1 THEN
                                ;.               ERROR (ERROR, FRAMING ERROR, RECEIVE BREAK NE 1)
                                ;.       ENDIF
                                ;.       LET XCSR<00>=#0
                                ;.       IF XCSR<00> NE #0 THEN
                                ;.               ERROR (XCSR<00> DOES NOT GO LOW)
                                ;.       ENDIF
                                ;.       WAIT FOR XCSR<07>=#1
                                ;.       LET XBUF=#NULL (SEND NULL CHARACTER TO SEE ERROR CLEARED)
                                ;.       WAIT FOR RCSR<07>=#1
                                ;.       IF RBUF<15!13!11> NE #0 THEN
                                ;.               ERROR
                                ;.       ENDIF
                                ;.       LET XCSR<00>=#1
                                ;.       EXECUTE "RESET"
                                ;.       IF XCSR<00> NE #0 THEN
                                ;.               ERROR
                                ;.       ENDIF
                                ;.       LET XCSR<02>=#0
                                ;ENDROUTINE
                                ;;*****************************************************************
     015654  000004             TST23:  SCOPE
3710 015656  032777  010000  163254        BIT     #BIT12,@SWR     ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
3711 015664  001426                        BEQ     1000$           ;    THEN TYPE TEST TRACE
3712 015666  104401  015674                TYPE    ,30049$         ;;TYPE ASCIZ STRING
     015672  000423                        BR      30048$          ;;GET OVER THE ASCIZ
                                ;;30049$:       .ASCIZ  <15><12>/TEST 23 - SLU BREAK CONDITION TEST/
     015742                     30048$:
3713 015742                     1000$:
3714
3715                            ;
3716                            ; SEND BREAK AND CHECK ERROR BITS IN RBUF
3717                            ;
3718
3719 015742  052737  000004  176504  1$:     BIS     #BIT02,XCSR1                    ;TRANSMIT IN LOOP BACK
3720 015750  052737  000001  176504          BIS     #BIT00,XCSR1                    ;SET SEND BREAK BIT
3721 015756  032737  000001  176504          BIT     #BIT00,XCSR1                    ;GOT SET OK?
3722 015764  001003                        BNE     2$                              ;IF YES, BRANCH
3723 015766  104027                        ERROR   +27                             ;WRITING 1 TO XCSR<0>
3724 015770  005037  172100                clr     @#RER
```

TEST - BREAK CONDITION

```
   3725 015774  012701  000100          2$:     MOV     #100,R1                         ;STALL DELAY
   3726 016000  105737  176500          4$:     TSTB    RCSR1                           ;RECEIVER READY?
   3727 016004  100401                          BMI     5$                              ;IF YES, BRANCH
   3728 016006  077104                          SOB     R1,4$                           ;WAIT JUST IN CASE OF A CHARACTER
   3729 016010  005737  176502          5$:     TST     RBUF1                           ;READ A CHARACTER
   3730 016014  052737  000001  176504        BIS     #BIT00,XCSR1                    ;TRANSMIT BREAK
   3731 016022  004737  030610                jsr     pc,delay                                ;  MOV  #1000,R1
 ;ANOTHER DELAY TO GET BREAK
   3732                                                                                 ;6$:    SOB     R1,6$
 ;WAIT A WHILE
   3733 016026  105737  176500                TSTB    RCSR1                           ;RECEIVER READY?
   3734 016032  100410                        bm      7$                              ;BPL    7$                              .IF
NOT, WAIT
   3735 016034  004737  030610                jsr     pc,delay
   3736 016040  105737  176500                tstb    rcsr1
   3737 016044  100403                        bmi     7$
   3738 016046  104076                        error   +76
   3739 016050  005037  172100                clr     mer
   3740
   3741 016054  013737  176502  001126  7$:   MOV     RBUF1,$BDDAT                    ;STORE WHATEVER RECEIVED
   3742 016062  022737  124000  001126        CMP     #BIT15!BIT13!BIT11,$BDDAT       ;ALL ERROR BITS SET?
   3743 016070  001407                        BEQ     8$                              ;IF YES, BRANCH
   3744 016072  042737  000004  176504        BIC     #BIT02,XCSR1                    ;RESET TO ENABLE SLU
   3745 016100  104025                        ERROR   +25                             ;BREAK DOES NOT CAUSE ERRORS
   3746 016102  005037  172100                clr     mer
   3747 016106  000467                        BR      TST24           ;;EXIT
   3748 016110  042737  000001  176504  8$:   BIC     #BIT00,XCSR1                    ;CLEAR TRANSMIT BREAK
   3749 016116  032737  000001  176504        BIT     #BIT00,XCSR1                    ;GOT CLEARED OK?
   3750 016124  001407                        BEQ     9$                              ;IF YES, BRANCH
   3751 016126  042737  000004  176504        BIC     #BIT02,XCSR1                    ;RESET TO ENABLE SLU
   3752 016134  104027                        ERROR   +27                             ;ERROR WRITING 0 TO XCSR<0>
   3753 016136  005037  172100                clr     mer
   3754 016142  000451                        BR      TST24           ;;EXIT
   3755                                 ;
   3756                                 ; CHECK THAT BREAK CONDITION IS CLEARED
   3757                                 ;
   3758 016144                          9$:;    MOV     SAVBR,BCSR                      ;RESTORE BCSR
   3759 016144  105737  176504                TSTB    XCSR1                           ;XMIT READY?
   3760 016150  100410                        bmi     10$                             ;BPL    10$                             .IF
NOT, WAIT
   3761 016152  004737  030610                jsr     pc,delay
   3762 016156  105737  176504                tstb    xcsr1
   3763 016162  100403                        bmi     10$
   3764 016164  104076                        error   +76
   3765 016166  005037  172100                clr     mer
   3766
   3767 016172  012737  000177  176506  10$:   MOV     #177,XBUF1                      ;TRY TO TRANSM   DELETE
   3768 016200  105737  176500                TSTB    RCSR1                           ;RECEIVER READY
   3769 016204  100410                        bmi     11$                             ;BPL    11$                             .IF
NOT, WAIT
   3770 016206  004737  030610                jsr     pc,delay
   3771 016212  105737  176500                tstb    rcsr1
   3772 016216  100403                        bmi     11$
   3773 016220  104076                        error   +76
   3774 016222  005037  172100                clr     mer
   3775
   3776 016226  013737  176502  001126  11$:   MOV     RBUF1,$BDDAT                    ;STORE RECEIVE BUFFER
   3777 016234  032737  124000  001126        BIT     #BIT15!BIT13!BIT11,$BDDAT       ;ERRORS CLEARED?
   3778 016242  001406                        BEQ     12$                             ;IF YES, BRANCH
   3779 016244  042737  000004  176504        BIC     #BIT02,XCSR1                    ;RESET TO ENABLE SLU
   3780 016252  104025                        ERROR   +25                             ;BREAK NOT CLEARED ON NEXT CHARACTER
   3781 016254  005037  172100                clr     mer
```

```
3782 016260  042737  000004  176504  12$:    BIC     #BIT02,XCSR1                  ;CLEAR LOOP BACK MODE
3783
3784
```

```
                                .SBTTL  TEST    OVERRUN CONDITION
                                ;CHECK OVERRUN CONDITION
                                ;
                                ;RCSR   <14>    OVERRUN ERROR
                                ;
                                ; SLU #1 IS TESTED SINCE SLU #0 IS BEING USED AS THE CONSOLE
                                ;
                                ;ROUTINE TEST
                                ;.      LET XCSR<02>=#1 (LOOPBACK MODE)
                                ;.      WAIT FOR XCSR<07>=#1
                                ;.      LET XBUF=#252
                                ;.      WAIT FOR XCSR<07>=#1
                                ;.      LET XBUF=#125 (SEND THE 2ND W/O READING THE 1ST CHARACTER)
                                ;.      WAIT FOR RCSR<07>=#1
                                ;.      STALL FOR LOWEST BAUD RATE TO GET 2ND CHARACTER
                                ;.      IF LOW BYTE OF RBUF NE #125 THEN
                                ;.      .       ERROR (1ST CHARACTER WASN'T OVERRUN)
                                ;.      ENDIF
                                ;.      IF RBUF<15!14> NE #1 THEN
                                ;.      .       ERROR (NO OVERRUN BIT SET)
                                ;.      ENDIF
                                ;.      WAIT FOR XCSR<07>=#1
                                ;.      LET XBUF=#NULL
                                ;.      WAIT FOR RCSR<07>=#1
                                ;.      IF RBUF<15!14> NE #0 THEN
                                ;.      .       ERROR (WASN'T CLEARED ON THE NEXT CHARACTER RECEIVED)
                                ;.      ENDIF
                                ;.      LET XCSR<02>=#0
                                ;ENDROUTINE

                                ;;*****************************************************************
 016266  000004                 TST24:  SCOPE
 016270  032777  010000  162642         BIT     #BIT12,@SWR     ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
 016276  001432                 BEQ     100$            ;       THEN TYPE TEST TRACE
 016300  104401  016306         TYPE    ,30051$         ;;TYPE ASCIZ STRING
 016304  000424                 BR      30050$          ;;GET OVER THE ASCIZ
                                ;;30051$:       .ASCIZ  <15><12>/TEST 24   SLU OVERRUN CONDITION TEST/
 016356                         30050$:
 016356  012701  100000                 MOV     #100000,R1
 016362  077101                 1001$:  SOB     R1,1001$        ; WAIT FOR LAST CHARACTER
 016364  052737  000004  176504 100$:   BIS     #BIT02,XCSR1                    ;SET LOOP BACK MODE
 016372  105737  176504                 tstb    xcsr1                           ;1$:    TSTB    XCSR1
;READY TO TRANSMIT?
 016376  100410                 bmi     1$                              ;BPL    1$                      ,IF
NOT, WAIT
 016400  004737  030610                 jsr     pc,delay
 016404  105737  176504                 tstb    xcsr1
 016410  100403                 bmi     1$
 016412  104076                 error   +76
 016414  005037  172100                 clr     mer
 016420  012737  000021  176506 1$:     MOV     #21,XBUF1                       ;TRANSMIT A CHARACTER
 016426  105737  176500                 tstb    rcsr1                           ;2$:    TSTB    RCSR1
;RECEIVE READY?
 016432  100410                 bmi     2$                              ;BPL    2$                      ,IF
NOT, WAIT
 016434  004737  030610                 jsr     pc,delay
 016440  105737  176500                 tstb    rcsr1
 016444  100403                 bmi     2$
 016446  104076                 error   +76
 016450  005037  172100                 clr     mer
```

```
TEST    OVERRUN CONDITION

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

TEST   OVERRUN CONDITION

```
3896 016716  000240                NOSLU:  NOP             ;WE SKIPPED ALL SLU TEST IN APT MODE ON MORE THAN 1 PASS
3897
3898
```

```
                                        .SBTTL  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
                                        ;;*****************************************************************
016720  000004                    TST25:  SCOPE
016722  032777  010000  162210          BIT     #BIT12,@SWR     ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
016730  001423                          BEQ     1000$           ;    THEN TYPE TEST TRACE
016732  104401  016740                  TYPE    ,30053$         ;;TYPE ASCIZ STRING
016736  000420                          BR      30052$          ;;GET OVER THE ASCIZ
                                        ;;30053$:       .ASCIZ  <15><12>/TEST 25 - KDJ11-DA LED TESTS/
017000                                  30052$:

017000                                  1000$:
017000  005005                          CLR     R5                              ;FLAG IN NO INTERRRUPT MODE
017002  032737  000001  000052          BIT     #BIT00,@#52                     ;IF RUNNING IN CHAIN MODE
017010  001413                          BEQ     1$                              ;SKIP PRINTOUTS
017012  005737  001206                  TST     $PASS                           ;1ST PASS?
017016  001010                          BNE     1$                              ;IF NOT, SKIP PRINTOUTS
017020  122737  000001  001220          CMPB    #APTENV,$ENV                    ;APT MODE?
017026  001404                          BEQ     1$                              ;YES, SKIP PRINTOUT'S
017030  005105                          COM     R5                              ;CLEAR FLAG IN INTERRUPT MODE
017032  012737  000005  002354          MOV     #5,LEDCNT                       ;DO PATTERN 5 TIMES     ;MC

017040  012704  000020          1$:     MOV     #20,R4                          ;FOR EACH LOOP
017044  012701  000000                  MOV     #0,R1                           ;START WITH 1
017050  110137  177520          2$:     MOVB    R1,@#NATREG                     ;TURN OFF FIRST LED
017054  012703  000004                  MOV     #4,R3                           ;STALL DELAY
017060  012702  177777          3$:     MOV     #177777,R2                      ;STALL DELAY
017064  077201                  4$:     SOB     R2,4$                           ;WAIT A WHILE
017066  077304                          SOB     R3,3$                           ;WAIT A WHILE
017070  000241                          CLC                                     ;

017072                                  7$:
017072  005201                          INC     R1                              ; 1.............19 OCTAL     ;MC
017074  077413                          SOB     R4,2$                           ; 20.............0 OCTAL     ;MC
017076  005705                          TST     R5                              ;RUNNING IN INTERACTIVE MODE?
017100  001411                          BEQ     6$                              ;IF NOT, EXIT
017102  104401  001170                  TYPE    ,$BELL                          ;
017106  005337  002354                  DEC     LEDCNT
017112  001352                          BNE     7$                              ;REPEAT PATTERN 5 TIMES

017114  005737  177562          5$:     TST     RBUF                            ;READ BUFFER
```

F8

SEQ 009#

TEST - LED'S ON

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

```
                                        .SBTTL  TEST    DIFFERENT LEVELS OF INTERRUPTS
                                        ;DIFFERENT LEVELS OF INTERRUPTS
                                        ;THIS TEST WILL PROGRAM Q22 BUS EXERCISER TO INTERRUPT AT DIFFERENT
                                        ;LEVELS.  ARBITRATION BETWEEN DIFFERENT LEVELS OF INTERRUPTS AND
                                        ;PIRQ'S WILL BE TESTED.
                                        ;
                                        ;CHECK DIFFERENT LEVELS OF INTERRUPTS.
                                        ;ROUTINE TEST
                                        ;.      SET VECTOR TO INTERRUPT_DMA
                                        ;.      FOR INTERRUPTS FROM 4 TO 7 DO
                                        ;.      .       ENABLE INTERRUPTS
                                        ;.      .       SET PRIORITY=INTERUPT
                                        ;.      .       IF INTERRUPT_FLAG SET THEN
                                        ;.      .       .       ERROR
                                        ;.      .       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

                                        ;;*****************************************************************
017134  000004                          TST26:  SCOPE
                                        ;       BIT     #BIT07,@#52                     ;UFD MODE?
                                        ;       SKIP    NE,<IF SO, EXIT TEST>
017136  005737  002334                  ;       TST     CSR1                            ;AT LEAST ONE Q22BE FOUND?
017142  001534                          ;       BEQ     TST27           ;;IF NOT, EXIT TEST
017144  032777  010000  161766          ;       BIT     #BIT12,@SWR     ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGISTER
017152  001444                          ;       BEQ     1000$           ;    THEN TYPE TEST TRACE
017154  104401  017162                  ;       TYPE    ,30055$         ;;TYPE ASCIZ STRING
017160  000441                          ;       BR      30054$          ;;GET OVER THE ASCIZ
                                        ;;30055$:       .ASCIZ  <15><12>/TEST 26 - ARBITRATION BETWEEN CPU PRIORITY AND Q BUS INTERR
UPTS/
017264                                  30054$:
017264                                  1000$:
                                        ;
                                        ; SETUP INITIAL PRIORITY TO 7
                                        ;
017264  013703  002334                  ;       MOV     CSR1,R3                         ;DO FOR FIRST FOUND Q22BE
017270  012777  017340  163050          ;       MOV     #5$,@VQBE1                      ;POINT INTERRUPT VECTOR TO PROGRAM
017276  012777  000340  163044          ;       MOV     #340,@VQPR1                     ;AT PRIORITY 7
017304  012700  003002                  ;       MOV     #Q22EN,R0                       ;START WITH 4 FOR INTERRUPTS
017310  012701  000340                  ;       MOV     #340,R1                         ;LOW BOUNDARY FOR NO INTERRUPTS
                                        ;
                                        ; CHECK THAT INTERRUPTS DON'T HAPPEN AT PRIORITY HIGHER THAN BR
                                        ;
017314                                  2$:
017314  106427  000200                  4$:     MTPS    #200                            ;SET PRIORITY NOT TO INTERRUPT
017320  004737  030026                  ;       JSR     PC,Q22INT                       ;ENABLE INTERRUPTS
017324  012077  163006                  ;       MOV     (R0)+,@CSR2                     ;CLEAR GO BIT
```

TEST DIFFERENT LEVELS OF INTERRUPTS

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

```
                                        .SBTTL  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
                                        ;.      ENDDO
                                        ;ENDROUTINE

                                        ;;*****************************************************************
 017434  000004                         TST27:  SCOPE
                                        ;       BIT     #BIT07,@#52                     ;UFD MODE?
                                        ;       SKIP    NE,<IF SO, EXIT TEST>
 017436  005737  002334                 .       TST     CSR1                            ;AT LEAST ONE Q22BE FOUND?
 017442  001523                         .       BEQ     TST30           ;;IF NOT, EXIT TEST
 017444  032777  010000  161466         .       BIT     #BIT12,@SWR                     ; IF BIT 12 IS SET IN SOFTWARE SWITCH REGIST
ER
 017452  001441                         .       BEQ     1000$                           ;    THEN TYPE TEST TRACE
 017454  104401  017462                 .       TYPE    ,30057$         ;;TYPE ASCIZ STRING
 017460  000436                         .       BR      30056$          ;;GET OVER THE ASCIZ
                                        ;;30057$:       .ASCIZ  <15><12>/TEST 27   ARBITRATION BETWEEN PIRQ'S AND Q-BUS INTERRUPTS/
 017556                                 30056$:
 017556                                 1000$:
 017556  012777  017656  162562         .       MOV     #3$,@VQBE1                      ;SETUP Q22BE VECTOR
 017564  012777  000340  162556         .       MOV     #340,@VQPR1                     ;AT PRIORITY 7
 017572  012737  017672  000240         .       MOV     #4$,PIRQVEC                     ;SETUP PIRQ VECTOR
 017600  012737  000340  000242         .       MOV     #340,PIRQVEC+2                  ;AT PRIORITY 7
 017606  012700  003002                 .       MOV     #Q22EN,R0                       ;POINT THRU PRIORITIES FOR Q22BE
 017612  012704  017710                 .       MOV     #PIRQT,R4                       ;POINTER THRU PIRQ'S
 017616  013703  002334                 .       MOV     CSR1,R3                         ;DO FOR FIRST Q22BE
 017622  012702  000140                 .       MOV     #140,R2                         ;START WITH CPU PRIORITY AT 7
 017626  106427  000340         2$:     .       MTPS    #340                            ;RAISE PRIORITY TO 7
 017632  012437  177772                 .       MOV     (R4)+,PIRQ                      ;SET PRIORITY FOR PIRQ'S
 017636  004737  030026                 .       JSR     PC,Q22INT                       ;INITIALISE Q22BE TO INTERRUPT
 017642  012077  162470                 .       MOV     (R0)+,@CSR2                     ;SET DONE BIT
 017646  106402                         .       MTPS    R2                              ;LOWER PRIORITY
 017650  000240                         .       NOP
 017652  000240                         .       NOP
 017654  000240                         .       NOP
 017656  104035                 3$:     .       ERROR   +35                             ;PIRQ'S DON'T TAKE OVER BIRQ S
 017660  005037  172100                 .       clr     mer
 017664  005726                         .       TST     (SP)+                           ;CLEAN UP STACK
 017666  005726                         .       TST     (SP)+
 017670  000402                         .       BR      5$                              ;BRANCH AROUND PIRQ INTERRUPT
 017672  005726                 4$:     .       TST     (SP)+                           ;CLEAN UP STACK
 017674  005726                         .       TST     (SP)+
 017676  005037  177772         5$:     .       CLR     PIRQ                            ;CLEAR ANY REQUESTS
 017702  005077  162430                 .       CLR     @CSR2                           ;CLEAR JUST IN CASE
 017706  000401                         .       BR      TST30           ;;GO TO THE NEXT TEST

 017710  010000                 PIRQT: .WORD    10000   ;PIRQ'S 4
```

```
4096                                    ;;*****************************************************************
     017712  000004                     TST30:  SCOPE
4097 017714  005037  177572                     CLR     @#SR0                   ;DISABLE MMU (DO NOT REMOVE !!!)
4098 017720  000137  031632                     JMP     $EOP                    ;EXIT
4099
4100 017724  123727  001220  000001     VIREOP: CMPB    $ENV,#1                 ; if not APT, don't worry about
4101 017732  001005                             BNE     1$                      ;
4102 017734  005737  003034                     TST     CCHPAS                  ; maintain cache routin pascnt
4103 017740  001402                             BEQ     1$
4104 017742  005337  003034                     DEC     CCHPAS
4105
4106                                            ; This VIREOP ROUTINE to provide common End of Pass exit point
4107 017746  000205                     1$:     rts     r5
4108
```

```
                                                .SBTTL  GLOBAL ERROR MESSAGES
                                        SWTSEL:
017750 015 012 113                              .ASCII  <15><12>/KDJ11 DA CPU DIAGNOSTIC - COKDDA0/<15><12><12>
017753 104 112 061
017756 061 055 104
017761 101 040 103
017764 120 125 040
017767 104 111 101
017772 107 116 117
017775 123 124 111
020000 103 040 055
020003 040 103 117
020006 113 104 104
020011 101 060 015
020014 012 012
020016 123 127 111                              .ASCII  /SWITCH REGISTER SELECTION:/<12><12><15>
020021 124 103 110
020024 040 122 105
020027 107 111 123
020032 124 105 122
020035 040 123 105
020040 114 105 103
020043 124 111 117
020046 116 072 012
020051 012 015
020053 102 111 124                              .ASCII  /BIT NUMBER                             USE/<12><15>
020056 040 116 125
020061 115 102 105
020064 122 011 011
020067 011 011 125
020072 123 105 012
020075 015
020076 055 055 055                              .ASCII  /  --------                     ----- ----  --- ---/<12><15>
020101 055 055 055
020104 055 055 055
020107 055 011 011
020112 011 055 055
020115 055 055 055
020120 055 055 055
020123 055 055 055
020126 055 055 055
020131 055 055 055
020134 055 055 012
020137 015
020140 011 061 065                              .ASCII  /       15                      HALT ON ERROR/<12><15>
020143 011 011 011
020146 110 101 114
020151 124 040 117
020154 116 040 105
020157 122 122 117
020162 122 012 015
020165 011 061 064                              .ASCII  /       14                      LOOP ON PRESENT TEST/<12><15>
020170 011 011 011
020173 114 117 117
020176 120 040 117
020201 116 040 120
020204 122 105 123
```

GLOBAL ERROR MESSAGES

```
                020207  105     116     124
                020212  040     124     105
                020215  123     124     012
                020220  015
 4118           020221  011     061     063     .ASCII  /       13      INHIBIT ERROR TYPEOUTS/<12><15>
                020224  011     011     011
                020227  111     116     110
                020232  111     102     111
                020235  124     040     105
                020240  122     122     117
                020243  122     040     124
                020246  131     120     105
                020251  117     125     124
                020254  123     012     015
 4119           020257  011     061     062     .ASCII  /       12      ENABLE TEST TRACING/<12><15>
                020262  011     011     011
                020265  105     116     101
                020270  102     114     105
                020273  040     124     105
                020276  123     124     040
                020301  124     122     101
                020304  103     111     116
                020307  107     012     015
 4120           020312  011     061     061     .ASCII  /       11      INHIBIT ITERATIONS/<12><15>
                020315  011     011     011
                020320  111     116     110
                020323  111     102     111
                020326  124     040     111
                020331  124     105     122
                020334  101     124     111
                020337  117     116     123
                020342  012     015
 4121           020344  011     061     060     .ASCII  /       10      BELL ON ERROR/<12><15>
                020347  011     011     011
                020352  102     105     114
                020355  114     040     117
                020360  116     040     105
                020363  122     122     117
                020366  122     012     015
 4122           020371  011     040     071     .ASCII  /       9       LOOP ON ERROR/<12><15>
                020374  011     011     011
                020377  114     117     117
                020402  120     040     117
                020405  116     040     105
                020410  122     122     117
                020413  122     012     015
 4123           020416  011     040     070     .ASCII  /       8       LOOP ON TEST IN SWR<5-0>/<12><15>
                020421  011     011     011
                020424  114     117     117
                020427  120     040     117
                020432  116     040     124
                020435  105     123     124
                020440  040     111     116
                020443  040     123     127
                020446  122     074     065
                020451  055     060     076
                020454  012     015
```

GLOBAL ERROR MESSAGES

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

EQ 0104

GLOBAL ERROR MESSAGES

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

GLOBAL ERROR MESSAGES

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

GLOBAL ERROR MESSAGES

```
                021416  040     103     114
                021421  105     101     122
                021424  105     104     040
                021427  101     106     124
                021432  105     122     040
                021435  122     105     101
                021440  104     111     116
                021443  107     040     122
                021446  102     125     106
                021451  000
 4145           021452  130     103     123     EM77:   .ASCIZ  /XCSR<07> NOT SET ON RESET/
                021455  122     074     060
                021460  067     076     040
                021463  116     117     124
                021466  040     123     105
                021471  124     040     117
                021474  116     040     122
                021477  105     123     105
                021502  124     000
 4146           021504  122     103     1.      EM100:  .ASCIZ  /RCSR<07> NOT CLEARED ON RESET/
                021507  122     074     06
                021512  067     076     040
                021515  116     117     124
                021520  040     103     114
                021523  105     101     122
                021526  105     104     040
                021531  117     116     040
                021534  122     105     123
                021537  105     124     000
 4147           021542  123     114     125     EM101:  .ASCIZ  /SLU INTERRUPTS HAPPEN AT 4/
                021545  040     111     116
                021550  124     105     122
                021553  122     125     120
                021556  124     123     040
                021561  110     101     120
                021564  120     105     116
                021567  040     101     124
                021572  040     064     000
 4148           021575  122     105     123     EM102:  .ASCIZ  /RESET DOES NOT CLEAR PROPER BITS IN SLU REGISTERS/
                021600  105     124     040
                021603  104     117     105
                021606  123     040     116
                021611  117     124     040
                021614  103     114     105
                021617  101     122     040
                021622  120     122     117
                021625  120     105     122
                021630  040     102     111
                021633  124     123     040
                021636  111     116     040
                021641  123     114     125
                021644  040     122     105
                021647  107     111     123
                021652  124     105     122
                021655  123     000
 4149           021657  124     122     101     EM103:  .ASCIZ  /TRANSMIT INTERRUPT DOES NOT CLEAR XCSR<07>/
                021662  116     123     115
```

GLOBAL ERROR MESSAGES

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

GLOBAL ERROR MESSAGES

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

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

GLOBAL ERROR MESSAGES

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

GLOBAL ERROR MESSAGES

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

GLOBAL ERROR MESSAGES

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

GLOBAL ERROR MESSAGES

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

GLOBAL ERROR MESSAGES

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

GLOBAL ERROR MESSAGES

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

GLOBAL ERROR MESSAGES

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

GLOBAL ERROR MESSAGES

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

GLOBAL ERROR MESSAGES

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

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

GLOBAL ERROR MESSAGES

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

E10

GLOBAL ERROR MESSAGES

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

GLOBAL ERROR MESSAGES

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

G10

MODIFIED ERROR MESSAGE TYPEOUT ROUTINE

```
                                        .SBTTL  MODIFIED ERROR MESSAGE TYPEOUT ROUTINE
                                        ;;*********************************************************************
                                        ;*THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
                                        ;*ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
                                        ;*AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
                                        ;*
                                        ;*THE ONLY DIFFERENCE BETWEEN THIS ROUTINE AND THE ORIGINAL "$ERRTYP" FROM
                                        ;*SYSMAC IS THAT YOU CAN PASS INFORMATION IN GENERAL PURPOSE REGISTERS TO THIS
                                        ;*ROUTINE. THE GENERAL PURPOSE REGISTERS USED ARE TO BE SPECIFIED IN DT*
                                        ;*FORMAT. R0 SHOULD NOT BE USED.

027030                                  ERTYPE:
027030  005037  001162                          CLR     $TMP1           ;;JUST CLEAR IT
027034  113737  001102  001162                  MOVB    $TSTNM,$TMP1    ;;STORE TEST NUMBER
027042  104401  001175                          TYPE    ,$CRLF          ;;'CARRIAGE RETURN" & "LINE FEED"
027046  010046                                  MOV     R0, -(SP)       ;;SAVE R0
027050  005000                                  CLR     R0              ;;PICKUP THE ITEM INDEX
027052  153700  001114                          BISB    @#$ITEMB,R0
027056  001004                                  BNE     1$              ;;IF ITEM NUMBER IS ZERO, JUST
                                                                        ;;TYPE THE PC OF THE ERROR
027060  013746  001116                          MOV     $ERRPC, (SP)    ;;SAVE $ERRPC FOR TYPEOUT
                                                                        ;;ERROR ADDRESS
027064  104402                                  TYPOC                   ;;GO TYPE- OCTAL ASCII(ALL DIGITS)
027066  000426                                  BR      6$              ;;GET OUT
027070  005300                          1$:     DEC     R0              ;;ADJUST THE INDEX SO THAT IT WILL
027072  006300                                  ASL     R0              ;;      WORK FOR THE ERROR TABLE
027074  006300                                  ASL     R0
027076  006300                                  ASL     R0
027100  062700  001324                          ADD     #$ERRTB,R0      ;;FORM TABLE POINTER
027104  012037  027114                          MOV     (R0)+,2$        ;;PICKUP 'ERROR MESSAGE" POINTER
027110  001404                                  BEQ     3$              ;;SKIP TYPEOUT IF NO POINTER
027112  104401                                  TYPE
027114  000000                          2$:     .WORD   0               ;;ERROR MESSAGE POINTER GOES HERE
027116  104401  001175                          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
027122  012037  027132                  3$:     MOV     (R0)+,4$        ;;PICKUP "DATA HEADER" POINTER
027126  001404                                  BEQ     5$              ;;SKIP TYPEOUT IF 0
027130  104401                                  TYPE                    ;;TYPE THE "DATA HEADER"
027132  000000                          4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
027134  104401  001175                          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED'
027140  011000                          5$:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
027142  001004                                  BNE     7$              ;;GO TYPE THE DATA
027144  012600                          6$:     MOV     (SP)+,R0        ;;RESTORE R0
027146  104401  001175                          TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED'
027152  000207                                  RTS     PC              ;;RETURN
027154                                  7$:
027154  021027  000005                          CMP     (R0),#5         ;;GENERAL PURPOSE REGISTER?
027160  101021                                  BHI     9$              ;;IF NOT, GO TYPE DATA
027162  042737  000700  027216                  BIC     #BIT8!BIT7!BIT6,8$      ;;CLEAR BITS FOR SOURCE REGISTER
027170  011037  001160                          MOV     (R0),$TMP0      ;;SAVE (R0)
027174  000337  001160                          SWAB    $TMP0           ;;GET REGISTER NUMBER TO HIGH BYTE
027200  006237  001160                          ASR     $TMP0           ;;GET REGISTER NUMBER TO BITS 8 6
027204  006237  001160                          ASR     $TMP0
027210  053737  001160  027216                  BIS     $TMP0,8$        ;;SET BITS IN MOV INSTRUCTION
                                                                        ;;ACCORDING TO REGISTER NUMBER
027216  010046                          8$:     MOV     R0, (SP)        ;;MOVE CONTEXT OF REGISTER TO STACK
027220  005720                                  TST     (R0)+           ;;ADVANCE POINTER
027222  000401                                  BR      10$             ;;GO TYPE
```

MODIFIED ERROR MESSAGE TYPEOUT ROUTINE

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

```
                                ;++
                                ; 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
                                ;
027410  012702  000006          LDPARS: MOV     #6,     R2              ;LET LOOP COUNTER COUNT FIRST 6 PARS
027414  005003                          CLR     R3                      ;INITIALIZE INDEX VALUE
027416  010321                  1$:     MOV     R3,     (R1)+           ;LOAD PARS
027420  062703  000200                  ADD     #200,   R3              ;INDEX IN 4K INCREMENTS
027424  077204                          SOB     R2,     1$              ;LOAD FIRST SIX PARS
027426  012721  002000                  MOV     #2000,  (R1)+           ;LET PAR6 MAP TO 200000
027432  012711  177600                  MOV     #177600,(R1)            ;LET PAR7 MAP TO I/O PAGE
027436  000207                          RTS     PC                      ;RETURN
```

GLOBAL SUBROUTINES SECTION

```
4395                                    ;++
4396                                    ; FUNCTIONAL DESCRIPTION:
4397                                    ;     SUBROUTINE TO INITIALIZE ALL THE MMU PAGE DECRIPTOR REGISTERS (PDRS).
4398                                    ;        THIS ROUTINE WILL INITIALIZE 8 PDRS STARTING AT A BASE ADDRESS
4399                                    ;        SUPPLIED BY THE CALLING ROUTINE. PDRS 0 5 WILL BE INITIALIZED TO
4400                                    ;        4K READ/WRITE BYPASS AND PDRS 6 AND 7 WILL BE INITIALIZED TO
4401                                    ;        4K READ/WRITE NO BYPASS.
4402                                    ;        NOTE:    THERE IS NO NEED TO BYPASS ON I/O PAGE REFERENCES BECAUSE
4403                                    ;                 THE CACHE DOES NOT ALLOCATE ANY OF THESE REFERENCES.
4404
4405                                    ; INPUTS:
4406                                    ;        R1 CONTAINS THE BASE ADDRESS OF THE NEXT 8 PDRS TO BE INITIALIZED
4407
4408                                    ; OUTPUTS: NONE
4409
4410                                    ; SUBORDINATE ROUTINES USED: NONE
4411
4412                                    ; FUNCTIONAL SIDE EFFECTS: NONE
4413
4414                                    ; CALLING SEQUENCE:     JSR     PC,LDPARS
4415
4416 027440  012702  000006          LDPDRS: MOV     #6,     R2              ;LET LOOP COUNTER COUNT FIRST 6 PARS
4417 027444  012721  177406          1$:     MOV     #177406,(R1)+           ;LOAD PDRS WITH 4K READ/WRITE BYPASS
4418 027450  077203                          SOB     R2,     1$              ;LOAD FIRST SIX PDRS
4419 027452  012721  077406                  MOV     #77406, (R1)+           ;LET PAR6 BE 4K READ/WRITE NO BYPASS
4420 027456  012711  077406                  MOV     #77406, (R1)            ;LET PAR7 BE 4K READ/WRITE NO BYPASS ALSO
4421 027462  000207                          RTS     PC                      ;RETURN
```

K10

GLOBAL SUBROUTINES SECTION

```
                                        ;**
                                        ; FUNCTIONAL DESCRIPTION:
                                        ;    SUBROUTINE TO HANDLE PARITY ERROR ABORTS FROM THE RAM STORE RAM TESTS.
                                        ;
                                        ; INPUTS:
                                        ;       MEMORY SYSTEM ERROR REGISTER CONTAINS BITS INDICATING FAILURE
                                        ;
                                        ; OUTPUTS: NONE
                                        ;
                                        ; SUBORDINATE ROUTINES USED: NONE
                                        ;
                                        ; FUNCTIONAL SIDE EFFECTS: NONE
                                        ;
                                        ; CALLING SEQUENCE: CALLED BY PARITY ABORT
                                        ;       MOV     @#114,  SLOCOO  ;SAVE CONTENTS OF PARITY ABORT VECTOR
                                        ;       MOV     #DSPAR, @#114   ;LET VECTOR POINT TO PARITY ABORT ROUTINE
                                        ;
                                        ;       (CACHE PARITY ERROR OCCURS)
027464  011637  001122        RAMPAR: MOV     (SP),$BDADR             ;STOR ADDESS TRAPPED
027470  032737  000100  177744        BIT     #BIT06, MSER            ;IF LOW BYTE PARITY ERROR
027476  001401                       BEQ     1$                      ;THEN
027500  104004                       ERROR   .4                      ;ERROR
027502                          1$:                                     ;       BIT     #BIT07, MSER            ;IF HIGH BYT
E PARITY ERROR
                                                                        ;       BEQ     2$                      ;THEN
                                                                        ;       ERROR   .4                      ;ERROR
027502  032737  000040  177744  2$:     BIT     #BIT05, MSER            ;IF TAG PARITY ERROR
027510  001401                       BEQ     3$                      ;THEN
027512  104004                       ERROR   .4                      ;ERROR
027514  005037  177744               3$:     CLR     MSER                    ;INITIALIZE MSER AFTER ERROR
027520  000002                       RTI                             ;RETURN
027522  005237  002402        LKSINT: INC     LKSFL                   ;INCREMENT FLAG
027526  000002                       RTI

027530                          clkcnt:
027530  022737  000003  002404  100$:   cmp     #3,lkcnt
027536  001402                       beq     101$
027540  005237  002404               inc     lkcnt
027544  000002                  101$:   rti

                                        .SBTTL Q22BE SIZE ROUTINE
                                        ;THIS ROUTINE WILL AUTOSIZE FOR UP TO TWO Q22 BUS EXERCISERS.  IF NONE
                                        ;FOUND LOCATIONS CSR1 AND CSR12 WILL BE LEFT ZEROES.  THIS ROUTINE WILL
                                        ;ONLY RUN IN NOT UFD MODE.

027546  032737  001000  177750  Q22SIZ: BIT     #BIT09,MAIREG                   ;UNIBUS SYSTEM?
027554  001401                       BEQ     1$                              ;IF NOT, ADVANCE TO ROUTINE
027556  000207                       RTS     PC                              ;OTHERWISE, RETURN
                                        ;
                                        ; PREPARE TO DO SIZING
                                        ;
027560  013701  000004          1$:     MOV     ERRVEC,R1                       ;STORE TIMEOUT VECTOR
027564  012737  027740  000004        MOV     #7$,ERRVEC                      ;POINT NEW TO PROGRAM
027572  012737  000340  000006        MOV     #340,ERRVEC+2                   ;AT PRIORITY 7
027600  005037  001160               CLR     $TMP0                           ;CLEAR Q22BE COUNTER
027604  012702  170000               MOV     #170000,R2                      ;FIRST POSSIBLE ADDRESS
```

Q22BE SIZE ROUTINE

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

```
                                        ;AT THE STARTING ADDRESS IN R3. THE TEST HAVE TO SET ACTUAL DONE BIT
                                        ;BY CLEARING GO.
                                        
030026  005013                Q22INT: CLR     (R3)                            ;CLEAR TRANSFER TYPE IN CSR1
030030  052710  000001                BIS     #BIT00,(R0)                     ;ZERO DONE
030034  011063  000002                MOV     (R0),2(R3)                      ;SET PRIORITY IN CSR2
030040  042710  000001                BIC     #BIT00,(R0)                     ;PREPARE TO SET DONE
030044  000207                        RTS     PC
                                        
                                        .SBTTL DMATRN DATO CYCLE THRU Q22BE
                                        ;THIS ROUTINE PERFORMS DATO FROM A LOCATION TEMP THRU THE FIRST
                                        ;FOUND Q22BE STARTING AT LOCATION @CSR1. R0 HAS 0 IF ONLY 1 TRANSFER IS
                                        ;TO BE PERFORMED. OTHERWISE 16 BLOCK MODE TRANSFERS ARE TO BE PERFORMED.
                                        ;IN THE LATTER CASE ADDRESS AND WORD COUNT HAS TO BE LOADED BEFORE.
                                        
030046  012777  012525  152270  DMATRN: MOV     #12525,@DATA                    ;DATA USED
030054  005700                        TST     R0                              ;DO 1 WORD?
030056  001404                        BEQ     1$                              ;IF YES, BRANCH
030060  012777  001001  152250        MOV     #BIT09!BIT00,@CSR2              ;BLOCK MODE, GO
030066  000414                        BR      2$                              ;BRANCH TO DO IT
030070  012777  001601  152236  1$:   MOV     #1601,@CSR1                     ;RESET LATENCY COUNT,DATO
030076  012777  002740  152234        MOV     #TEMP,@BA                       ;LOAD DMA ADDRESS
030104  012777  177777  152230        MOV     #177777,@WC                     ;DO 1 WORD
030112  012777  000001  152216        MOV     #BIT00,@CSR2                    ;DO IT
030120  105777  152212        2$:     TSTB    @CSR2                           ;DMA DONE?
030124  100375                        BPL     2$                              ;WAIT TILL DONE
030126  000207                3$:     RTS     PC                              ;RETURN FROM SUBROUTINE
                                        
                                        .SBTTL DMARD DATI THRU Q22BE
                                        ;THIS ROUTINE PERFORMS DATI CYCLE THRU Q22BE IN EITHER BLOCK MODE OR A SINGLE
                                        ;TRANSFER MODE. MEMORY LOCATION USED IS TEMP. R0 IS ZERO FOR SINGLE TRANSFER
                                        
030130  012777  002740  152202  DMARD:  MOV     #TEMP,@BA                       ;LOAD DMA ADDRESS
030136  005700                        TST     R0                              ;DO 1 WORD?
030140  001412                        BEQ     1$                              ;IF YES, BRANCH
030142  012777  001507  152164        MOV     #1507,@CSR1                     ;16 DATIB
030150  012777  177770  152164        MOV     #177770,@WC                     ;DO 8 WORD
030156  012777  001001  152152        MOV     #BIT09!BIT00,@CSR2              ;BLOCK MODE GO
030164  000411                        BR      2$                              ;GO CHECK
030166  012777  001407  152140  1$:   MOV     #1407,@CSR1                     ;RESET LATENCY COUNT,DATI
                                                                                ;LOAD NEW DATA TO DATA R.
030174  012777  177777  152140        MOV     #177777,@WC                     ;DO 1 WORD
030202  012777  000001  152126        MOV     #BIT00,@CSR2                    ;DO IT
030210  105777  152122        2$:     TSTB    @CSR2                           ;DMA DONE?
030214  100375                        BPL     2$                              ;WAIT TILL DONE
030216  000207                3$:     RTS     PC                              ;RETURN FROM SUBROUTINE
                                        
                                        
                                        
030220  011637  001122          TOUT:   MOV     (SP),$BDADR             ;STORE TRAPPED PC
030224  104041                          ERROR   +41                     ;UNEXPECTED TRAP
030226  000002                          RTI
                                        
                                        ;
                                        ;MMU GLOBAL SUBROUTINES
                                        ;
                                        ;
```

```
4586                                    ;ROUTINE TO INITIALIZE MEMORY MANAGEMENT
4587                                    ;
4588 030230  010046                     MMU:    MOV     R0, (SP)                ;SAVE CONTENTS OF REGISTERS
4589 030232  010146                             MOV     R1, (SP)                ;
4590 030234  010246                             MOV     R2, (SP)                ;
4591 030236  012700  177600                     MOV     #177600,R0              ;
4592 030242  004737  030330                     JSR     PC,PDR                  ;INIT I AND D USER PDR S
4593 030246  004737  030352                     JSR     PC,PAR                  ;INIT I USER PAR'S
4594 030252  004737  030352                     JSR     PC,PAR                  ;INIT D USER PAR'S
4595 030256  012700  172200                     MOV     #172200,R0
4596 030262  004737  030330                     JSR     PC,PDR                  ;INIT I AND D SUP PDR'S
4597 030266  004737  030352                     JSR     PC,PAR                  ;INIT I SUP PAR'S
4598 030272  004737  030352                     JSR     PC,PAR                  ;INIT D SUP PAR'S
4599 030276  004737  030330                     JSR     PC,PDR                  ;INIT I AND D KER PDP S
4600 030302  004737  030352                     JSR     PC,PAR                  ;INIT I KER PAR'S
4601 030306  004737  030352                     JSR     PC,PAR                  ;INIT D KER PAR'S
4602 030312  012737  000027  172516             MOV     #27,@#172516            ;INIT MMR3
4603 030320  012602                             MOV     (SP)+,R2                ;RESTORE REGISTERS
4604 030322  012601                             MOV     (SP)+,R1                ;
4605 030324  012600                             MOV     (SP)+,R0                ;
4606 030326  000207                             RTS     PC                      ;RETURN
4607                                    ;
4608                                    ;ROUTINE TO INITIALIZE PDR'S
4609                                    ;
4610 030330  005002                     PDR:    CLR     R2                      ;INIT CNTR
4611 030332  012720  077406             PDR1:   MOV     #77406,(R0)+            ;INIT PDR
4612 030336  062702  000001                     ADD     #1,R2                   ;INCREMENT CNTR
4613 030342  022702  000020                     CMP     #16.,R2                 ;ARE WE DONE?
4614 030346  001371                             BNE     PDR1                    ;BRANCH IF NOT
4615 030350  000207                             RTS     PC                      ;RETURN
4616                                    ;
4617                                    ;ROUTINE TO INITIALIZE PAR'S
4618                                    ;
4619 030352  005001                     PAR:    CLR     R1                      ;SETUP TO INIT PAR
4620 030354  010120                     PAR1:   MOV     R1,(R0)+                ;INIT PAR
4621 030356  062701  000200                     ADD     #200,R1                 ;GET READY FOR NEXT PAR
4622 030362  022701  001600                     CMP     #1600,R1                ;REACHED A PAR7
4623 030366  001372                             BNE     PAR1                    ;BRANCH IF NOT
4624 030370  012720  177600                     MOV     #177600,(R0)+           ;INIT PAR7
4625 030374  000207                             RTS     PC                      ;RETURN
4626                                    ;
4627                                    ;TIME OUT ROUTINE
4628                                    ;
4629 030376  005205                     ADDTRP: INC     R5                      ;INCREMENT TIME OUT FLAG
4630 030400  000002                             RTI                             ;RETURN
4631                                    ;
4632                                    ;MMU TRAP ROUTINE
4633                                    ;
4634 030402  023727  003032  000001     MMUTRP: CMP     FLAG,#1                 ;ARE WE EXPECTING AN ABORT
4635 030410  001401                             BEQ     1$                      ;YES GO ON
4636 030412  104002                             ERROR   +2              ;NO GO TO ERROR
4637 030414  010046                     1$:     MOV     R0, (SP)                ;SAVE CONTENTS OF REG 0
4638 030416  013700  177776                     MOV     @#177776,R0             ;SAVE A COPY OF PSW
4639 030422  072027  177764                     ASH     # 14,R0                 ;LOOK AT BITS<15:14>
4640 030426  020027  000002                     CMP     R0,#2                   ;WAS PS<15:14>=10
4641 030432  001001                             BNE     OK                      ;NO GO ON
4642 030434  000411                             BR      NOTOK                   ;YES CHANGE BITS TO 00
```

```
4643 030436  013700  177776          OK:     MOV     @#177776,R0             ;SAVE A COPY OF PSW
4644 030442  072027  000002                  ASH     #2,R0                   ;LOOK AT BITS<13:12>
4645 030446  072027  177764                  ASH     # 14,R0                 ;
4646 030452  020027  000002                  CMP     R0,#2                   ;WAS PS<13:12>=10
4647 030456  001002                          BNE     OK1                     ;NO GO ON
4648 030460  005066  000004          NOTOK:  CLR     4(SP)                   ;CLEAR ILLEGAL MODE FFROM OLD PSW
4649 030464  013737  177572  003044  OK1:    MOV     @#177572,SAVMR0         ;SAVE A COPY OF MMR0
4650 030472  013737  177574  003046          MOV     @#177574,SAVMR1         ;SAVE A COPY OF MMR1
4651 030500  013737  177576  003050          MOV     @#177576,SAVMR2         ;SAVE A COPY OF MMR2
4652 030506  005037  177572                  CLR     @#177572                ;CLEAR ABORT BITS AND TURN MMU OFF
4653 030512  005037  003032                  CLR     FLAG                    ;CLEAR MMU ABORT FLAG
4654 030516  012600                          MOV     (SP)+,R0                ;RESTORE ORIGINAL CONTENTS OF REG 0
4655 030520  000002                          RTI                             ;RETURN
```

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

```
4707                            ;******************************************************************************
4708                            ; DELAY   INTERRUPT TIMER
4709                            ;
4710                            ; This test is called by the interrupt receiver and transmitter
4711                            ;
4712                            ;
4713                            ;-- --          --      -     - -  --      ----    ---
4714 030610                    DELAY:
4715 030610  013746  000004                    MOV     @#4, (SP)       ;SAVE
4716 030614  013746  000006                    MOV     @#6,-(SP)       ;    OLD
4717 030620  010046                            MOV     R0, (SP)        ;      STACK
4718 030622  010146                            MOV     R1, (SP)        ;           AND REGS
4719
4720 030624  012737  030652  000004            MOV     #2$,@#4         ;SET TRAP
4721 030632  106737  000006                    MFPS    @#6             ;PUT PSW ON PS / SAME PRIORITY
4722
4723 030636  012700  000001                    MOV     #1.,R0          ;SET OUTER LOOP COUNT
4724
4725 030642  012701  011610            3$:     MOV     #5000.,R1       ;SET INNER LOOP 1/200TH
4726
4727 030646  005737  160000            1$:     TST     @#160000        ;TAKES ABOUT 10u SEC
4728
4729 030652  022626                    2$:     CMP     (SP)+,(SP)+     ;CLEAN UP STACK
4730 030654  077104                            SOB     R1,1$           ;DO INNER LOOP
4731 030656  077007                            SOB     R0,3$           ;DO OUTER LOOP
4732
4733 030660  012601                            MOV     (SP)+,R1        ;RESTORE
4734 030662  012600                            MOV     (SP)+,R0        ;       WHAT
4735 030664  012637  000006                    MOV     (SP)+,@#6       ;          WE
4736 030670  012637  000004                    MOV     (SP)+,@#4       ;           SAVED
4737
4738 030674  000207                            RTS     PC
4739
```

```
 4741                                   ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 4742                                   ; SETMMU   SET UP THE MMU REGISTERS
 4743                                   ;
 4744                                   ; This test is called by the memory test to set up the PARS so
 4745                                   ; all of memory is accessed
 4746                                   ;
 4747                                   ;---  -----  -      -     --   - --   ----   - -- - -   --
 4748 030676                            SETMMU:                          ; SUBR SET UP MMU REGISTERS
 4749 030676  012737  000000  172340            MOV     #0,@#KIPAR0      ;  POINT TO PAGE 0 TO ITSELF
 4750 030704  012737  000200  172342            MOV     #200,@#KIPAR1    ;  POINT TO PAGE 1 TO ITSELF
 4751 030712  012737  000400  172344            MOV     #400,@#KIPAR2    ;  POINT TO PAGE 2 TO ITSELF
 4752 030720  012737  000600  172346            MOV     #600,@#KIPAR3    ;  POINT TO PAGE 3 TO ITSELF
 4753 030726  012737  001000  172350            MOV     #1000,@#KIPAR4   ;  POINT TO PAGE 4 TO ITSELF
 4754 030734  012737  001200  172352            MOV     #1200,@#KIPAR5   ;  POINT TO PAGE 5 TO ITSELF
 4755 030742  012737  001400  172354            MOV     #1400,@#KIPAR6   ;  POINT TO PAGE 6 TO ITSELF
 4756 030750  012737  177600  172356            MOV     #177600,@#KIPAR7;  POINT I/O PAGE TO IT'S PLACE IN 22 BIT
 4757 030756  012702  000010                    MOV     #10,R2           ;  BGNDO
 4758 030762  012701  172300                    MOV     #KIPDR0,R1       ;  : SET UP PDR'S TO 4K R/W
 4759 030766  012721  077406            10$:    MOV     #77406,(R1)+     ;  :
 4760 030772  077203                            SOB     R2,10$           ;  DOUNTIL WE HAVE INITIALIZED ALL OF THEM
 4761 030774  012737  000001  177572            MOV     #1,@#SR0         ;  ENABLE MEMORY MANAGEMENT
 4762 031002  012737  000020  172516            MOV     #BIT4,@#SR3      ;  ENABLE 22 BIT ADDRESSING
 4763 031010  000207                            RTS     PC               ; ENDSUBR SET UP MMU REGISTERS
```

```
                                        ;*****************************************************************
                                        ; INIMEM   FILL UP MEMORY
                                        ;
                                        ; This test is called by the Quick verify memory test. It init'alizes
                                        ; memory to look like the following.
                                        ;
                                        ;   Address     Data
                                        ;       -         -
                                        ;   0           377
                                        ;   1           0
                                        ;   2           0
                                        ;   3           377
                                        ;   4           0
                                        ;   5           0
                                        ;       .
                                        ;       .
                                        ;       .
                                        ;
031012                                  INIMEM:                                 ; SUBR INITIALIZE MEMORY
031012  012737  001600  172344                MOV     #1600,@#KIPAR2          ;  POINT KPAR2 TO LOW ADDRESS
031020  012701  040000                       MOV     #40000,R1               ;  SET VIRTUAL ADDRESS TO MAP THRU KPAR2
031024                                  10$:                                    ;  BGNDO
031024  112711  177777                       MOVB    #-1,(R1)                ;  : WRITE A  1 TO 0 OFFSET ADDRESS
031030  005201                               INC     R1                      ;  :
031032  020127  060000                       CMP     R1,#60000               ;  : IF WE HAVE PASSED THE PAGE BOUNDARY
031036  103413                               BLO     20$                     ;  : : THEN
031040  062737  000200  172344               ADD     #200,@#KIPAR2           ;  : :  POINT KPAR2 TO A NEW PAGE
031046  042701  160000                       BIC     #160000,R1              ;  : :  CLEAR OUT THE PAGE POINTER
031052  052701  040000                       BIS     #40000,R1               ;  : :  POINT VIRTUAL ADDRESS TO KPAR2
031056  023727  172344  020000               CMP     @#KIPAR2,#20000         ;  : ENDIF
031064  001443                               BEQ     40$                     ;  DOUNTIL WE HAVE READ ALL ADDRESSES
031066  112711  000000                20$:    MOVB    #0,(R1)                 ;  : WRITE A 0 TO 1 OFFSET ADDRESS
031072  005201                               INC     R1
031074  020127  060000                       CMP     R1,#60000               ;  : IF WE HAVE PASSED THE PAGE BOUNDARY
031100  103415                               BLO     30$                     ;  : : THEN
031102  062737  000200  172344               ADD     #200,@#KIPAR2           ;  : :  POINT KPAR2 TO A NEW PAGE
031110  042701  160000                       BIC     #160000,R1              ;  : :  CLEAR OUT THE PAGE POINTER
031114  052701  040000                       BIS     #40000,R1               ;  : :  POINT VIRTUAL ADDRESS TO KPAR2
031120  023727  172344  020000               CMP     @#KIPAR2,#20000         ;  : ENDIF
031126  001422                               BEQ     40$                     ;  DOUNTIL WE HAVE READ ALL ADDRESSES
031130  112711  000000                       MOVB    #0,(R1)                 ;  : WRITE A 0 TO 2 OFFSET ADDRESS
031134  005201                         30$:  INC     R1                      ;  : BUMP UP ADDRESS BY 3
031136  020127  060000                       CMP     R1,#60000               ;  : IF WE HAVE PASSED THE PAGE BOUNDARY
031142  103730                               BLO     10$                     ;  : : THEN
031144  062737  000200  172344               ADD     #200,@#KIPAR2           ;  : :  POINT KPAR2 TO A NEW PAGE
031152  042701  160000                       BIC     #160000,R1              ;  : :  CLEAR OUT THE PAGE POINTER
031156  052701  040000                       BIS     #40000,R1               ;  : :  POINT VIRTUAL ADDRESS TO KPAR2
031162  023727  172344  020000               CMP     @#KIPAR2,#20000         ;  : ENDIF
031170  001401                               BEQ     40$                     ;  DOUNTIL WE HAVE READ ALL ADDRESSES
031172  000714                               BR      10$                     ;
031174                                  40$:
031174  000207                         50$:  RTS     PC                      ;  ENDIF
                                                                            ; ENDSUBR INITIALIZE MEMORY
```

```
                                        ;*************************************************************
                                        ; RWTMEM   READ OR WRITE TO THE MEMORY
                                        ;
                                        ; This test 's 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
                                        ; --            ---   - -   -                  - ---  - - - -
 031176                                 RWTMEM:                                         ; SUBR READ WRITE MEMORY
 031176  012737  001600  172344                 MOV     #1600,@#KIPAR2                  ;  POINT TO KPAR2 TO LOW ADDRESS
 031204  012701  040000                         MOV     #40000,R1                       ;  POINT TO PAGE 2 VIRTUAL ADDRESS
 031210  012702  000003                         MOV     #3,R2                           ;  WE ARE BUMPING UP BY 3
 031214  160402                                 SUB     R4,R2                           ;  SUBTRACT OFF ADDRESS INCREMENT
 031216                                 10$:                                            ;  BGNDO
 031216  060401                                 ADD     R4,R1                           ;  : GET ADDRESS TO LOOK AT
 031220  020127  060000                         CMP     R1,#60000                       ;  : IF WE HAVE PASSED THE PAGE BOUNDARY
 031224  103413                                 BLO     15$                             ;  : : THEN
 031226  062737  000200  172344                 ADD     #200,@#KIPAR2                   ;  : :  POINT KPAR2 TO A NEW PAGE
 031234  042701  160000                         BIC     #160000,R1                      ;  : :  CLEAR OUT THE PAGE BITS
 031240  052701  040000                         BIS     #40000,R1                       ;  : :  SET THEM BACK TO PAGE 2
 031244  023727  172344  020000                 CMP     @#KIPAR2,#20000                 ;  : ENDIF
 031252  001435                                 BEQ     40$                             ;  :
 031254  005703                         15$:    TST     R3                              ;  : IF THE OPERATION IS A READ
 031256  001012                                 BNE     20$                             ;  : : THEN
 031260  120511                                 CMPB    R5,(R1)                         ;  : :  IF CONTENTS <> EXPECTED PATTERN
 031262  001411                                 BEQ     30$                             ;  : :  : THEN
 031264  010537  001124                         MOV     R5,$GDDAT                       ;  : :  :     GET THE EXPECTED PATTERN
 031270  111137  001126                         MOVB    (R1),$BDDAT                     ;  : :  :     GET THE RECIEVED PATTERN
 031274  010137  001122                         MOV     R1,$BDADR                       ;  : :  :     GET THE VIRTUAL ADDRESS
 031300  104066                                 ERROR   +66                             ;  : :  :     ERROR IN MEMORY
 031302  000401                                 BR      30$                             ;  : :  ENDIF
 031304                                 20$:                                            ;  : : ELSE
 031304  110511                                 MOVB    R5,(R1)                         ;  : :  WRITE PATTERN TO MEMORY
 031306                                 30$:                                            ;  : ENDIF
 031306  060201                                 ADD     R2,R1                           ;  : ADD ON DIFFERENCE TO GET ADDRESS+3
 031310  020127  060000                         CMP     R1,#60000                       ;  : IF WE HAVE PASSED THE PAGE BOUNDARY
 031314  103740                                 BLO     10$                             ;  : : THEN
 031316  062737  000200  172344                 ADD     #200,@#KIPAR2                   ;  : :  POINT KPAR2 TO A NEW PAGE
 031324  042701  160000                         BIC     #160000,R1                      ;  : :  CLEAR OUT THE PAGE BITS
 031330  052701  040000                         BIS     #40000,R1                       ;  : :  SET PAGE BITS BACK TO PAGE 2
 031334  023727  172344  020000                 CMP     @#KIPAR2,#20000                 ;  : ENDIF
 031342  001401                                 BEQ     40$                             ;  DOUNTIL WE HAVE READ ALL ADDRESSES
 031344  000724                                 BR      10$                             ;
 031346  000207                         40$:    RTS     PC                              ; ENDSUBR READ WRITE_MEMORY
```

DMARD DATI THRU Q22BE

```
 4873                                   ;+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
 4874                                   ; RRMEM - READ THE MEMORY
 4875                                   ;
 4876                                   ; This test is called by the memory test wr'te to memory
 4877                                   ;
 4878                                   ; R4 = THE ADDRESS INCREMENT
 4879                                   ; R5 = THE EXPECTED PATTERN
 4880                                   ;-- ---- -- --- ---- ------ ---  - --  ---- -  ------  ---
 4881 031350                            RRMEM:                          ; SUBR READ LOCATIONS 0,1
 4882 031350  012737  001600  172344            MOV     #1600,@#KIPAR2  ;  LET PAR POINT TO LOW ADDRESS
 4883 031356  012701  040000                    MOV     #40000,R1       ;  POINT TO KPAR2 VIRTUAL ADDRESS
 4884 031362                            10$:                            ;  BGNDO
 4885 031362  111137  001126                    MOVB    (R1),$BDDAT     ;  : GET A COPY OF RECIEVED PATTERN
 4886 031366  122737  177777  001126            CMPB    #-1,$BDDAT      ;  : IF 0 OFFSET ADDRESS <> -1 THEN
 4887 031374  001406                            BEQ     15$             ;  :    ERROR IN MEMORY
 4888 031376  012737  177777  001124            MOV     #-1,$GDDAT      ;  : GET THE EXPECTED PATTERN
 4889 031404  010137  001122                    MOV     R1,$BDADR       ;  : GET THE VIRTUAL ADDRESS
 4890 031410  104066                            ERROR   +66             ;  :
 4891 031412  005201                    15$:    INC     R1              ;  : POINT TO 1 OFFSET ADDRESS
 4892 031414  020127  060000                    CMP     R1,#60000       ;  : IF WE HAVE PASSED THE PAGE BOUNDARY
 4893 031420  103413                            BLO     20$             ;  : : THEN
 4894 031422  062737  000200  172344            ADD     #200,@#KIPAR2   ;  : :  POINT KPAR2 TO A NEW PAGE
 4895 031430  042701  160000                    BIC     #160000,R1      ;  : :  CLEAR THE PAGE BITS
 4896 031434  052701  040000                    BIS     #40000,R1       ;  : :  POINT BACK TO PAGE 2
 4897 031440  023727  172344  020000            CMP     @#KIPAR2,#20000 ;  : ENDIF
 4898 031446  001435                            BEQ     40$             ;  :
 4899 031450  111137  001126            20$:    MOVB    (R1),$BDDAT     ;  : GET THE RECIEVED PATTERN
 4900 031454  122737  177777  001126            CMPB    #-1,$BDDAT      ;  : IF 1 OFFSET ADDRESS <>  1 THEN
 4901 031462  001406                            BEQ     30$             ;  :    ERROR IN MEMORY
 4902 031464  112737  177777  001124            MOVB    #-1,$GDDAT      ;  : GET THE EXPECTED PATTERN
 4903 031472  011137  001122                    MOV     (R1),$BDADR     ;  : GET THE VIRTUAL ADDRESS
 4904 031476  104066                            ERROR   +66             ;  :
 4905 031500  062701  000002            30$:    ADD     #2,R1           ;  : ADD 2 TO ADDRESS TO GET ADDRESS + 3
 4906 031504  020127  060000                    CMP     R1,#60000       ;  : IF WE HAVE PASSED THE PAGE BOUNDARY
 4907 031510  103724                            BLO     10$             ;  : : THEN
 4908 031512  062737  000200  172344            ADD     #200,@#KIPAR2   ;  : :  POINT KPAR2 TO A NEW PAGE
 4909 031520  042701  160000                    BIC     #160000,R1      ;  : :  CLEAR THE PAGE BITS
 4910 031524  052701  040000                    BIS     #40000,R1       ;  : :  POINT BACK TO PAGE 2
 4911 031530  023727  172344  020000            CMP     @#KIPAR2,#20000 ;  : ENDIF
 4912 031536  001401                            BEQ     40$             ;  DOUNTIL WE HAVE READ ALL ADDRESSES
 4913 031540  000710                            BR      10$             ;
 4914 031542  000207                    40$:    RTS     PC              ; ENDSUBR READ LOCATIONS 0,1
 4915
```

```
                                        ;
                                        ;$$$
                                        ;
                                        ; SUBROUTINE   DETERMINE FLOATING POINT ACCELERATOR (DETFPA)
                                        ;
                                        ; THIS SUBROUTINE IS CALLED IF AN ERROR IS DETECTED DURING EXECUTION OF THE
                                        ; FLOATING POINT TESTS.
                                        ; IT DETERMINES WHEATHER OR NOT THE FLOATING POINT ACCELERATOR CHIP OPTION
                                        ; IS PRESENT ON THE CPU BOARD AND PRINTS THE APPROPRIATE ERROR MESSAGE.
                                        ; THIS DETERMINATION IS MADE BASED ON THE "FPA AVAILABLE" FLAG, BIT 8
                                        ; OF THE MAINTENANCE REGISTER AT LOCATION 17777750. IF THE FPA BIT IS SET
                                        ; THEN THE FLOATING POINT ACCELERATOR CHIP IS INSTALLED ON THE CPU BOARD AND
                                        ; AN ERROR MESSAGE IS PRINTED WHICH STATES THAT THE FLOATING POINT ERROR IS
                                        ; DUE TO THIS CHIP. OTHERWISE, THE J11 IS BLAMED FOR THE FLOATING POINT ERROR.
                                        ;
                                        ;$$$
                                        ;
                                        ; CALLED BY: CALL       @#DETFPA ;$$$
                                        ;
                                        ; INPUTS: NONE                   ;$$$
                                        ;
                                        ; OUTPUTS: ERROR MESSAGES        ;$$$
                                        ;
031544                                  DETFPA:
031544  104401  031552                          TYPE    ,J11FLT         ; $$$
031550  000207                                  RTS     PC              ; $$$
                                        ;
031552      105     122     122         J11FLT: .ASCIZ  /ERROR DETECTED IN J11 FLOATING POINT PROCESSOR./ ; $$$
031555      117     122     040
031560      104     105     124
031563      105     103     124
031566      105     104     040
031571      111     116     040
031574      112     061     061
031577      040     106     114
031602      117     101     124
031605      111     116     107
031610      040     120     117
031613      111     116     124
031616      040     120     122
031621      117     103     105
031624      123     123     117
031627      122     056     000
                                                .EVEN                   ; $$$
                                        ;
```

```
4949
4950
4951                                    .SBTTL  END OF PASS ROUTINE
                                        ;;*****************************************************************
                                        ;*INCREMENT THE PASS NUMBER ($PASS)
                                        ;*INDICATE END-OF PROGRAM AFTER 1 PASSES THRU THE PROGRAM
                                        ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
                                        ;*IF THERES A MONITOR GO TO IT
                                        ;*IF THERE ISN'T JUMP TO LOOP
     031632                             $EOP:
     031632  032737  0' :00  000052             BIT #BIT06,@#52
     031640  001030                             BNE $GET42
     031642  004537  017724                     jsr r5,v`reop
     031646  005037  001102                     CLR     $TSTNM          ;;ZERO THE TEST NUMBER
     031652  005037  001164                     CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
     031656  005237  001206                     INC     $PASS           ;;INCREMENT THE PASS NUMBER
     031662  042737  100000  001206           BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
     031670  005327                           DEC     (PC)+           ;;LOOP?
     031672  000001                     $EOPCT: .WORD   1
     031674  003022                             BGT     $DOAGN          ;;YES
     031676  012737                             MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
     031700  000001                     $ENDCT: .WORD   1
     031702  031672                             $EOPCT
     031704  104401  031751                     TYPE    ,$ENDMG         ;;TYPE "END PASS #"
     031710  013746  001206                     MOV     $PASS, (SP)     ;;SAVE $PASS FOR TYPEOUT
     031714  104405                             TYPDS                   ;;GO TYPE  DECIMAL ASCII WITH SIGN
     031716  104401  031746                     TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
     031722  013700  000042             $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
     031726  001405                             BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
     031730  000005                             RESET                   ;;CLEAR THE WORLD
     031732  004710                     $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
     031734  000240                             NOP                     ;;SAVE ROOM
     031736  000240                             NOP                     ;;FOR
     031740  000240                             NOP                     ;;ACT11
     031742                             $DOAGN:
     031742  000137                             JMP     @(PC)+          ;;RETURN
     031744  005064                     $RTNAD: .WORD   LOOP
     031746     377     377     000     $ENULL: .BYTE    1,-1,0         ;;NULL CHARACTER STRING
     031751     015     012     105     $ENDMG: .ASCIZ  <15><12>/END PASS #/
     031754     116     104     040
     031757     120     101     123
     031762     123     040     043
     031765     000
4952                                    .SBTTL  SCOPE HANDLER ROUTINE
                                        ;;*****************************************************************
                                        ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
                                        ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
                                        ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
                                        ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                        ;*SW14=1         LOOP ON TEST
                                        ;*SW11=1         INHIBIT ITERATIONS
                                        ;*SW09=1         LOOP ON ERROR
                                        ;*SW08=1         LOOP ON TEST IN SWR<5:0>
                                        ;*CALL
                                        ;*      SCOPE           ;;SCOPE=IOT
     031766                             $SCOPE:
     031766  104407                             CKSWR                   ;;TEST  FOR CHANGE IN SOFT SWR
```

```
031770  052737  001000  177520          BIS #1000,BCSR ;ENABLE
031776  032777  040000  147134  1$:     BIT     @#BIT14,@SWR    ;;LOOP ON PRESENT TEST?
032004  001117                          BNE     $OVER           ;;YES IF SW14=1
                                ;*****START OF CODE FOR THE XOR TESTER*****
032006  000416                $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR' TESTER CHANGE
                                                                ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
032010  013746  000004                  MOV     @#ERRVEC, (SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
032014  012737  032034  000004          MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
032022  005737  177060                  TST     @#177060        ;;TIME OUT ON XOR?
032026  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
032032  000466                          BR      $SVLAD          ;;GO TO THE NEXT TEST
032034  022626                  5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
032036  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
032042  000426                          BR      7$              ;;LOOP ON THE PRESENT TEST
032044                          6$:;*****END OF CODE FOR THE XOR TESTER*****
032044  032777  000400  147066          BIT     @#BIT08,@SWR    ;;LOOP ON SPEC. TEST?
032052  001407                          BEQ     2$              ;;BR IF NO
032054  017746  147060                  MOV     @SWR, (SP)      ;;SET DESIRED TEST NUM. FROM SWR
032060  042716  000300                  BIC     #$SWRMK,(SP)    ;;STRIP AWAY UNDESIRED BITS
032064  122637  001102                  CMPB    (SP)+,$TSTNM    ;;ON THE RIGHT TEST?
032070  001465                          BEQ     $OVER           ;;BR IF YES
032072  105737  001103          2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
032076  001421                          BEQ     3$              ;;BR IF NO
032100  123737  001115  001103          CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
032106  101015                          BHI     3$              ;;BR IF NO
032110  032777  001000  147022          BIT     @#BIT09,@SWR    ;;LOOP ON ERROR?
032116  001404                          BEQ     4$              ;;BR IF NO
032120  013737  001110  001106  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
032126  000446                          BR      $OVER
032130  105037  001103          4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
032134  005037  001164                  CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
032140  000415                          BR      1$              ;;ESCAPE TO THE NEXT TEST
032142  032777  004000  146770  3$:     BIT     @#BIT11,@SWR    ;;INHIBIT ITERATIONS?
032150  001011                          BNE     1$              ;;BR IF YES
032152  005737  001206                  TST     $PASS           ;;IF FIRST PASS OF PROGRAM
032156  001406                          BEQ     1$              ;;      INHIBIT ITERATIONS
032160  005237  001104                  INC     $ICNT           ;;INCREMENT ITERATION COUNT
032164  023737  001164  001104          CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
032172  002024                          BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
032174  012737  000001  001104  1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
032202  013 37  032266  001164          MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
032210  105237  001102          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
032214  113737  001102  001204          MOVB    $TSTNM,$TESTN   ;;SET TEST NUMBER IN APT MAILBOX
032222  011637  001106                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
032226  011637  001110                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
032232  005037  001166                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
032236  112737  000001  001115          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
032244  013777  001102  146670  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
032252  013716  001106                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
032256  042737  001000  177520          BIC #1000,BCSR ;DISABLE
032264  000002                          RTI
032266  000001                  $MXCNT: 1                       ;;MAX. NUMBER OF ITERATIONS
4953                            .SBTTL  ERROR HANDLER ROUTINE
                                ;;**************************************************************
                                ;*THIS ROUTINE WILL INCREMENT THE ERROR FLAG AND THE ERROR COUNT,
                                ;*SAVE THE ERROR ITEM NUMBER AND THE ADDRESS OF THE ERROR CALL
                                ;*AND GO TO ERTYPE ON ERROR
```

ERROR HANDLER ROUTINE

```
                                ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
                                ;*SW15=1         HALT ON ERROR
                                ;*SW13=1         INHIBIT ERROR TYPEOUTS
                                ;*SW10=1         BELL ON ERROR
                                ;*SW09=1         LOOP ON ERROR
                                ;*CALL
                                ;*      ERROR   +N      ;;ERROR=EMT AND N ERROR ITEM NUMBER
032270                          $ERROR:
032270  005737  004154                  TST     UQUIET          ;;TEST FOR USER-QUIET MODE
032274  001403                          BEQ     9$              ;;BRANCH IF FIELD SERVICE MODE
032276  005000                          CLR     R0              ;;IN CASE R0 HAS A #3 IN IT (^C)
032300  004737  032512                  JSR     PC,ABORT        ;;TEST FOR ABORT CONDITION
032304                          9$:
032304  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT SWR
032306  052737  001000  177520          BIS     #1000,BCSR              ;ENABLE HALT ON BREAK
032314  105237  001103          7$:     INCB    $ERFLG          ;;SET THE ERROR FLAG
032320  001775                          BEQ     7$              ;;DON'T LET THE FLAG GO TO ZERO
032322  013777  001102  146612          MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER AND ERROR FLAG
032330  032777  002000  146602          BIT     #BIT10,@SWR     ;;BELL ON ERROR?
032336  001402                          BEQ     1$              ;;NO   SKIP
032340  104401  001170                  TYPE    ,$BELL          ;;RING BELL
032344  005237  001112          1$:     INC     $ERTTL          ;;COUNT THE NUMBER OF ERRORS
032350  011637  001116                  MOV     (SP),$ERRPC     ;;GET ADDRESS OF ERROR INSTRUCTION
032354  162737  000002  001116          SUB     #2,$ERRPC
032362  117737  146530  001114          MOVB    @$ERRPC,$ITEMB  ;;STRIP AND SAVE THE ERROR ITEM CODE
032370  032777  020000  146542          BIT     #BIT13,@SWR     ;;SKIP TYPEOUT IF SET
032376  001004                          BNE     20$             ;;SKIP TYPEOUTS
032400  004737  027030                  JSR     PC,ERTYPE       ;;GO TO USER ERROR ROUTINE
032404  104401  001175                  TYPE    ,$CRLF
032410                          20$:
032410  122737  000001  001220          CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
032416  001007                          BNE     2$              ;;NO,SKIP APT ERROR REPORT
032420  113737  001114  032432          MOVB    $ITEMB,21$      ;;SET ITEM NUMBER AS ERROR NUMBER
032426  004737  032670                  JSR     PC,$ATY4        ;;REPORT FATAL ERROR TO APT
032432     000          21$:    .BYTE   0
032433     000                  .BYTE   0
032434  000777          22$:    BR      22$             ;;APT ERROR LOOP
032436  005777  146476          2$:     TST     @SWR            ;;HALT ON ERROR
032442  100002                          BPL     3$              ;;SKIP IF CONTINUE
032444  000000                          HALT                    ;;HALT ON ERROR!
032446  104407                          CKSWR                   ;;TEST FOR CHANGE IN SOFT-SWR
032450  032777  001000  146462  3$:     BIT     #BIT09,@SWR     ;;LOOP ON ERROR SWITCH SET?
032456  001402                          BEQ     4$              ;;BR IF NO
032460  013716  001110                  MOV     $LPERR,(SP)     ;;FUDGE RETURN FOR LOOPING
032464  005737  001166          4$:     TST     $ESCAPE         ;;CHECK FOR AN ESCAPE ADDRESS
032470  001402                          BEQ     5$              ;;BR IF NONE
032472  013716  001166                  MOV     $ESCAPE,(SP)    ;;FUDGE RETURN ADDRESS FOR ESCAPE
032476                          5$:
032476  022737  031732  000042          CMP     #$ENDAD,@#42    ;;ACT 11 AUTO ACCEPT?
032504  001001                          BNE     6$              ;;BRANCH IF NO
032506  000000                          HALT                    ;;YES
032510                          6$:
032510  000002                          RTI                     ;;RETURN
                                .SBTTL  ABORT ROUTINE FOR LCP/ORION UFD MODE
032512  005737  004152          ABORT:  TST     UFDFLG          ;TEST FOR USER FRIENDLY MODE
032516  001454                          BEQ     NOABRT          ;IF NOT UFD THEN CONTINUE NORMAL OPERATION
032520  020027  000032                  CMP     R0,#32          ;IS IT A ^Z ?
```

ABORT ROUTINE FOR LCP/ORION UFD MODE

```
032524  001443                          BEQ     ABORTZ          ;JUST GO BACK TO CHAIN IF IT IS (NO ERROR)
032526  020027  000003                   CMP     R0,#3           ;IS IS A ↑C ?
032532  001404                          BEQ     ABORTC          ;BR TO LOAD ↑C ON XXDP+ STACK (NO ERROR)
032534  005737  004154                   TST     UQUIET          ;TEST FOR USER-QUIET MODE
032540  001443                          BEQ     NOABRT          ;IF FIELD-SERVICE MODE, CONTINUE NORMAL OPERATION
                                                                ; BECAUSE FIELD-SERVICE MODE DOES NOT QUIT ON ERROR
032542  000422                          BR      ABORTE          ;SET DRSERR THEN LEAVE
032544  013737  004146  000030  ABORTC: MOV     SAV30,30        ;RESTORE EMT LOCATION (30)
032552  013737  004150  000032          MOV     SAV32,32        ;RESTORE EMT PRIORITY LOCATION (32)
032560  104043                          EMT     +43             ;GET XXDP STACK LOC. INTO R0 FROM MONITOR
032562  005720                  1$:     TST     (R0)+           ;FIND END OF STACK
032564  001376                          BNE     1$
032566  112760  000057  177777          MOVB    #'/, 1(R0)      ;LOAD SLASH OVER ZERO
032574  112720  000136                  MOVB    #'↑,(R0)+       ;LOAD UPARROW
032600  112720  000103                  MOVB    #'C,(R0)+       ;LOAD C
032604  105010                          CLRB    (R0)            ;MAKE NEW END TO STACK
032606  000412                          BR      ABORTZ          ;NOW LEAVE
032610  013737  004146  000030  ABORTE: MOV     SAV30,30        ;RESTORE EMT LOCATION (30)
032616  013737  004150  000032          MOV     SAV32,32        ;RESTORE EMT PRIORITY LOCATION (32)
032624  104042                          EMT     +42             ;GET DCA LOCATION INTO R0 FROM MONITOR
032626  012760  177777  000042          MOV     #-1,42(R0)      ;SET A -1 INTO LOCATION DRSERR IN MONITOR
032634  013700  000042  ABORTZ: MOV     @#42,R0         ;AND PUT THE MONITOR RETURN ADDRESS IN R0
032640  005037  000042                  CLR     @#42            ;CLEAR MONITOR RETURN FLAG
032644  000137  031732                  JMP     $ENDAD          ;RETURN TO MONITOR DO NOT PUSH STACK HERE
032650  000207          NOABRT: RTS     PC              ;IF NOTUFD RETURN TO MAINLINE
4954                    .SBTTL  APT COMMUNICATIONS ROUTINE
                        ;;*****************************************************************
032652  112737  000001  033116  $ATY1:  MOVB    #1,$FFLG        ;;TO REPORT FATAL ERROR
032660  112737  000001  033114  $ATY3:  MOVB    #1,$MFLG        ;;TO TYPE A MESSAGE
032666  000403                          BR      $ATYC
032670  112737  000001  033116  $ATY4:  MOVB    #1,$FFLG        ;;TO ONLY REPORT FATAL ERROR
032676                  $ATYC:
032676  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
032700  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
032702  105737  033114                  TSTB    $MFLG           ;;SHOULD TYPE A MESSAGE?
032706  001450                          BEQ     5$              ;;IF NOT:  BR
032710  122737  000001  001220          CMPB    #APTENV,$ENV    ;;OPERATING UNDER APT?
032716  001031                          BNE     3$              ;;IF NOT:  BR
032720  132737  000100  001221          BITB    #APTSPOOL,$ENVM ;;SHOULD SPOOL MESSAGES?
032726  001425                          BEQ     3$              ;;IF NOT-  BR
032730  017600  000004                  MOV     @4(SP),R0       ;;GET MESSAGE ADDR.
032734  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
032742  005737  001200          1$:     TST     $MSGTYPE        ;;SEE IF DONE W/ LAST XMISSION?
032746  001375                          BNE     1$              ;;IF NOT:  WAIT
032750  010037  001214                  MOV     R0,$MSGAD       ;;PUT ADDR IN MAILBOX
032754  105720                  2$:     TSTB    (R0)+           ;;FIND END OF MESSAGE
032756  001376                          BNE     2$
032760  163700  001214                  SUB     $MSGAD,R0       ;;SUB START OF MESSAGE
032764  006200                          ASR     R0              ;;GET MESSAGE LNGTH IN WORDS
032766  010037  001216                  MOV     R0,$MSGLGT      ;;PUT LENGTH IN MAILBOX
032772  012737  000004  001200          MOV     #4,$MSGTYPE     ;;TELL APT TO TAKE MSG.
033000  000413                          BR      5$
033002  017637  000004  033026  3$:     MOV     @4(SP),4$       ;;PUT MSG ADDR IN JSR LINKAGE
033010  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDRESS
033016  013746  177776                  MOV     177776,-(SP)    ;;PUSH 177776 ON STACK
033022  004737  033120                  JSR     PC,$TYPE        ;;CALL TYPE MACRO
033026  000000          4$:     .WORD   0
```

```
033030                                  5$:
033030  105737  033116          10$:    TSTB    $FFLG           ;;SHOULD REPORT FATAL ERROR?
033034  001416                          BEQ     12$             ;;IF NOT:  BR
033036  005737  001220                  TST     $ENV            ;;RUNNING UNDER APT?
033042  001413                          BEQ     12$             ;;IF NOT:  BR
033044  005737  001200          11$:    TST     $MSGTYPE        ;;FINISHED LAST MESSAGE?
033050  001375                          BNE     11$             ;;IF NOT:  WAIT
033052  017637  000004  001202          MOV     @4(SP),$FATAL   ;;GET ERROR #
033060  062766  000002  000004          ADD     #2,4(SP)                ;;BUMP RETURN ADDR.
033066  005237  001200                  INC     $MSGTYPE        ;;TELL APT TO TAKE ERROR
033072  105037  033116          12$:    CLRB    $FFLG           ;;CLEAR FATAL FLAG
033076  105037  033115                  CLRB    $LFLG           ;;CLEAR LOG FLAG
033102  105037  033114                  CLRB    $MFLG           ;;CLEAR MESSAGE FLAG
033106  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
033110  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
033112  000207                          RTS     PC              ;;RETURN
033114     000          $MFLG:  .BYTE   0               ;;MESSG. FLAG
033115     000          $LFLG:  .BYTE   0               ;;LOG FLAG
033116     000          $FFLG:  .BYTE   0               ;;FATAL FLAG
                                .EVEN
            000200      APTSIZE=200
            000001      APTENV=001
            000100      APTSPOOL=100
            000040      APTCSUP=040
4955                            .SBTTL  TYPE ROUTINE
                        ;;******************************************************************
                        ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
                        ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
                        ;*NOTE1:         $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
                        ;*NOTE2:         $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
                        ;*NOTE3:         $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
                        ;*
                        ;*CALL:
                        ;*1) USING A TRAP INSTRUCTION
                        ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
                        ;*OR
                        ;*      TYPE
                        ;*      MESADR
                        ;*
033120  105737  001157  $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
033124  100002                 BPL     1$              ;;BR IF YES
033126  000000                 HALT                    ;;HALT HERE IF NO TERMINAL
033130  000430                 BR      3$              ;;LEAVE
033132  010046          1$:    MOV     R0,-(SP)        ;;SAVE R0
033134  017600  000002         MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
033140  122737  000001  001220 CMPB    #APTENV,$ENV    ;;RUNNING IN APT MODE
033146  001011                 BNE     62$             ;;NO,GO CHECK FOR APT CONSOLE
033150  132737  000100  001221 BITB    #APTSPOOL,$ENVM ;;SPOOL MESSAGE TO APT
033156  001405                 BEQ     62$             ;;NO,GO CHECK FOR CONSOLE
033160  010037  033170         MOV     R0,61$          ;;SETUP MESSAGE ADDRESS FOR APT
033164  004737  032660         JSR     PC,$ATY3        ;;SPOOL MESSAGE TO APT
033170  000000          61$:   .WORD   0               ;;MESSAGE ADDRESS
033172  132737  000040  001221 62$:    BITB    #APTCSUP,$ENVM  ;;APT CONSOLE SUPPRESSED
033200  001003                 BNE     60$             ;;YES,SKIP TYPE OUT
033202  112046          2$:    MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
033204  001005                 BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
033206  005726                 TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
```

```
* PE ROUTINE

033210  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
033212  062716  000002            3$:    ADD     #2,(SP)         ;;ADJUST RETURN PC
033216  000002                           RTI                     ;;RETURN
033220  122716  000011            4$:    CMPB    #HT,(SP)        ;;BRANCH IF <HT>
033224  001430                           BEQ     8$
033226  122716  000200                   CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
033232  001006                           BNE     5$
033234  005726                           TST     (SP)+           ;;POP  <CR><LF> EQUIV
033236  104401                           TYPE                    ;;TYPE A CR AND LF
033240  001175                           $CRLF
033242  105037  033450                   CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
033246  000755                           BR      2$              ;;GET NEXT CHARACTER
033250  004737  033332            5$:    JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
033254  123726  001156            6$:    CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
033260  001350                           BNE     2$              ;;IF NO GO GET NEXT CHAR.
033262  013746  001154                   MOV     $NULL, (SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                 ;;AND THE NULL CHAR.
033266  105366  000001            7$:    DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
033272  002770                           BLT     6$              ;;BR IF NO- GO POP THE NULL OFF OF STACK
033274  004737  033332                   JSR     PC,$TYPEC       ;;GO TYPE A NULL
033300  105337  033450                   DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
033304  000770                           BR      7$              ;;LOOP
                                ;HORIZONTAL TAB PROCESSOR
033306  112716  000040            8$:    MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
033312  004737  033332            9$:    JSR     PC,$TYPEC       ;;TYPE A SPACE
033316  132737  000007  033450           BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
033324  001372                           BNE     9$              ;;TAB STOP
033326  005726                           TST     (SP)+           ;;POP SPACE OFF STACK
033330  000724                           BR      2$              ;;GET NEXT CHARACTER
033332                          $TYPEC:
033332  105777  145606                   TSTB    @$TKS           ;;CHAR IN KYBD BUFFER?                  ;MJD001
033336  100022                           BPL     10$             ;;BR IF NOT                             ;MJD001
033340  017746  145602                   MOV     @$TKB, (SP)     ;;GET CHAR                              ;MJD001
033344  042716  177600                   BIC     #177600,(SP)    ;;STRIP EXTRANEOUS BITS                 ;MJD001
033350  122716  000023                   CMPB    #$XOFF,(SP)     ;;WAS CHAR XOFF                         ;MJD001
033354  001012                           BNE     102$            ;;BR IF NOT                             ;MJD001
033356                          101$:                                                                   ;MJD001
033356  105777  145562                   TSTB    @$TKS           ;;WAIT FOR CHAR                         ;MJD001
033362  100375                           BPL     101$                                                    ;MJD001
033364  117716  145556                   MOVB    @$TKB,(SP)      ;;GET CHAR                              ;MJD001
033370  042716  177600                   BIC     #177600,(SP)    ;;STRIP IT                              ;MJD001
033374  122716  000021                   CMPB    #$XON,(SP)      ;;WAS IT XON?                           ;MJD001
033400  001366                           BNE     101$            ;;BR IF NOT                             ;MJD001
033402                          102$:                                                                   ;MJD001
033402  005726                           TST     (SP)+           ;;FIX STACK                             ;MJD001
033404                          10$:                                                                    ;MJD001
033404  105777  145540                   TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
033410  100375                           BPL     10$                                                     ;MJD001
033412  116677  000002  145532           MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
033420  122766  000015  000002           CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
033426  001003                           BNE     1$              ;;BRANCH IF NO
033430  105037  033450                   CLRB    $CHARCNT        ;;YES -CLEAR CHARACTER COUNT
033434  000406                           BR      $TYPEX          ;;EXIT
033436  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
033444  001402                           BEQ     $TYPEX          ;;BRANCH IF YES
033446  105227                           INCB    (PC)+           ;;COUNT THE CHARACTER
033450  000000                  $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
```

```
      033452  000207                  $TYPEX: RTS     PC
 4956                                 .SBTTL  BINARY TO OCTAL (ASCII) AND TYPE
                                      ;;*****************************************************************
                                      ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6 DIGIT
                                      ;*OCTAL (ASCII) NUMBER AND TYPE IT.
                                      ;*$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
      033454  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
      033460  116637  000001  033677          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
      033466  112637  033701                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
      033472  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
      033476  000406                          BR      $TYPON
      033500  112737  000001  033677  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
      033506  112737  000006  033701          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
      033514  112737  000005  033676  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
      033522  010346                          MOV     R3,-(SP)        ;;SAVE R3
      033524  010446                          MOV     R4,-(SP)        ;;SAVE R4
      033526  010546                          MOV     R5,-(SP)        ;;SAVE R5
      033530  113704  033701                  MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
      033534  005404                          NEG     R4
      033536  062704  000006                  ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
      033542  110437  033700                  MOVB    R4,$OMODE       ;;SAVE IT FOR USE
      033546  113704  033677                  MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
      033552  016605  000012                  MOV     12(SP),R5       ;;PICKUP THE INPUT NUMBER
      033556  005003                          CLR     R3              ;;CLEAR THE OUTPUT WORD
      033560  006105                  1$:     ROL     R5              ;;ROTATE MSB INTO "C'
      033562  000404                          BR      3$              ;;GO DO MSB
      033564  006105                  2$:     ROL     R5              ;;FORM THIS DIGIT
      033566  006105                          ROL     R5
      033570  006105                          ROL     R5
      033572  010503                          MOV     R5,R3
      033574  006103                  3$:     ROL     R3              ;;GET LSB OF THIS DIGIT
      033576  105337  033700                  DECB    $OMODE          ;;TYPE THIS DIGIT?
      033602  100016                          BPL     7$              ;;BR IF NO
      033604  042703  177770                  BIC     #177770,R3      ;;GET RID OF JUNK
      033610  001002                          BNE     4$              ;;TEST FOR 0
      033612  005704                          TST     R4              ;;SUPPRESS THIS 0?
      033614  001403                          BEQ     5$              ;;BR IF YES
      033616  005204                  4$:     INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
      033620  052703  000060                  BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
```

BINARY TO OCTAL (ASCII) AND TYPE

```
033624  052703  000040          5$:     BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
033630  110337  033674                  MOVB    R3,8$           ;;SAVE FOR TYPING
033634  104401  033674                  TYPE    ,8$             ;;GO TYPE THIS DIGIT
033640  105337  033676          7$:     DECB    $OCNT           ;;COUNT BY 1
033644  003347                          BGT     2$              ;;BR IF MORE TO DO
033646  002402                          BLT     6$              ;;BR IF DONE
033650  005204                          INC     R4              ;;INSURE LAST DIGIT ISN T A BLANK
033652  000744                          BR      2$              ;;GO DO THE LAST DIGIT
033654  012605                  6$:     MOV     (SP)+,R5        ;;RESTORE R5
033656  012604                          MOV     (SP)+,R4        ;;RESTORE R4
033660  012603                          MOV     (SP)+,R3        ;;RESTORE R3
033662  016666  000002  000004          MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
033670  012616                          MOV     (SP)+,(SP)
033672  000002                          RTI                     ;;RETURN
033674     000                  8$:     .BYTE   0               ;;STORAGE FOR ASCII DIGIT
033675     000                          .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
033676     000                  $OCNT:  .BYTE   0               ;;OCTAL DIGIT COUNTER
033677     000                  $OFILL: .BYTE   0               ;;ZERO FILL SWITCH
033700  000000                  $OMODE: .WORD   0               ;;NUMBER OF DIGITS TO TYPE
 4957                           .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
                                ;;**************************************************************
                                ;*THIS ROUTINE IS USED TO CHANGE A 16 BIT BINARY NUMBER TO A 5 DIGIT
                                ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
                                ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
                                ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
                                ;*REPLACED WITH SPACES.
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
                                ;*      TYPDS                   ;;GO TO THE ROUTINE
033702                          $TYPDS:
033702  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
033704  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
033706  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
033710  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
033712  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
033714  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
033720  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
033724  100004                          BPL     1$              ;;BR IF INPUT IS POS.
033726  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
033730  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
033736  005000                  1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
033740  012703  034116                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
033744  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
033750  005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
033752  016001  034106                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
033756  160105                  3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
033760  002402                          BLT     4$              ;;BR IF DONE
033762  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
033764  000774                          BR      3$
033766  060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
033770  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
033772  001002                          BNE     5$              ;;FALL THROUGH IF 0
033774  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0 S?
033776  100407                          BMI     7$              ;;BR IF YES
034000  106316                  5$:     ASLB    (SP)            ;;MSD?
034002  103003                          BCC     6$              ;;BR IF NO
034004  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
```

```
034012  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
034016  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
034022  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
034024  005720                          TST     (R0)+           ;;JUST INCREMENTING
034026  020027  000010                  CMP     R0,#10          ;;CHECK THE TABLE INDEX
034032  002746                          BLT     2$              ;;GO DO THE NEXT DIGIT
034034  003002                          BGT     8$              ;;GO TO EXIT
034036  010502                          MOV     R5,R2           ;;GET THE LSD
034040  000764                          BR      6$              ;;GO CHANGE TO ASCII
034042  105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON ZERO?
034044  100003                          BPL     9$              ;;BR IF NO
034046  116663  177777  177776        MOVB    1(SP), 2(R3)    ;;YES  SET THE SIGN FOR TYPING
034054  105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
034056  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
034060  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
034062  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
034064  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
034066  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
034070  104401  034116                  TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
034074  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
034102  012616                          MOV     (SP)+,(SP)
034104  000002                          RTI                     ;;RETURN TO USER
034106  023420                  $DTBL:  10000.
034110  001750                          1000.
034112  000144                          100.
034114  000012                          10.
034116                          $DBLK:  .BLKW   4
 4958                           .SBTTL  TTY INPUT ROUTINE
                                ;;**********************************************************************
                                .ENABL  LSB
                                ;;**********************************************************************
                                ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
                                ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
                                ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
                                ;*WHEN OPERATING IN TTY FLAG MODE.
034126  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT SWR SELECTED?
034134  001074                          BNE     15$             ;;BRANCH IF NO
034136  105777  145002                  TSTB    @$TKS           ;;CHAR THERE?
034142  100071                          BPL     15$             ;;IF NO, DON'T WAIT AROUND
034144  117746  144776                  MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
034150  042716  177600                  BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
034154  022726  000007                  CMP     #7,(SP)+        ;;IS IT A CONTROL G?
034160  001062                          BNE     15$             ;;NO, RETURN TO USER
034162  123727  001134  000001          CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO MODE?
034170  001456                          BEQ     15$             ;;BRANCH IF YES
034172  104401  034663                  TYPE    ,$CNTLG         ;;ECHO THE CONTROL G (^G)
034176  104401  034670          $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
034202  013746  000176                  MOV     SWREG, (SP)     ;;SAVE SWREG FOR TYPEOUT
034206  104402                          TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
034210  104401  034701                  TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
034214  005046                  19$:    CLR     -(SP)           ;;CLEAR COUNTER
034216  005046                          CLR     -(SP)           ;;THE NEW SWR
034220  105777  144720          7$:     TSTB    @$TKS           ;;CHAR THERE?
034224  100375                          BPL     7$              ;;IF NOT TRY AGAIN
034226  117746  144714                  MOVB    @$TKB, (SP)     ;;PICK UP CHAR
034232  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
034236  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL U?
```

```
034242  001005                          BNE     10$             ;;BRANCH IF NOT
034244  104401  034656                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL U (^U)
034250  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
034254  000757                          BR      19$             ;;LET'S TRY IT AGAIN
034256  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
034262  001022                          BNE     16$             ;;BRANCH IF NO
034264  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
034270  001403                          BEQ     11$             ;;BRANCH IF YES
034272  016677  000002  144640          MOV     2(SP),@SWR      ;;SAVE NEW SWR
034300  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
034304  104401  001175          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
034310  123727  001135  000001          CMPB    $INTAG,#1       ;;RE ENABLE TTY KBD INTERRUPTS?
034316  001003                          BNE     15$             ;;BRANCH IF NOT
034320  012777  000100  144616          MOV     #100,@$TKS      ;;RE ENABLE TTY KBD INTERRUPTS
034326  000002          15$:    RTI                             ;;RETURN
034330  004737  033332          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
034334  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
034340  002420                          BLT     18$             ;;BRANCH IF YES
034342  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
034346  003015                          BGT     18$             ;;BRANCH IF YES
034350  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
034354  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
034360  001403                          BEQ     17$             ;;BRANCH IF YES
034362  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
034364  006316                          ASL     (SP)            ;;    CHAR OVER TO MAKE
034366  006316                          ASL     (SP)            ;;    ROOM FOR NEW ONE.
034370  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
034374  056616  177776                  BIS     2(SP),(SP)      ;;SET IN NEW CHAR
034400  000707                          BR      7$              ;;GET THE NEXT ONE
034402  104401  001174          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
034406  000720                          BR      20$             ;;SIMULATE CONTROL U
                                .DSABL  LSB
                                ;;*****************************************************************
                                ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
                                ;*CALL:
                                ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
                                ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
                                ;*                              ;;WITH PARITY BIT STRIPPED OFF
                                ;
034410  011646          $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
034412  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
034420  105777  144520          1$:     TSTB    @$TKS           ;;WAIT FOR
034424  100375                          BPL     1$              ;;A CHARACTER
034426  117766  144514  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
034434  042766  177600  000004          BIC     #^C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
034442  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
034450  001013                          BNE     3$              ;;BRANCH IF NO
034452  105777  144466          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
034456  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
034460  117746  144462                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
034464  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
034470  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL Q?
034474  001366                          BNE     2$              ;;IF NOT DISCARD IT
034476  000750                          BR      1$              ;;YES, RESUME
034500  026627  000004  000021  3$:     CMP     4(SP),#$XON     ;;IS IT A RANDOM XON?           ;RAN001
034506  001744                          BEQ     1$              ;;BRANCH IF YES                 ;RAN001
034510  026627  000004  000140          CMP     4(SP),#140      ;;IS IT UPPER CASE?
```

```
034516  002407                          BLT     4$                      ;;BRANCH IF YES
034520  026627  000004  000175          CMP     4(SP),#175              ;;IS IT A SPECIAL CHAR?
034526  003003                          BGT     4$                      ;;BRANCH IF YES
034530  042766  000040  000004          BIC     #40,4(SP)               ;;MAKE IT UPPER CASE
034536  000002                  4$:     RTI                             ;;GO BACK TO USER
                                ;;**************************************************************
                                ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                ;*CALL:
                                ;*      RDLIN                           ;;INPUT A STRING FROM THE TTY
                                ;*      RETURN HERE                     ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                ;*                                      ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
034540  010346                  $RDLIN: MOV     R3, (SP)                ;;SAVE R3
034542  012703  034646          1$:     MOV     #$TTYIN,R3              ;;GET ADDRESS
034546  022703  034656          2$:     CMP     #$TTYIN+8.,R3           ;;BUFFER FULL?
034552  101405                          BLOS    4$                      ;;BR IF YES
034554  104410                          RDCHR                           ;;GO READ ONE CHARACTER FROM THE TTY
034556  112613                          MOVB    (SP)+,(R3)              ;;GET CHARACTER
034560  122713  000177          10$:    CMPB    #177,(R3)               ;;IS IT A RUBOUT
034564  001003                          BNE     3$                      ;;SKIP IF NOT
034566  104401  001174          4$:     TYPE    ,$QUES                  ;;TYPE A '?'
034572  000763                          BR      1$                      ;;CLEAR THE BUFFER AND LOOP
034574  111337  034644          3$:     MOVB    (R3),9$                 ;;ECHO THE CHARACTER
034600  104401  034644                  TYPE    ,9$
034604  122723  000015                  CMPB    #15,(R3)+               ;;CHECK FOR RETURN
034610  001356                          BNE     2$                      ;;LOOP IF NOT RETURN
034612  105063  177777                  CLRB    1(R3)                   ;;CLEAR RETURN (THE 15)
034616  104401  001176                  TYPE    ,$LF                    ;;TYPE A LINE FEED
034622  012603                          MOV     (SP)+,R3                ;;RESTORE R3
034624  011646                          MOV     (SP),-(SP)              ;;ADJUST THE STACK AND PUT ADDRESS OF THE
034626  016666  000004  000002          MOV     4(SP),2(SP)             ;;      FIRST ASCII CHARACTER ON IT
034634  012766  034646  000004          MOV     #$TTYIN,4(SP)
034642  000002                          RTI                             ;;RETURN
034644     000                  9$:     .BYTE   0                       ;;STORAGE FOR ASCII CHAR. TO TYPE
034645     000                          .BYTE   0                       ;;TERMINATOR
034646                          $TTYIN: .BLKB   8.                      ;;RESERVE 8 BYTES FOR TTY INPUT
034656     136     125     015  $CNTLU: .ASCIZ  /↑U/<15><12>            ;;CONTROL "U"
034661     012     000
034663     136     107     015  $CNTLG: .ASCIZ  /↑G/<15><12>            ;;CONTROL "G"
034666     012     000
034670     015     012     123  $MSWR:  .ASCIZ  <15><12>/SWR = /
034673     127     122     040
034676     075     040     000
034701     040     040     116  $MNEW:  .ASCIZ  /  NEW = /
034704     105     127     040
034707     075     040     000
4959                            .SBTTL  READ AN OCTAL NUMBER FROM THE TTY
                                ;;**************************************************************
                                ;*THIS ROUTINE WILL READ AN OCTAL (ASCII) NUMBER FROM THE TTY AND
                                ;*CHANGE IT TO BINARY.
                                ;*CALL:
                                ;*      RDOCT                           ;;READ AN OCTAL NUMBER
                                ;*      RETURN HERE                     ;;LOW ORDER BITS ARE ON TOP OF THE STACK
                                ;*                                      ;;HIGH ORDER BITS ARE IN $HIOCT
034712  011646                  $RDOCT: MOV     (SP), (SP)              ;;PROVIDE SPACE FOR THE
034714  016666  000004  000002          MOV     4(SP),2(SP)             ;;INPUT NUMBER
034722  010046                          MOV     R0,-(SP)                ;;PUSH R0 ON STACK
034724  010146                          MOV     R1,-(SP)                ;;PUSH R1 ON STACK
```

READ AN OCTAL NUMBER FROM THE TTY

```
034726  010246                  MOV     R2, (SP)        ;;PUSH R2 ON STACK
034730  104411          1$:     RDLIN                   ;;READ AN ASCIZ LINE
034732  012600                  MOV     (SP)+,R0        ;;GET ADDRESS OF 1ST CHARACTER
034734  005001                  CLR     R1              ;;CLEAR DATA WORD
034736  005002                  CLR     R2
034740  112046          2$:     MOVB    (R0)+,-(SP)     ;;PICKUP THIS CHARACTER
034742  001412                  BEQ     3$              ;;IF ZERO GET OUT
034744  006301                  ASL     R1              ;;*2
034746  006102                  ROL     R2
034750  006301                  ASL     R1              ;;*4
034752  006102                  ROL     R2
034754  006301                  ASL     R1              ;;*8
034756  006102                  ROL     R2
034760  042716  177770          BIC     #^C7,(SP)       ;;STRIP THE ASCII JUNK
034764  062601                  ADD     (SP)+,R1        ;;ADD IN THIS DIGIT
034766  000764                  BR      2$              ;;LOOP
034770  005726          3$:     TST     (SP)+           ;;CLEAN TERMINATOR FROM STACK
034772  010166  000012          MOV     R1,12(SP)       ;;SAVE THE RESULT
034776  010237  035012          MOV     R2,$HIOCT
035002  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
035004  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
035006  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
035010  000002                  RTI                     ;;RETURN
035012  000000          $HIOCT: .WORD   0               ;;HIGH ORDER BITS GO HERE
  4960                  .SBTTL  TRAP DECODER
                        ;;**************************************************************
                        ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
                        ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
                        ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
                        ;*GO TO THAT ROUTINE.
035014  010046          $TRAP:  MOV     R0,-(SP)        ;;SAVE R0
035016  016600  000002          MOV     2(SP),R0        ;;GET TRAP ADDRESS
035022  005740                  TST      (R0)           ;;BACKUP BY 2
035024  111000                  MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
035026  006300                  ASL     R0              ;;POSITION FOR INDEXING
035030  016000  035050          MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
035034  000200                  RTS     R0              ;;GO TO ROUTINE
                        ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
035036  011646          $TRAP2: MOV     (SP), (SP)      ;;MOVE THE PC DOWN
035040  016666  000004  000002          MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
035046  000002                  RTI                     ;;RESTORE THE PSW
                        .SBTTL  TRAP TABLE
                        ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
                        ;*BY THE "TRAP" INSTRUCTION.
                        ;       ROUTINE
                        ;        ---
035050  035036          $TRPAD: .WORD   $TRAP2
035052  033120                  $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
035054  033500                  $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
035056  033454                  $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
035060  033514                  $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
035062  033702                  $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
035064  034176                  $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT SWR SETTING
035066  034126                  $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
035070  034410                  $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
035072  034540                  $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
035074  034712                  $RDOCT  ;;CALL=RDOCT    TRAP+12(104412) READ AN OCTAL NUMBER FROM TTY
```

POWER DOWN AND UP ROUTINES

```
4961                                    .SBTTL  POWER DOWN AND UP ROUTINES
                                        ;;*****************************************************************
                                        ;POWER DOWN ROUTINE
035076  012737  035236  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
035104  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
035112  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
035114  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
035116  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
035120  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
035122  010446                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
035124  010546                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
035126  017746  144006                MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
035132  010637  035242                MOV     SP,$SAVR6       ;;SAVE SP
035136  012737  035150  000024        MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
035144  000000                        HALT
035146  000776                        BR      .-2             ;;HANG UP
                                        ;;*****************************************************************
                                        ;POWER UP ROUTINE
035150  012737  035236  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
035156  013706  035242                MOV     $SAVR6,SP       ;;GET SP
035162  005037  035242                CLR     $SAVR6          ;;WAIT LOOP FOR THE TTY
035166  005237  035242          1$:   INC     $SAVR6          ;;WAIT FOR THE INC
035172  001375                        BNE     1$              ;;OF  WORD
035174  012677  143740                MOV     (SP)+,@SWR      ;;POP STACK INTO @SWR
035200  012605                        MOV     (SP)+,R5        ;;POP STACK INTO R5
035202  012604                        MOV     (SP)+,R4        ;;POP STACK INTO R4
035204  012603                        MOV     (SP)+,R3        ;;POP STACK INTO R3
035206  012602                        MOV     (SP)+,R2        ;;POP STACK INTO R2
035210  012601                        MOV     (SP)+,R1        ;;POP STACK INTO R1
035212  012600                        MOV     (SP)+,R0        ;;POP STACK INTO R0
035214  012737  035076  000024        MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
035222  012737  000340  000026        MOV     #340,@#PWRVEC+2 ;;PRIO:7
035230  104401                        TYPE                    ;REPORT THE POWER FAILURE
035232  035244          $PWRMG: .WORD   $POWER          ;;POWER FAIL MESSAGE POINTER
035234  000002                        RTI
035236  000000          $ILLUP: HALT                    ;;THE POWER UP SEQUENCE WAS STARTED
035240  000776                        BR      .-2             ;; BEFORE THE POWER DOWN WAS COMPLETE
035242  000000          $SAVR6: 0                       ;;PUT THE SP HERE
035244     015     012     120  $POWER: .ASCIZ  <15><12>"POWER"
035247     117     127     105
035252     122     000

                                        .EVEN
4963       000001                       .END
```

J12

SEQ 0152

Symbol table

```
ABASE = 000000        APRIOR= 000000        CSR22   002366        EM100   021504        EM61    021013
ABORT   032512        APTCSU= 000040        CURADD  004054        EM101   021542        EM63    021042
ABORTC  032544        APTENV- 000001        CURDAT  004046        EM102   021575        EM64    021073
ABORTE  032610        APTSIZ= 000200        DATA    002344        EM103   021657        EM65    021132
ABORTZ  032634        APTSPO= 000100        DATA2   002374        EM104   021732        EM71    021202
ACDW1 = 000000        ASWREG= 000000        DATVER  030556        EM105   022002        EM72    021236
ACDW2 = 000000        ATESTN= 000000        DATVFR  030540        EM106   022055        EM73    021274
ACPUOP= 000000        AUNIT = 000000        DAT1    030570        EM107   022133        EM74    021320
ACTCHS  002410        AUSWR = 000000        DCOUNT  004030        EM110   022167        EM75    021351
AC0   =%000000        AVECT1= 000000        DDISP = 177570        EM111   022231        EM76    021402
AC1   =%000001        AVECT2= 000000        DELAY   030610        EM112   022307        EM77    021452
AC2   =%000002        BA      002340        DETFPA  031544        EM113   022372        ERR     030536
AC3   =%000003        BA2     002370        DH1     025314        EM114   022456        ERRCNT  003024
AC4   =%000004        BCR   = 177524        DH105   026255        EM123   022477        ERRFP   030534
AC5   =%000005        BCSR  = 177520        DH115   026307        EM124   022531        ERROR = 104000
AC6   =%000006        BDR   = 177524        DH134   026364        EM125   022623        ERRVEC= 000004
AC7   =%000007        BEGROM= 100000        DH24    025600        EM126   022662        ERTYPE  027030
ADDTRP  030376        BIT0  = 000001        DH27    025644        EM127   022717        EXPDAT  004032
ADDW0 = 000000        BIT00 = 000001        DH4     025341        EM130   022761        FLAG    003032
ADDW1 = 000000        BIT01 = 000002        DH41    025707        EM131   023006        FLO     003066
ADDW10= 000000        BIT02 = 000004        DH43    025760        EM132   023036        FLOAT   003056
ADDW11= 000000        BIT03 = 000010        DH47    026060        EM135   023074        FPVEC = 000244
ADDW12= 000000        BIT04 = 000020        DH5     025426        EM136   023132        FSTADD  004050
ADDW13= 000000        BIT05 = 000040        DH65    026145        EM137   023174        GOODAD  003022
ADDW14= 000000        BIT06 = 000100        DH7     025525        EM140   023241        GTSWR = 104406
ADDW15= 000000        BIT07 = 000200        DH72    026211        EM141   023332        HITMIS= 177752
ADDW2 = 000000        BIT08 = 000400        DISPLA  001142        EM142   023415        HT    = 000011
ADDW3 = 000000        BIT09 = 001000        DISPRE  000174        EM143   023506        INIMEM  031012
ADDW4 = 000000        BIT1  = 000002        DMARD   030130        EM144   023561        INITMM  027246
ADDW5 = 000000        BIT10 = 002000        DMATRN  030046        EM145   023636        INQ22   017352
ADDW6 = 000000        BIT11 = 004000        DSWR  = 177570        EM146   023704        IOTVEC= 000020
ADDW7 = 000000        BIT12 = 010000        DT1     026502        EM147   023747        J11FLT  031552
ADDW8 = 000000        BIT13 = 020000        DT105   026760        EM150   024024        KDPAR0= 172360
ADDW9 = 000000        BIT14 = 040000        DT115   026770        EM151   024105        KDPAR1= 172362
ADEVCT= 000000        BIT15 = 100000        DT130   027004        EM152   024153        KDPAR2= 172364
ADEVM = 000000        BIT2  = 000004        DT134   027012        EM153   024235        KDPAR3= 172366
ADLSB   004040        BIT3  = 000010        DT14    026550        EM154   024315        KDPAR4= 172370
AENV  = 000000        BIT4  = 000020        DT17    026562        EM155   024402        KDPAR5= 172372
AENVM = 000000        BIT5  = 000040        DT24    026574        EM156   024455        KDPAR6= 172374
AFATAL= 000000        BIT6  = 000100        DT27    026604        EM157   024527        KDPAR7= 172376
ALLCTR  003156        BIT7  = 000200        DT35    026616        EM160   024575        KDPDR0= 172320
AMADR1= 000000        BIT8  = 000400        DT4     026510        EM161   024617        KDPDR1= 172322
AMADR2= 000000        BIT9  = 001000        DT41    026630        EM162   024657        KDPDR2= 172324
AMADR3= 000000        BPTVEC= 000014        DT43    026640        EM163   024722        KDPDR3= 172326
AMADR4= 000000        BTEXP   003104        DT47    026654        EM164   024766        KDPDR4= 172330
AMAMS1= 000000        BTRES   003114        DT5     026522        EM165   025045        KDPDR5= 172332
AMAMS2= 000000        CCHPAS  003034        DT50    026666        EM166   025114        KDPDR6= 172334
AMAMS3= 000000        CCR   = 177746        DT51    026700        EM167   025162        KDPDR7= 172336
AMAMS4= 000000        CKSWR = 104407        DT52    026712        EM170   025214        KIPAR0= 172340
AMSGAD= 000000        CLKCNT  027530        DT64    026724        EM171   025242        KIPAR1= 172342
AMSGLG= 000000        COUNT   003124        DT65    026736        EM172   025273        KIPAR2= 172344
AMSGTY= 000000        CPEREG= 177766        DT7     026536        EM2     020653        KIPAR3= 172346
AMTYP1= 000000        CR    = 000015        DT75    026746        EM3     020665        KIPAR4= 172350
AMTYP2= 000000        CRLF  = 000200        EEPAS   003036        EM51    020677        KIPAR5= 172352
AMTYP3= 000000        CSR1    002334        EMTSAV  004060        EM54    020677        KIPAR6= 172354
AMTYP4= 000000        CSR12   002364        EMTVEC= 000030        EM56    020735        KIPAR7= 172356
APASS = 000000        CSR2    002336        EM1     020617        EM57    020761        KIPDR0= 172300
```

Symbol table

```
KIPDR1= 172302   PIRQT   017710   SDPAR6= 172274   SW10  = 002000   TAB6A   003330
KIPDR2= 172304   PIRQVE= 000240   SDPAR7= 172276   SW11  = 004000   TAB7    003340
KIPDR3= 172306   POLY   = 120001   SDPDR0= 172220   SW12  = 010000   TAB8    003350
KIPDR4= 172310   PR0    = 000000   SDPDR1= 172222   SW13  = 020000   TAB9    003360
KIPDR5= 172312   PR1    = 000040   SDPDR2= 172224   SW14  = 040000   TBITVE= 000014
KIPDR6= 172314   PR2    = 000100   SDPDR3= 172226   SW15  = 100000   TCHAR   003004
KIPDR7= 172316   PR3    = 000140   SDPDR4= 172230   SW2   = 000004   TCOUNT  002360
KMCR  = 177734   PR4    = 000200   SDPDR5= 172232   SW3   = 000010   TEMP    002740
LDPARS  027410   PR5    = 000240   SDPDR6= 172234   SW4   = 000020   TIMERR  006050
LDPDRS  027440   PR6    = 000300   SDPDR7= 172236   SW5   = 000040   TIMOUT  003000
LEDCNT  002354   PR7    = 000340   SEQ      003076   SW6   = 000100   TKVEC = 000060
LF    = 000012   PS     = 177776   SETMMU  030676   SW7   = 000200   TOUT    030220
LKCNT   002404   PSW    = 177776   SIMGOA  002352   SW8   = 000400   TPVEC = 000064
LKS   = 177546   PWDSEQ  004036   SIPAR0= 172240   SW9   = 001000   TRAPVE= 000034
LKSFL   002402   PWRVEC= 000024   SIPAR1= 172242   TAB1    003240   TRPFLG  030532
LKSINT  027522   Q22EN   003002   SIPAR2= 172244   TAB10   003370   TRTVEC= 000014
LOOP    005064   Q22INT  030026   SIPAR3= 172246   TAB11   003400   TSTADD  003026
LOOPIN  003160   Q22SIZ  027546   SIPAR4= 172250   TAB11A  003410   TSTLOC  003166
LOPBAK  002362   RAMPAR  027464   SIPAR5= 172252   TAB12   003420   TST1    005220
LOWADD  003020   RBUF   = 177562   SIPAR6= 172254   TAB13   003430   TST10   010532
LSTADD  004052   RBUF1  = 176502   SIPAR7= 172256   TAB13B  003440   TST11   011552
MAIREG= 177750   RCOUNT  002356   SIPDR0= 172200   TAB14   003450   TST12   011762
MASK    003164   RCSR   = 177560   SIPDR1= 172202   TAB15   003460   TST13   012332
MER   = 172100   RCSR1  = 176500   SIPDR2= 172204   TAB16   003470   TST14   012546
MERTAG  003054   RDCHR  = 104410   SIPDR3= 172206   TAB17   003500   TST15   012706
MMR0  = 177572   RDLIN  = 104411   SIPDR4= 172210   TAB18   003510   TST16   013104
MMR1  = 177574   RDOCT  = 104412   SIPDR5= 172212   TAB2    003250   TST17   013302
MMR2  = 177576   RECDAT  004034   SIPDR6= 172214   TAB21   003520   TST2    005502
MMR3  = 172516   RECDST  003146   SIPDR7= 172216   TAB22   003530   TST20   013700
MMU     030230   RECFEC  003126   SLEND   016714   TAB23   003540   TST21   014056
MMUERR  006016   RECST   003136   SLOC00  003010   TAB24   003550   TST22   015100
MMUTRP  030402   RESVEC= 000010   SLOC01  003012   TAB25   003560   TST23   015654
MMVEC = 000250   RITEDA  004042   SPS     003100   TAB26   003570   TST24   016266
MSER  = 177744   RRMEM   031350   SPSJ    003102   TAB27   003600   TST25   016720
NAIREG= 177520   RWTMEM  031176   SR0    = 177572   TAB28   003610   TST26   017134
NEWADD  003030   R6     =%000006   SR1    = 177574   TAB29   003620   TST27   017434
NEWDAT  004044   R7     =%000007   SR2    = 177576   TAB29A  003630   TST3    006102
NOABRT  032650   SAVBR   002414   SR3    = 172516   TAB3    003260   TST30   017712
NOSLU   016716   SAVLOC  002406   STACK  = 001100   TAB30   003640   TST4    006342
NOTOK   030460   SAVMR0  003044   START   004060   TAB31   003650   TST5    006570
NULL  = 000000   SAVMR1  003046   STBOT  = 001000   TAB32   003660   TST6    007452
NXTST   011552   SAVMR2  003050   STKLMT= 177774   TAB33   003670   TST7    010200
OK      030436   SAVPCR  002412   SWR     001140   TAB34   003700   TYPDS = 104405
OK1     030464   SAVPOS  003162   SWREG   000176   TAB4    003270   TYPE  = 104401
ONOQ22  027772   SAVSUP  003040   SWTSEL  017750   TAB40   003710   TYPOC = 104402
PAR     030352   SAVSWR  003052   SW0    = 000001   TAB41   003720   TYPON = 104404
PARABT  004056   SAVTIM  003016   SW00   = 000001   TAB42   003730   TYPOS = 104403
PAREND  007452   SAVUSE  003042   SW01   = 000002   TAB43   003740   UDPAR0= 177660
PARINT  007444   SAV30   004146   SW02   = 000004   TAB45   003750   UDPAR1= 177662
PARPAT  003014   SAV32   004150   SW03   = 000010   TAB46   003760   UDPAR2= 177664
PAR1    030354   SCOPE  = 000004   SW04   = 000020   TAB47   003770   UDPAR3= 177666
PATT    006324   SDPAR0= 172260   SW05   = 000040   TAB47A  004000   UDPAR4= 177670
PCR   = 177522   SDPAR1= 172262   SW06   = 000100   TAB48   004010   UDPAR5= 177672
PDR     030330   SDPAR2= 172264   SW07   = 000200   TAB49   004020   UDPAR6= 177674
PDR1    030332   SDPAR3= 172266   SW08   = 000400   TAB5    003300   UDPAR7= 177676
PIR   = 177772   SDPAR4= 172270   SW09   = 001000   TAB5A   003310   UDPDR0= 177620
PIRQ  = 177772   SDPAR5= 172272   SW1    = 000002   TAB6    003320   UDPDR1= 177622
```

L12

Symbol table

```
UDPDR2= 177624          $ATY1   032652          $ENDAD  031732          $MAIL   001200          $SVPC = 000232
UDPDR3= 177626          $ATY3   032660          $ENDCT  031700          $MAMS1  001230          $SWR  = 167400
UDPDR4= 177630          $ATY4   032670          $ENDMG  031751          $MAMS2  001234          $SWREG  001222
UDPDR5= 177632          $AUTOB  001134          $ENULL  031746          $MAMS3  001240          $SWRMK= 000300
UDPDR6= 177634          $BASE   001254          $ENV    001220          $MAMS4  001244          $TESTN  001204
UDPDR7= 177636          $BDADR  001122          $ENVM   001221          $MBADR  000234          $TIMES  001164
UFDFLG  004152          $BDDAT  001126          $EOP    031632          $MFLG   033114          $TKB    001146
UFDSET= 000001          $BELL   001170          $EOPCT  031672          $MNEW   034701          $TKS    001144
UIPAR0= 177640          $CDW1   001260          $ERFLG  001103          $MSGAD  001214          $TMP0   001160
UIPAR1= 177642          $CDW2   001262          $ERMAX  001115          $MSGLG  001216          $TMP1   001162
UIPAR2= 177644          $CHARC  033450          $ERROR  032270          $MSGTY  001200          $TN   = 000031
UIPAR3= 177646          $CKSWR  034126          $ERRPC  001116          $MSWR   034670          $TPB    001152
UIPAR4= 177650          $CMTAG  001100          $ERRTB  001324          $MTYP1  001231          $TPFLG  001157
UIPAR5= 177652          $CM3  = 000000          $ERTTL  001112          $MTYP2  001235          $TPS    001150
UIPAR6= 177654          $CM4  = 000002          $ESCAP  001166          $MTYP3  001241          $TRAP   035014
UIPAR7= 177656          $CNTLG  034663          $ETABL  001220          $MTYP4  001245          $TRAP2  035036
UIPDR0= 177600          $CNTLU  034656          $ETEND  001324          $MXCNT  032266          $TRP  - 000013
UIPDR1= 177602          $CPUOP  001226          $FATAL  001202          $NULL   001154          $TRPAD  035050
UIPDR2= 177604          $CRLF   001175          $FFLG   033116          $NWTST= 000000          $TSTM   000236
UIPDR3= 177606          $DBLK   034116          $FILLC  001156          $OCNT   033676          $TSTNM  001102
UIPDR4= 177610          $DDW0   001264          $FILLS  001155          $OMODE  033700          $TTYIN  034646
UIPDR5= 177612          $DDW1   001266          $GDADR  001120          $OVER   032244          $TYPDS  033702
UIPDR6= 177614          $DDW10  001310          $GDDAT  001124          $PASS   001206          $TYPE   033120
UIPDR7= 177616          $DDW11  001312          $GET42  031722          $PASTM  000240          $TYPEC  033332
UQUIET  004154          $DDW12  001314          $GTSWR  034176          $POWER  035244          $TYPEX  033452
VIREOP  017724          $DDW13  001316          $HD   = 000001          $PWRDN  035076          $TYPOC  033500
VMKOR   004156          $DDW14  001320          $HIBTS  000232          $PWRMG  035232          $TYPON  033514
VQBE1   002346          $DDW15  001322          $HIOCT  035012          $PWRUP  035150          $TYPOS  033454
VQBE2   002376          $DDW2   001270          $ICNT   001104          $QUES   001174          $UNIT   001212
VQPR1   002350          $DDW3   001272          $ILLUP  035236          $RDCHR  034410          $UNITM  000242
VQPR2   002400          $DDW4   001274          $INTAG  001135          $RDLIN  034540          $USWR   001224
WC      002342          $DDW5   001276          $ITEMB  001114          $RDOCT  034712          $VECT1  001250
WC2     002372          $DDW6   001300          $LF     001176          $RDSZ = 000010          $VECT2  001252
WLDTRP  030522          $DDW7   001302          $LFLG   033115          $RTNAD  031744          $XOFF = 000023
XBUF  = 177566          $DDW8   001304          $LPADR  001106          $SAVR6  035242          $XON  = 000021
XBUF1 = 176506          $DDW9   001306          $LPERR  001110          $SCOPE  031766          $XTSTR  032006
XCSR  = 177564          $DEVCT  001210          $MADR1  001232          $SETUP= 000137          $$GET4= 000000
XCSR1 = 176504          $DEVM   001256          $MADR2  001236          $STUP = 177777          $OFILL  033677
$APTHD  000232          $DOAGN  031742          $MADR3  001242          $SVLAD  032210          .$X   = 000232
$ATYC   032676          $DTBL   034106          $MADR4  001246

. ABS.  035254    000   (RW,I,GBL,ABS,OVR)
        000000    001   (RW,I,LCL,REL,CON)
Errors detected:  0

*** Assembler statistics


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

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