RK11

BASIC LOGIC TEST NO. 1

MD-11-DZRKJ-D

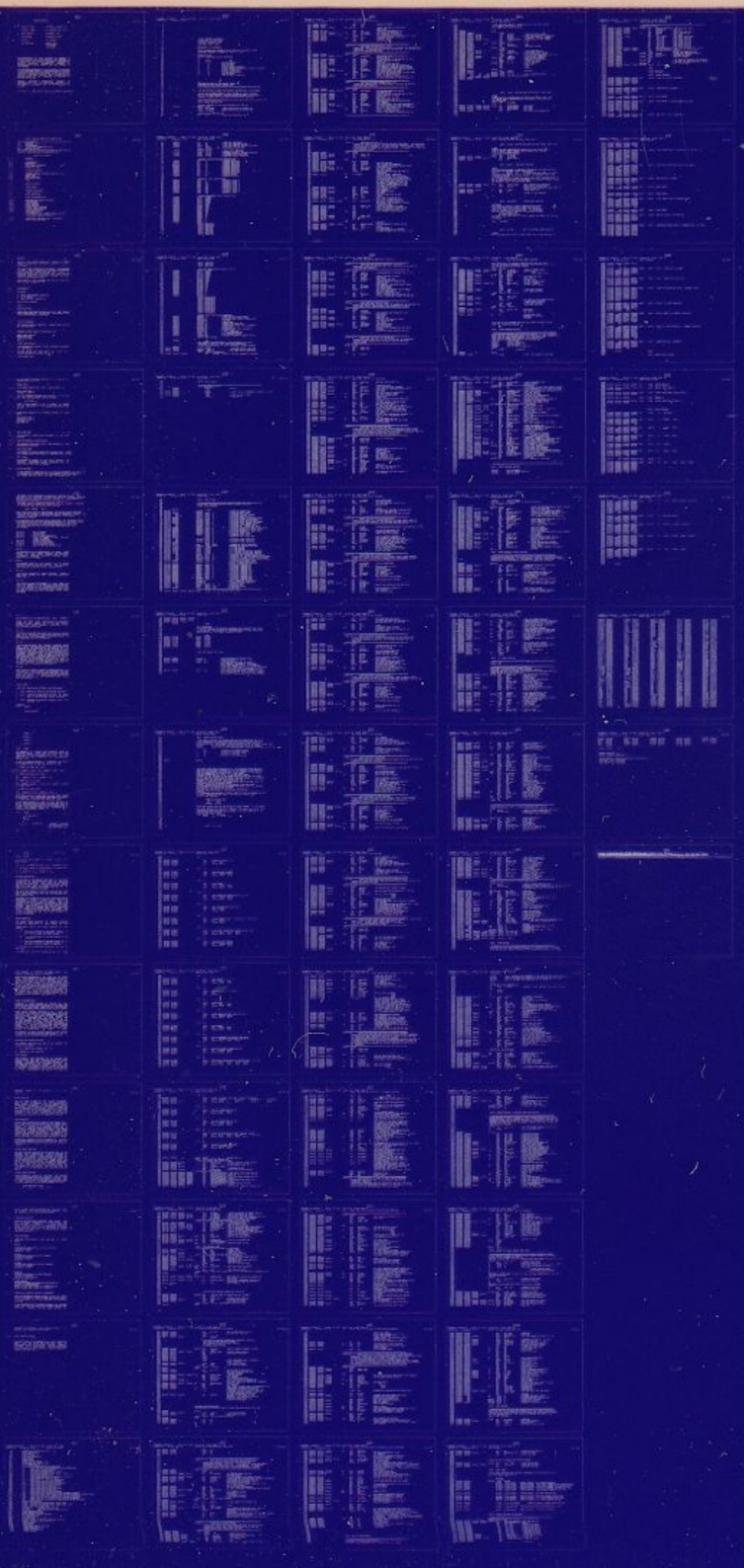
EP-DZRKJ-D-DL-A

FICHE 1 OF 1

DEC 1976

digital

MADE IN USA



SEQ 000:

IDENTIFICATION

PRODUCT CODE:	MAINDEC-11-DZRKJ-D-D
PRODUCT NAME:	RK11 BASIC LOGIC TEST I
DATE CREATED:	DECEMBER, 1976
MAINTAINER:	DIAGNOSTIC GROUP
AUTHOR:	JIM KAPADIA
REVISED BY:	PERVEZ ZAKI TOM SAWYER CHUCK MESS

QUICK LOOK-UP OPERATING INSTRUCTIONS

FOR A QUICK REFERENCE, LOOK UP THE FOLLOWING SECTIONS:

FOR A MORE COMPLETE EXPLANATION REFER TO THE TABLE OF CONTENTS BELOW AND THE FOLLOWING DOCUMENT.

TABLE OF CONTENTS

1.0 ABSTRACT

THE RK11 LOGIC TESTS CONSIST OF A SERIES OF TESTS AIMED AT CHECKING THE BASIC LOGIC OF THE RK11 CONTROLLER.

THE LOGIC TESTS CONSISTS OF TWO PARTS. THIS PROGRAM IS PART-I AND IT CHECKS ONLY THE DRIVE-INDEPENDENT LOGIC OF THE RK11 CONTROLLER (SEE SEC. 9-0). IT SHOULD BE NOTED THAT LOGIC TEST-I AND LOGIC TEST-II TOGETHER CONSTITUTE A COMPLETE PROGRAM AND HENCE BOTH OF THEM SHOULD BE RUN.

USED CORRECTLY THIS PROGRAM CAN BE AN EFFECTIVE ANALYTIC AND DIAGNOSTIC TOOL.

2.0 REQUIREMENTS

2.1 EQUIPMENT

A. PDP11 WITH CONSOLE TELETYPE.
B. 8K OF MEMORY
C. RK11 OR RKV11 CONTROLLER

2.2 PRELIMINARY PROGRAMS

NONE

2.3 EXECUTION TIME

ERROR FREE FIRST PASS ON PDP11/20 WITH CORE MEMORY TAKES APPROXIMATELY ONE MINUTE. CONSIDERABLY LESS FOR FASTER MACHINES OR MEMORIES.

3.0 STARTING ADDRESS

200 FOR ANY MODE OF OPERATION. NORMAL START UP WITH ALL SWITCHES DOWN.

4.0 PROGRAM CONTROL MODES & OPERATOR ACTION

PAPER TAPE LOADING
RKDP DUMP MODE
RKDP CHAIN MODE
ACT11

4.1 PAPER TAPE LOADING

4.1.1 LOAD PROGRAM INTO MEMORY USING STANDARD PROCEDURE FOR .ABS TAPES.

4.1.2 PUT THE DRIVES ON 'WRT PROT' AND 'LOAD' AS A PRECAUTION AGAINST MALFUNCTIONING.

4.1.3 LOAD ADDRESS 200

4.1.4 SET SWITCHES IF DESIRED (SEE SEC 5.0. IF TESTING ON SIMULATOR PUT SW 10 UP. SEQ 0004

PRESS START.

4.1.5 THE PROGRAM IDENTIFIES ITSELF (NAME,MAINDEC NO).

RK11 LOGIC TEST 1
MAINDEC-11-DZRKJ-D

4.1.6 THEN THE PROGRAM PROCEEDS WITH TESTING. AT THE END OF A PASS THE FOLLOWING TYPE-OUT OCCURS

END PASS # X

WHERE X= PASS NUMBER (1,2,3---), CONTROL IS PASSED TO THE BEGINNING OF THE PROGRAM AND RE-EXECUTION BEGINS.

4.1.7 ERROR FREE PASSES OF THE PROGRAM APPEAR AS SHOWN BELOW.

RK11 LOGIC TEST 1
MAINDEC-11-DZRKJ-D
END PASS # 1
END PASS # 2
...
...

4.2 RKDP DUMP MODE

4.2.1 THE PROGRAM IS LOADED INTO THE MEMORY BY THE RKDP MONITOR

4.2.2 START AS NORMALLY USING SA 200

4.2.3 THE PROGRAM IDENTIFIES ITSELF (NAME,MAINDEC NO.) AND PROCEEDS WITH TESTING.

4.3 RKDP CHAIN MODE

THE PROGRAM IS CHAIN-LOADED FROM THE RKDP PACK. AFTER THE PROGRAM IDENTIFIES ITSELF, IT PROCEEDS WITH TESTING.

4.4 ACT11 MODE

THE PROGRAM IS LOADED BY THE ACT11 MONITOR. ON STARTING, IDENTIFIES ITSELF, PROCEEDS WITH THE EXECUTION OF THE TESTS AS BEFORE.

5.0 SWITCH OPTIONS

IF THE PROGRAM IS BEING RUN ON A SWITCHLESS PROCESSOR (I.E. AN 11/34) THE PROGRAM WILL DETERMINE THAT THE HARDWARE SWITCH REGISTER IS NOT PRESENT AND WILL USE A 'SOFTWARE' SWITCH REGISTER. THE

'SOFTWARE' SWITCH REGISTER IS LOCATED AT LOCATION 176 (8). THE SETTINGS OF THE 'SOFTWARE' SWITCHES ARE CONTROLLED THROUGH A KEYBOARD ROUTINE WHICH IS CALLED BY TYPING A 'CONTROL G'. THE PROGRAM WILL RECOGNIZE THE 'CONTROL G' whenever the program enters the scope routine or begins a new test, the 'SOFTWARE' SWITCH VALUES ARE ENTERED AS AN OCTAL NUMBER IN RESPONSE TO THE PROMPT FROM THE SWITCH ENTRY ROUTINE:

'SWR = NNNNNN NEW ='

EACH TIME SWITCH SETTING ARE ENTERED, THE ENTIRE SWITCH REGISTER IMAGE MUST BE ENTERED. LEADING ZEROS ARE NOT REQUIRED. 'RUBOUT' AND 'CONTROL U' FUNCTIONS MAY BE USED TO CORRECT TYPING ERRORS DURING SWITCH ENTRY.

ON PROCESSORS WITH HARDWARE SWITCH REGISTERS, THE 'SOFTWARE' SWITCH REGISTER MAY BE USED. IF THE PROGRAM FINDS ALL 16 SWITCHES IN THE 'UP' POSITION, ALL SWITCH REGISTER REFERENCES WILL BE TO THE 'SOFTWARE' REGISTER AND THE PROCEDURES DESCRIBED ABOVE MUST BE FOLLOWED.

SW<15>=1	HALT ON ERROR
SW<14>=1	LOOP ON TEST
SW<13>=1	INHIBIT ERROR PRINTOUTS
SW<12>=1	CYCLE ON ERROR TO THE PREVIOUS 'SCOPE' STATEMENT
SW<11>=1	INHIBIT ITERATIONS
SW<10>=1	TESTING ON SIMULATOR
SW<09>=1	LOOP ON SPECIFIC ERROR
SW<08>=1	LOOP ON TEST AS PER SW<07:00>

5.1 SW<15>

THE PROGRAM HALTS ON ENCOUNTERING AN ERROR, AFTER TYPING OUT THE ERROR MESSAGE AND PERTINENT INFORMATION. PRESSING "CONTINUE" RESTORES NORMAL OPERATION OF THE PROGRAM.

5.2 SW<14>

THE PROGRAM LOOPS ON THE SUBTEST THAT IS BEING EXECUTED WHEN THE SWITCH IS PUT ON. THIS SWITCH IS USED NORMALLY ALONG WITH SW 15. SEE SEC 8.0.

5.3 SW <13>

THIS SWITCH INHIBITS ALL ERROR MESSAGES. NORMALLY USED WHEN LOOPING ON TEST (SW 14) OR LOOPING ON ERROR (SW 9..

5.4 SW <12>

THIS SWITCH ALLOWS THE PORGRAM TO CYCLE FROM THE POINT OF ERROR TO THE PREVIOUS SCOPE STATEMENT. NOTE THAT IN DOING SO ANY INITIALIZATION BEING DONE AT THE BEGINING OF THE SUBTEST WILL BE DONE AGAIN AND AGAIN. SEE SEC 8.0 FOR DIFFERENT SCOPE LOOPS AVAILABLE.

5.5 SW <11>

EACH SUBTEST WILL BE EXECUTED ONLY ONCE. NORMALLY

AFTER THE FIRST PASS, EACH SUBTEST IS ITERATED A NUMBER OF TIMES (USUALLY 50, 5 IN SOME CASES). SETTING THIS SWITCH INHIBITS ITERATIONS, SO THAT QUICK PASSES CAN BE MADE.

5.6 SW <10>

THIS SWITCH WHEN SET INDICATES THAT TESTING IS BEING DONE ON A SIMULATOR. THE SWITCH SHOULD BE PUT UP BEFORE START- ING THE PROGRAM. NOTE THAT RK11C IS NOT COMPATIBLE WITH THE SIMULATOR.

5.7 SW <09>

THIS SWITCH PROVIDES THE TIGHTEST POSSIBLE SCOPE LOOP. NOTE THAT UNLIKE SW12 THE INITIALIZATION OF PARAMETERS AT THE BEGINNING OF THE SUBTEST MAY NOT BE DONE IN THIS CASE. THIS SWITCH IS HELPFUL WHEN A PARTICULAR PART OF A SUBTEST IS BEING REPEATED USING DIFFERENT PARAMETERS AND YOU WANT TO SCOPE ON THE PARAMETER IN ERROR. (EXAMPLE: RKDA IS BEING WRITTEN AND READ BACK WITH COUNT PATTERNS FROM 1 TO 177777. PATTERN 561 IS GIVING ERROR, YOU MIGHT NOT WANT TO GO THROUGH THE 560 PATTERNS BEFORE HITTING ERROR ON THE 561TH PATTERN. IN THIS CASE SW 9 WILL GIVE YOU A SCOPE LOOP ON THE 561TH PATTERN ONLY.)

5.8 SW <08>

THIS SWITCH IS USED TO SELECT A PARTICULAR TEST (AS PER SW<00-07>) FOR EXECUTION AND SUBSEQUENT LOOPING. THUS IF TEST 15 IS TO BE SELECTED THE SWITCH SETTING WOULD BE 000415. IT SHOULD BE NOTED THAT BEFORE SELECTING TEST 15, ALL THE PREVIOUS TESTS (1-14) WILL BE EXECUTED.

6.0 SCOPE LOOPS

THERE ARE THREE KINDS OF SCOPE LOOPS AVAILABLE

1. SW14: LOOPING IS DONE FOR THE ENTIRE SUB-TEST

2. SW12: LOOPING IS DONE FROM THE POINT OF ERROR BACK TO THE PREVIOUS 'SCOPE' STATEMENT.

3. SW09: PROVIDE THE TIGHTEST POSSIBLE SCOPE LOOP SEE SEC. 5.7

EXAMPLE:

```
TST1:   SCOPE
        :

        INITIALIZATION
        :
```

```
        ERROR 1
        :
        ERROR 2
        :
        ERROR 3
        :
        ERROR 4
        :
        :
TST2:   SCOPE
```

THE SEQUENCE OF LOOPING FOR DIFFERENT CASES IS EXPLAINED BELOW. NOTE THAT 'TST1' AND 'TST2' ARE TAGS WHICH DEFINE THE BOUNDARY OF A TEST, (IN THIS CASE TEST 1). TEST 1 STARTS AT 'TST1' AND ENDS JUST BEFORE 'TST2'.

IN THE ILLUSTRATION BELOW --> INDICATES THE POINT FROM WHERE RETURN IS MADE AND LOOPING IS DONE.

1. ERROR 2 OCCURS, SW 14 SET.

TST1..ERROR 2..TST2-->TST1..ERROR 2..TST2-->TST1...

2. ERROR 2 OCCURS, SW 12 SET.

TST1...ERROR 2-->TST1...ERROR2-->TST1...

3. ERROR 2,3; SW 14 SET.

TST1..ERROR 2..ERROR 3..TST2-->TST1..ERROR 2..ERROR 3..TST2-->TST1...

4. ERROR 2,3; SW 12 SET.

TST1...ERROR 2-->TST1...ERROR 2-->TST1....

NOTE THAT LOOPING IS DONE FROM THE VERY FIRST ERROR ENCOUNTERED. THE MORE BASIC AND ERROR THE EARLIER IT OCCURS AND IS DETECTED AND SHOULD BE FIXED.

IN THE ABOVE EXAMPLE NO PART OF THE SUB-TEST IS BEING REPEATED USING DIFFERENT PARAMETERS, HENCE IT SO HAP- PENS THAT SW 9 AND 12 GIVE THE SAME KIND OF LOOPS. THE EXAMPLE BELOW WILL DEMONSTRATE THE DIFFERENCE BETWEEN SW 9 AND 12.

```
TST1:   SCOPE
        :
        INITIALIZATION
        :
        ERROR 1

        :
        MOV     #1$,$LPERR      ;'$LPERR'   CONTAINS
                                ;THE ADDRESS TO LOOP
                                ;BACK ON ERROR- SW 9
        :
1$:     :                       ----
        :                       I
```

```
        ERROR 2                 I   N REPETITIONS
                                I
TST2:   $SCOPE                  ----
```

1. SW 12 SET, ERROR 2 OCCURS DURING K.TH REPETITIONS

TST1..1,2...K.ERROR 2-->TST1..1,2...K.ERROR 2-->TST1..

2. SW 9 SET, ERROR 2 OCCURS DURING K.TH REPETITION

1$..K..ERROR 2-->1$..K..ERROR 2-->1$...

7.0 PROGRAM DESCRIPTION

IN THIS PART OF THE PROGRAM THAT PART OF THE RK11 CONTROLLER IS CHECKED WHICH DOES NOT DEPEND ON SIGNALS FROM THE DRIVE. THUS A DRIVE IS NOT NEEDED FOR THIS TEST, BUT IT SHOULD BE NOTED THAT THE PART-II OF THE 'BASIC LOGIC TESTS' MUST BE RUN, IN ORDER TO GET A COMPLETE COVERAGE.

THE TESTS ARE GRADUALLY BUILT UP, CHECKING THE MOST BASIC AND SIMPLE LOGIC FIRST AND THEN PROGRESSIVELY MORE COMPLEX LOGIC.

THE FIRST TEST CHECKS THAT ALL RK11 REGISTERS CAN BE REFERENCED WITHOUT TIMING OUT. THEN THE INITIALIZATION LOGIC OF RK11 IS CHECKED. THEN IT IS CHECKED THAT ALL REGISTERS CAN BE WRITTEN AND READ CORRECTLY, BY FLOATING A '1' AND THEN USING A COUNT PATTERN. THEN IT IS CHECKED THAT THE RK11 REGISTERS CAN BE CLEARED USING CONTROL RESET AND RESET (BUS INIT). FINALLY, THE WORD AND BYTE ADDRESSING LOGIC OF RK11 IS CHECKED TO SEE THAT EACH REGISTER IS UNIQUELY ADDRESSED.

8.0 ERROR REPORTING

THE ERROR TABLE STARTING AT $ERRTB CONTAINS INFORMATION PERTAINING TO EVERY ERROR THAT CAN

OCCUR. EACH ITEM IN THE TABLE CONSISTS OF FOUR ENTRIES.

A. EM - THIS IS A POINTER TO THE ERROR MESSAGE TO BE TYPED OUT WHEN THE ERROR OCCURS.

B. DH - THIS IS A POINTER TO THE DATA HEADER TO BE TYPED OUT.

C. DT - THIS IS A POINTER TO THE DATA WHICH IS TO BE TYPED TYPED OUT UNDER THE HEADERS.

D. 0 - THIS IS A TERMINATOR SIGNIFYING THE END OF THE ITEM.

THE ERROR CALL IS AN EMT INSTRUCTION WITH ITS LOWER

BYTE ENCODED TO INDICATE THE ERROR NUMBER. THUS "ERROR 1" WOULD BE (EMT+1) IE 104001.

EVERY ERROR CORRESPONDS TO AN ITEM IN THE ERROR TABLE. THUS "ERROR 14" WOULD CORRESPOND TO ITEM 14. AS FAR AS POSSIBLE, THE ERROR MESSAGES HAVE BEEN KEPT SHORT, BUT CLARITY IS NOT SACRIFICED FOR BREVITY. INSPITE OF THIS, IF THE USER FINDS A NEED, HE CAN LOOK UP THE ENTIRE ERROR MESSAGE IN THE ERROR ITEMS TABLE FOUND IN THE BEGINNING OF THE LISTINGS. THUS FOR "ERROR 14", "ITEM 14" IN THE ITEM TABLE CAN BE LOOKED UP. WHEN THE ERROR INSTRUCTION IS EXECUTED A TRAP OCCURS TO THE ERROR HANDLER LOCATED AT $ERROR WHICH PROCESSES THE ERROR CALL. SEE SEC 12.3

9.0 ERROR INTERPRETATION

WHENEVER AN ERROR MESSAGE IS PRINTED OUT, ALL REGISTERS AND OTHER DATA PERTAINING TO THE ERROR ARE ALSO GIVEN. RKDS, RKER...RKBA INDICATE THE CONTENTS OF THE CORRESPONDING REGISTERS AT THE TIME OF ERROR.

EVERY ERROR MESSAGE CONTAINS A PC. THIS PC INDICATES THE POSITION IN PROGRAM WHERE THE ERROR CALL IS LOCATED. THE ERROR MESSAGE, BECAUSE OF PRACTICAL CONSIDERATIONS IS MADE SHORT AND MEANINGFUL. THE USER IS ADVISED TO LOOK UP THE PC IN THE PROGRAM LISTING, WHERE HE WILL FIND MORE INFORMATION ABOUT THE ERROR. IN MANY INSTANCES, A SINGLE FAULT WILL GIVE RISE TO MORE THAN ONE ERROR REPORT. A LITTLE DELIBERATION AND CAREFUL EXAMINATION OF THE DATA GIVEN WILL BE CERTAINLY VERY HELPFUL IN PINPOINTING THE FAULT. A BRIEF

EXPLANATION OF WHAT IS BEING CHECKED IN THE SUBTEST IS GIVEN AT THE BEGINNING OF EVERY SUBTEST. ALL THE NUMBERS GIVEN WITH ERROR MESSAGES ARE IN OCTAL.

10.0 HANDLERS AND COMMON ROUTINES

THE COMMONLY USED ROUTINES USED IN THE PROGRAM ARE CALLED IN TWO WAYS.

A. AS A SUBROUTINE THROUGH 'JSR' CALL

B. THROUGH A 'TRAP' HANDLER

10.1 TRAP HANDLER

MANY COMMONLY USED ROUTINES IN THE PROGRAM ARE CALLED USING THE TRAP INSTRUCTION AND THE 'TRAP' HANDLER. THE LOWER BYTE OF THE TRAP INSTRUCTION IS ENCODED DIFFERENTLY FOR DIFFERENCT ROUTINES. THE TRAP HANDLER IS LOCATED AT '$TRAP'. WHEN A CALL FOR A ROUTINE IS EXECUTED, A TRAP OCCURS TO THE HANDLER AT '$TRAP'. THE HANDLER PICKS UP THE LOWER BYTE OF THE "CALL INSTRUCTION" AND USES IT TO FORM THE

STARTING ADDRESS OF THE ROUTINE TO GO TO FOR SERVICE.

10.2 SCOPE HANDLER

THE 'IOT' TRAP IS USED BY THE 'SCOPE' STATEMENT. WHEN 'SCOPE' IS EXECUTED, AN IOT TRAP OCCURS TO MEMORY LOCATION '$SCOPE'. THE SCOPE HANDLER STARTS AT $SCOPE. DEPENDING ON THE SWITCH SETTINGS THE HANDLER DECIDES TO LOOP ON TEXT, INHIBIT ITERATIONS ETC. THERE ARE CERTAIN POINTERS AND FLAGS WHICH ARE ADJUSTED. THUS, IT IS NOT ADVISABLE TO START THE PROGRAM AT ANY GIVEN LOCATION SINCE THE VARIOUS POINTERS AND FLAGS MAY NOT BE CORRECTLY ADJUSTED.

10.3 ERROR HANDLER

AN EMT TRAP INSTRUCTION IS USED BY THE ERROR CALL. THE LOWER BYTE IS ENCODED TO GIVE DIFFERENT ERROR CALLS. (EX: ERROR 1 = 104000+1; ERROR 16 = 104000+16). WHEN THE ERROR STATEMENT IS EXECUTED, A TRAP OCCURS TO MEMORY LOCATION '$ERROR'. THE ERROR HANDLER IS LOCATED AT '$ERROR'. THE HANDLER FORMS THE POINTER TO ERROR TABLE, WHICH IS USED IF AN ERROR MESSAGE IS TO BE TYPED OUT. DEPENDING ON THE SWITCH SETTINGS, A DECISION ABOUT HALTING ON ERROR,

INHIBITING TYPEOUT, LOOPING ON ERROR ETC. IS MADE. IF AN ERROR MESSAGE IS TO BE TYPED OUT AN EXIT IS MADE TO THE ERROR MESSAGE TYPEOUT ROUTINE LOCATED AT '$ERRTYP'.

10.4 CONTROL RESET ROUTINE

THE CALL FOR THIS ROUTINE IS "CNT.RESET" AND IS AN ENCODED 'TRAP' INSTRUCTION. WHEN "CNT.RESET" IS EXECUTED THE CONTROL RESET ROUTINE STARTING AT "CN.RST" IS ENTERED. A CONTROL RESET IS ISSUED AND THE PROGRAM WAITS TILL THE CONTROL READY SETS, ON WHICH THE ROUTINE IS EXITED. IF CONTROL READY DOES NOT SET WITHIN A CERTAIN TIME AN ERROR IS REPORTED. THE PC TYPED OUT IS THE LOCATION WHERE THE "CNT.RESET" CALL IS LOCATED. THE WAITING TIME IS 2.8 MS FOR 11/20 AND 560 US FOR 11/45 WITH BIPOLAR MEMORY.

10.5 CONTROL READY ROUTINE

THIS ROUTINE IS CALLED BY "CNT.RDY" (AN ENCODED 'TRAP' INSTRUCTION) AND IS LOCATED AT "CN.RDY". THE ROUTINE WAITS FOR THE CONTROL READY TO SET AND WHEN IT DOES, EXITS OUT. IF CONTROL READY DOES NOT SET WITHIN A SPECIFIED TIME AN ERROR MESSAGE IS GIVEN

```
        CNTRL RDY DIDN'T SET
        PC = XXXXXX     RKCS = YYYYYY
```

THE PC IS THE LOCATION AT WHICH THE "CNT.RDY" CALL IS LOCATED. THE WAITING TIME IS 949 MS FOR 11/20 AND 189 MS FOR 11/45 WITH BIPOLAR MEMORY.

10.6 TIME DELAY ROUTINE

THIS ROUTINE PROVIDES A VARIABLE TIME DELAY. THE CALL IS DELAY ,N WHERE N=1 TO 177777 (OCTAL) TIME DELAY PROVIDED= 7.5 TIMES(X) N MICRO SECS FOR 11/20, 1.5N US FOR 11/45 (N CONVERTED TO DECIMAL BEFORE COMPUTING DELAY) IF THE USER WANTS TO CHANGE THE DELAY AT ANY POINT IT CAN BE DONE BY SIMPLY CHANGING VARIABLE 'N'.

10.7 OTHER ROUTINES

THERE ARE OTHER COMMONLY USED ROUTINES AS LISTED BELOW.

$TYPE:

TYPE ROUTINE FOR TYPING OUT ASCII STRINGS.
LOCATED AT "$TYPE"
CALLED BY "TYPE"

$TYPOC:
ROUTINE FOR TYPING OUT OCTAL NUMBERS.
LOCATED AT "$TYPOC"
CALLED BY "TYPOC"

$TYPDS:
ROUTINE FOR TYPING OUT DECIMAL NUMBERS.
LOCATED AT "$TYPDS"
CALLED BY "TYPDS"

$ERRTYP:
ROUTINE FOR TYPING OUT ERROR MESSAGES.
LOCATED AT $ERRTYP
CALLED BY "JSR $ERRTYP"

$PWRDN,$PWRUP:
ROUTINE FOR HANDLING POWER FAILURE/POWER UP.
LOCATED AT $PWRDN,$PWRUP
$PWRFL,CALLED WHEN THERE IS A POWER FAILURE.
$PWRUP,CALLED WHEN THERE IS A POWER UP.

11.0 UNEXPECTED TIMEOUTS AND RK11 INTERRUPTS

WHEN AN UNEXPECTED TIMEOUT OCCURS, THE PC AT WHICH TIME OUT OCCURED IS TYPED OUT AND THE PROGRAM HALTS. IF IT IS INTACT, IT CAN BE RESTARTED BY PRESSING CONTINUE.

IF AN UNEXPECTED RK11 INTERRUPT;OCBURS THE PROGRAM TYPES OUT THE PC AT WHICH THE INTERRUPT CAME IN AND THEN HALTS. PRESSING CONTINUE WOULD RESTART THE

PROGRAM FROM BEGINING. SW 9- LOOPING CAPABILITY IS PROVIDED AS A TROUBLE SHOOTING AID.

12.0 QUICK VERIFYING MODE

THE FIRST PASS OF THE PROGRAM IS A QUICK VERIFYING MODE. ALL THE TESTS ARE DONE ONLY ONCE, ON SUBSEQUENT PASSES THE TESTS ARE ITERATED (NORMALLY 50 TIMES, 5 IN SOME CASES). THUS THE FIRST PASS TAKES A SHORTER TIME TO COMPLETE, WHEREAS SUBSEQUENT PASSES TAKE MORE TIME.

```
                        .TITLE  MAINDEC-11-DZRKJ-D
                        ;*COPYRIGHT (C) 1974,1976
                        ;*DIGITAL EQUIPMENT CORP.
                        ;*MAYNARD, MASS. 01754
                        ;*
                        ;*PROGRAM BY JIM KAPADIA
                        ;*
                        ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                        ;*PACKAGE (MAINDEC-11-DZQAC-C2), SEPT 14, 1976.
                        ;*
                        ;*JANUARY 1975
                        .SBTTL  OPERATIONAL SWITCH SETTINGS
                        ;*
                        ;*      SWITCH                  USE
                        ;*      ------          ------------------
                        ;*        15            HALT ON ERROR
                        ;*        14            LOOP ON TEST
                        ;*        13            INHIBIT ERROR TYPEOUTS
                        ;*        12            CYCLE ON ERROR TO PREVIOUS 'SCOPE' STATEMENT
                        ;*        11            INHIBIT ITERATIONS
                        ;*        10            TESTING ON SIMULATOR
                        ;*         9            LOOP ON ERROR
                        ;*         8            LOOP ON TEST IN SWR<7:0>



                        ;*
                        ;*PROGRAM REVISED BY TOM SAWYER, MARCH 1976
                        ;*REVISED BY CHUCK HESS, AUGUST 1976
                        ;*****************************************************************

                        ;YOU ARE ADVISED TO READ THE DOCUMENT BEFORE USING THIS PROGRAM.

                        ;ON GETTING AN ERROR REFER TO THE LISTINGS AT THE PC POINTED
                        ;OUT IN THE ERROR MESSAGE. ADJACENT ERROR MESSAGES IF FOLLOWED
                        ;CAREFULLY COULD LEAD TO AN EASY PINPOINTING OF THE FAULT

                        ;*****************************************************************
                        .SBTTL  BASIC DEFINITIONS

                        ;*INITIAL ADDRESS OF THE STACK POINTER *** 1100 ***
001100                  STACK=  1100
                        .EQUIV  EMT,ERROR       ;;BASIC DEFINITION OF ERROR CALL
                        .EQUIV  IOT,SCOPE       ;;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
                        .EQUIV  PS,PSW
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
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
                        .EQUIV  SW09,SW9
                        .EQUIV  SW08,SW8
                        .EQUIV  SW07,SW7
                        .EQUIV  SW06,SW6
                        .EQUIV  SW05,SW5
                        .EQUIV  SW04,SW4
                        .EQUIV  SW03,SW3
```

```
                                .EQUIV  SW02,SW2
                                .EQUIV  SW01,SW1
                                .EQUIV  SW00,SW0

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

                                ;*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
                                .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
                                .SBTTL  STARTING ADDRESS(ES)
000200  000137  001542          JMP     @#START ;;JUMP TO STARTING ADDRESS OF PROGRAM
```

```
                                .SBTTL  ACT11 HOOKS

                                ;;**********************************************************************
                                ;HOOKS REQUIRED BY ACT11
000204                                  $SVPC=.                 ;SAVE PC
000046                                  .=46
000046  005370                          $ENDAD                  ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .SCOP
000052                                  .=52
000052  000000                          .WORD   0               ;;2)SET LOC.52 TO ZERO
000204                                  .=$SVPC                 ;; RESTORE PC
```

```
                                .SBTTL  COMMON TAGS

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

                                        .=1100
001100                  $CMTAG:                         ;;START OF COMMON TAGS
001100  000000          $PASS:  .WORD   0               ;;CONTAINS PASS COUNT
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)
001160  000000          $REGAD: .WORD   0               ;;CONTAINS THE ADDRESS FROM
                                                        ;;WHICH  ($REG0) WAS OBTAINED
001162  000000          $REG0:  .WORD   0               ;;CONTAINS (($REGAD)+0)
001164  000000          $REG1:  .WORD   0               ;;CONTAINS (($REGAD)+2)
001166  000000          $REG2:  .WORD   0               ;;CONTAINS (($REGAD)+4)
001170  000000          $REG3:  .WORD   0               ;;CONTAINS (($REGAD)+6)
001172  000000          $REG4:  .WORD   0               ;;CONTAINS (($REGAD)+10)
001174  000000          $REG5:  .WORD   0               ;;CONTAINS (($REGAD)+12)
001176  000000          $REG6:  .WORD   0               ;;CONTAINS (($REGAD)+14)
001200  000000          $REG7:  .WORD   0               ;;CONTAINS (($REGAD)+16)
001202  000000          $REG10: .WORD   0               ;;CONTAINS (($REGAD)+20)
001204  000000          $REG11: .WORD   0               ;;CONTAINS (($REGAD)+22)
001206  000000          $TIMES: 0                       ;;MAX. NUMBER OF ITERATIONS
001210  000000          $ESCAPE:0                       ;;ESCAPE ON ERROR ADDRESS
001212  077             $QUES:  .ASCII  /?/             ;;QUESTION MARK
001213  015             $CRLF:  .ASCII  <15>            ;;CARRIAGE RETURN
001214  000012          $LF:    .ASCIZ  <12>            ;;LINE FEED
                                ;;**********************************************************************
001216  005015  047103  020124  MSG3:   .ASCIZ  <15><12>/CNT RDY DIDN'T SET/
```

```
                    001224  042122  020131  044504
                    001232  047104  052047  051440
                    001240  052105     000
                    001244                          .EVEN

                                            ;RK11 REGISTERS
                                            ;IF FOR ANY REASON THE REGISTER ADDRESSES ARE DIFFERENT FROM THESE
                                            ;(GIVEN BELOW), THE CONTENTS OF THE APPROPRIATE POINTERS SHOULD BE
                                            ;MODIFIED SO THAT THE CORRECT ADDRESS IS USED.
                                            ;
                                                    .EVEN
001244  177400                              RKDS:   177400
001246  177402                              RKER:   177402
001250  177404                              RKCS:   177404
001252  177406                              RKWC:   177406
001254  177410                              RKBA:   177410
001256  177412                              RKDA:   177412
001260  177416                              RKDB:   177416


                                            ;TAGS AND GENERAL DATA AREA
                                            ;
                                            ;
001262  000000                              FTITLE: 0               ;FLAG FOR PRINTING PROGRAM TITLE
001264  000000                              TIMER:  0               ;TIMER REGISTER
001266  000200                              RKPRI:  200             ;CONTAINS THE CPU LEVEL AT WHICH
                                                                    ;RK11 NORMALLY INTERRUPTS. THIS WORD
                                                                    ;SHOULD BE CHANGED IF RK11 IS DESINGATED
                                                                    ;A BR LEVEL OTHER THAN 5. E.G. IF IT IS CHANGED
                                                                    ;TO 6, THIS WORD SHOULD BE CHANGED TO 240.
001270  000220                              RKVEC:  220             ;CONTAINS THE NORMAL VECTOR ADDRESS TO WHICH
                                                                    ;RK11 INTERRUPTS. IF THIS IS NOT SO, CHANGE
                                                                    ;THIS WORD TO CONTAIN MODIFIED VECTOR ADDRESS.
```

```
                        .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

001272                  $ERRTB:



                        ;THE ERROR ITEMS TABLE CONSISTS OF ALL THE POSSIBLE ERROR MESSAGES
                        ;USED IN THIS PROGRAM. AN ERROR CALL IN THE PROGRAM CORRESPONDS TO
                        ;THE ITEM NUMBER IN THE ERROR TABLE. THUS 'ERROR        1' IN THE
                        ;PROGRAM CORRESPONDS TO 'ITEM   1' IN THE ERROR TABLE.
                        ;'EM###' IS THE  POINTER TO THE ERROR MESSAGE WHICH WILL BE TYPED
                        ;OUT IN CASE THAT ERROR WERE TO OCCUR. THUS FOR 'ERROR  1' THE ERROR
                        ;MESSAGE TYPE OUT WILL BE 'TIME OUT ON RK11 REG'.
                        ;'DH###' IS THE POINTER TO THE HEADER BLOCK WHICH WILL BE TYPED OUT
                        ;IMMEDIATELY AFTER THE ERROR MESSAGE.
                        ;'DT###'  SERVES AS A POINTER TO THE MEMORY LOCATIONS WHERE
                        ;THE INFORMATION RELEVANT TO THE ERROR TYPE OUTS (LIKE PC, CONTENTS
                        ;OF RKCS ETC.) WILL BE PICKED UP FROM.
                        ;THE LAST ROW CONTAINING '0' SERVES AS A TERMINATOR.
                        ;EXAMPLE:
                        ;IF ON RUNNING THIS PROGRAM A TIMEOUT WERE TO OCCUR ON ADDRESSING RKDS
                        ;(177400), BECAUSE OF SOME FAULT, THE FOLOWING TYPEOUT WOULD
                        ;OCCUR ON THE TELETYPE.
                        ;
                        ;       TIME OUT ON RK11 REG
                        ;         PC        REG
                        ;       ######    177400
                        ;
                        ;NOTE THAT ###### WOULD BE THE ACTUAL PC WHERE 'ERROR   1' IS LOCATED.

                        ;THE ERROR HANDLER IS LOCATED AT '$ERROR'. THE ERROR CALL IS AN 'EMT'
                        ;INSTRUCTION WITH ITS LOWER BYTE ENCODED TO PROVIDE INDEXING TO THE
                        ;ITEMS IN THE ERROR TABLE.
                        ;THUS 'ERROR    1'   IS 104001
                        ;     'ERROR    126' IS 104126   ETC.



                            ;ERROR ITEMS TABLE
```

```
                                ;ITEM   1
                                
001272  010316                          EM1     ;TIME OUT ON RK11 REG
001274  011526                          DH1     ;PC     REG
001276  011456                          DT1     ;$ERRPC $REG0
001300  000000                          0
                                
                                ;ITEM   2
                                
001302  010350                          EM2     ;REGISTER NOT CLEARED
001304  011546                          DH2     ;PC     REGADD  RECVD
001306  011464                          DT2     ;$ERRPC $REG0   $REG1
001310  000000                          0
                                
                                ;ITEM   3
                                
001312  010375                          EM3     ;RKCS ERROR
001314  011623                          DH3     ;PC     WROTE   READ
001316  011464                          DT2     ;$ERRPC $REG0   $REG1
001320  000000                          0
                                
                                ;ITEM   4
                                
001322  010410                          EM4     ;RKCS ERROR-ON WRITING READ ONLY BITS
001324  011575                          DH4     ;PC     EXPCT   RECVD
001326  011464                          DT2     ;$ERRPC $REG0   $REG1
001330  000000                          0
                                
                                ;ITEM   5
                                
001332  010455                          EM5     ;BUS INIT DID NOT CLEAR RKCS
001334  011650                          DH5     ;PC     RECVD
001336  011455                          DT1     ;$ERRPC $REG0
001340  000000                          0
                                
                                ;ITEM   6
                                
001342  010511                          EM6     ;'CNTRL RESET' DIDN'T CLEAR RKCS, ON SETING GO
001344  011650                          DH5     ;PC     RECVD
001346  011456                          DT1     ;$ERRPC $REG0
001350  000000                          0
                                
                                ;ITEM   7
                                
001352  010567                          EM7     ;'CNTRL RDY' DIDN'T SET AFTER CONTROL RESET
001354  012201                          DH30    ;PC     RKCS    RKER    RKDS
001356  011514                          DT26    ;$ERRPC $REG0   $REG1   $REG2
001360  000000                          0
                                
                                ;ITEM   10
                                
001362  010636                          EM10    ;REGISTER NOT CLEARED
001364  011546                          DH2     ;PC     REGADD  RECVD
001366  011464                          DT2     ;$ERRPC $REG0   $REG1
001370  000000                          0
```

```
                                ;ITEM   11
                                
001372  010663                          EM11    ;RKWC ERROR
001374  011666                          DH11    ;PC     WROTE   READ
001376  011464                          DT2     ;$ERRPC $REGO   $REG1
001400  000000                          0
                                
                                ;ITEM   12
                                
001402  011423                          EM43    ;UNEXPECTED RK11 INTERRUPT
001404  011713                          DH21    ;PC
001406  011510                          DT21    ;$ERRPC
001410  000000                          0
                                
                                ;ITEM   13
                                
001412  010676                          EM13    ;RKBA ERROR
001414  011666                          DH11    ;PC     WROT    READ
001416  011464                          DT2     ;$ERRPC $REGO   $REG1
001420  000000                          0
                                
                                ;ITEM   14
                                
001422  010711                          EM14    ;CNTRL RESET DID NOT CLEAR REGISTER
001424  011546                          DH2     ;PC     REGADD  RECVD
001426  011464                          DT2     ;$ERRPC $REGO   $REG1
001430  000000                          0
                                
                                ;ITEM   15
                                
001432  010753                          EM15    ;RKDA ERROR
001434  011666                          DH11    ;PC     WROTE   READ
001436  011464                          DT2     ;$ERRPC $REGO   $REG1
001440  000000                          0
                                
                                ;ITEM   16
                                
001442  011301                          EM26    ;RKCS ALTERERED ON CLEARING 'REG-BYTE'
001444  012067                          DH26    ;PC     REG-BYTE (RKCS)EXP (RKCS)RECVD
001446  011514                          DT26    ;$ERRPC $REGO   $REG1   $REG2
001450  000000                          0
                                
                                ;ITEM   17
                                
001452  010766                          EM17    ;BUS INIT DIDN'T CLEAR REGISTER
001454  011546                          DH2     ;PC     REGADD  RECVD
001456  011464                          DT2     ;$ERRPC $REGO   $REG1
001460  000000                          0
                                
                                ;ITEM   20
                                
001462  011022                          EM20    ;ADDRESSING ERROR-TRIED TO ADDRESS REG1, GOT REG2
001464  011720                          DH20    ;PC     REG1    REG2    (REG1)  (REG2)
001466  011474                          DT20    ;$ERRPC $REGO   $REG1   $REG2   $REG3
001470  000000                          0
```

```
                                ;ITEM   21

001472  011345                        EM27    ;TRIED TO CLEAR 'REG-BYTE', CHANGED 'REG-BYT2'
001474  012131                        DH27    ;PC     REG-BYT1        REG-BYT2        BYT2-EXPCT      BYT2-RECVD
001476  011474                        DT20    ;$ERRPC $REG0           $REG1           $REG2           $REG3
001500  000000                        0

                                ;ITEM   22

001502  011103                        EM22    ;DID NOT CLEAR RKCS LO BYTE
001504  011575                        DH4     ;PC     EXPCT   RECVD
001506  011464                        DT2     ;$ERRPC $REG0   $REG1
001510  000000                        0

                                ;IEM    23

001512  011136                        EM23    ;DID NOT CLEAR RKCS HI BYTE
001514  011575                        DH4     ;PC     EXPCT   RECVD
001516  011464                        DT2     ;$ERRPC $REG0   $REG1
001520  000000                        0

                                ;ITEM   24

001522  011172                        EM24    ;TRIED TO CLEAR RKCS 'BYTE', CHANGED 'REGIS'
001524  011767                        DH24    ;PC     BYTE    REGIS   (REG)EXP        (REG)RECVD
001526  011474                        DT20    ;$ERRPC $REG0   $REG1   $REG2   $REG3
001530  000000                        0

                                ;ITEM   25

001532  011246                        EM25    ;FAILED TO CLEAR 'REG-BYTE'
001534  012041                        DH25    ;PC     REG-BYTE RECVD
001536  011464                        DT2     ;$ERRPC $REG0   $REG1
001540  000000                        0


001542  000005          START:  RESET                   ;CLEAR THE BUS
                        .SBTTL  INITIALIZE THE COMMON TAGS
                        ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
001544  012706  001100          MOV     #$CMTAG,R6      ;;FIRST LOCATION TO BE CLEARED
001550  005026          1$:   CLR     (R6)+           ;;CLEAR MEMORY LOCATION
001552  022706  001140          CMP     #SWR,R6 ;;DONE?
001556  001374                  BNE     .-6             ;;LOOP BACK IF NO
001560  012706  001100          MOV     #STACK,SP       ;;SETUP THE STACK POINTER
                        ;;INITIALIZE A FEW VECTORS
001564  012737  005632  000020  MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
001572  012737  000340  000022  MOV     #340,@#IOTVEC+2 ;;LEVEL 7
001600  012737  006104  000030  MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
001606  012737  000340  000032  MOV     #340,@#EMTVEC+2 ;;LEVEL 7
001614  012737  010046  000034  MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
001622  012737  000340  000036  MOV     #340,@#TRAPVEC+2;LEVEL 7
001630  012737  010134  000024  MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
001636  012737  000340  000026  MOV     #340,@#PWRVEC+2 ;;LEVEL 7
001644  005037  001206          CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
```

```
                                        CLR     $ESCAPE         ;;CLEAR THE ESCAPE ON ERROR ADDRESS
                                        MOVB    #1,$ERMAX       ;;ALLOW ONE ERROR PER TEST
                                        MOV     #.,$LPADR       ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
                                        MOV     #.,$LPERR       ;;SETUP THE ERROR LOOP ADDRESS
                                ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                ;;EQUAL TO A "-1"  SETUP FOR A SOFTWARE SWITCH REGISTER.
                                        MOV     @#ERRVEC,-(SP)  ;;SAVE ERROR VECTOR
                                        MOV     #64$,@#ERRVEC   ;;SET UP ERROR VECTOR
                                        MOV     #DSWR,SWR       ;;SETUP FOR A HARDWARE SWICH REGISTER
                                        MOV     #DDISP,DISPLAY  ;;AND A HARDWARE DISPLAY REGISTER
                                        CMP     #-1,@SWR        ;;TRY TO REFERENCE HARDWARE SWR
                                        BNE     66$             ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                ;;AND  THE HARDWARE SWR IS NOT = -1
                                        BR      65$             ;;BRANCH IF NO TIMEOUT
                                64$:    MOV     #65$,(SP)       ;;SET UP FOR TRAP RETURN
                                        RTI
                                65$:    MOV     #SWREG,SWR      ;;POINT TO SOFTWARE SWR
                                        MOV     #DISPREG,DISPLAY
                                66$:    MOV     (SP)+,@#ERRVEC  ;;RESTORE ERROR VECTOR

                                .SBTTL  TYPE PROGRAM NAME
                                ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
                                        INC     #-1             ;;FIRST TIME?
                                        BNE     67$             ;;BRANCH IF NO
                                        TYPE    ,68$            ;;TYPE ASCIZ STRING

                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
                                        TST     @#42            ;;ARE WE RUNNING UNDER XXDP/ACT?
                                        BNE     69$             ;;BRANCH IF YES
                                        CMP     SWR,#SWREG      ;;SOFTWARE SWITCH REG SELECTED?
                                        BNE     70$             ;;BRANCH IF NO
                                        GTSWR                   ;;GET SOFT-SWR SETTINGS
                                        BR      70$
                                69$:    MOVB    #1,$AUTOB       ;;SET AUTO-MODE INDICATOR
                                70$:
                                        BR      67$             ;;GET OVER THE ASCIZ
                                ;;68$:  .ASCIZ  <CRLF>/RK11 LOGIC TEST I/<15><12>/MAINDEC-11-DZRKJ-D/<CRLF>
                                67$:
                                ;
                                START1: MOV     #BADTMO,@#4     ;SET TIME OUT VECTOR FOR UNEXPECTED
                                                                ;TIME OUTS
                                        MOV     #BADINT,@RKVEC  ;SET UP RK11 INTERRUPT VECTOR FOR
                                                                ;UNEXPECTED INTERRUPTS FROM RK11
                                        BR      TST1            ;GO TO TEST 1


                                ;THIS ROUTINE HANDLES UNEXPECTED TIME OUTS

                                BADTMO: MOV     (SP),R0 ;SAVE PC WHERE TIME OUT OCCURED
                                        TST     -(R0)
                                        CMP     (SP)+,(SP)+     ;RESTORE STACK POINTER
                                        TYPE    ,65$            ;;TYPE ASCIZ STRING
                                        BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <15><12>/UNEXPECTED TIME OUT AT PC=/
                                64$:
```

```
                                        MOV     R0,-(SP)        ;SET UP FOR TYPING OUT PC
                                        TYPOC                   ;GO TYPE OUT OCTAL PC
                                        HALT
                                        JMP     @#START

                                        ;THIS ROUTINE HANDLES UNEXPECTED INTERRUPTS FROM RK11
                                        ;SW 9 AND 10 FOR LOOPING ON ERROR
                                        ;AND LOOPING ON TEST IN WHICH TIMEOUT
                                        ;OCCURRED, ARE PROVIDED.

                                BADINT: MOV     (SP),R0         ;SAVE PC WHERE INTERRUPT OCCURED
                                        TST     -(R0)
                                        BIT     #20000,@SWR     ;INHIBIT ERROR TYPEOUT?
                                        BNE     1$              ;YES, DON'T TYPE OUT
                                        TYPE
                                        $CRLF
                                        TYPE
                                        EM43                    ;TYPE 'UNEXPEXTED RK11 INTERRUPT'
                                                                ;TYPE ' AT PC='
                                        TYPE    ,65$            ;;TYPE ASCIZ STRING
                                        BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  / AT PC=/
                                64$:
                                        MOV     R0,-(SP)        ;SET UP FOR TYPING OUT PC
                                        TYPOC                   ;GO TYPE OCTAL PC WHERE BAD
                                                                ;INTERUPT OCCURED
                                1$:     BIT     #1000,@SWR      ;LOOP ON ERROR?
                                        BEQ     2$              ;NO, BRANCH
                                        CMP     (SP)+,(SP)+     ;YES, REPOSITION STACK
                                        JMP     @$LPADR         ;GO TO THE STARTING ADDRESS OF
                                                                ;THE TEST THAT GAVE UNEXPECTED INTERRUPT
                                2$:     BIT     #40000,@SWR     ;LOOP ON TEST?
                                        BEQ     3$              ;NO, BRANCH
                                        RTI                     ;YES, LOOP. GO BACK WHER U INTERRUPTED FROM.
                                3$:     HALT                    ;UNEXPECTED INTERRUPT OCCURED AS
                                                                ;INDICATED IN THE TYPE OUT.U CAN LOOP
                                                                ;ON ERROR, TEST,OR INHIBIT TYPEOUT BY
                                                                ;SETTING APPROPRIATE SWITCHES.
                                        JMP     @#START         ;GO BACK TO THE START OF THE
                                                                ;PROGRAM. THUS PRESSING CONTINUE
                                                                ;AFTER THE ABOVE HALT WILL
                                                                ;RESTART THE PROGRAM


                                ;RESTART AFTER POWER FAIL
                                ;THE PROGRAM WOULD RESTART HERE IF POWER CAME BACK AFTER A FALIURE.

                                PFSTRT:
                                        TYPE    ,65$            ;;TYPE ASCIZ STRING
                                        BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <15><12>/PWR UP,RESTART/
                                64$:
                                        CLR     R0
                                        CLR     R1
                                1$:     INC     R1
```

```
                002344  001376                  BNE     1$
                002346  105200                  INCB    R0
                002350  001374                  BNE     1$

                                        ;;**********************************************************************
                                        ;*TEST 1         TEST THAT ALL RK11 REGISTERS CAN BE REFERENCED
                                        ;*THIS TEST CHECKS IF EVERY RK11 REGISTER CAN BE
                                        ;*REFERENCED WITHOUT TIMING OUT.  IF A TIME OUT OCCURS THE ERROR IS
                                        ;*REPORTED & AN ERROR FLAG (R2) IS INCREMENTED.  IT THERE WAS
                                        ;*AN ERROR DURING THIS TEST, THE ENTIRE PROGRAM IS ABORTED
                                        ;;**********************************************************************
                002352  000004          TST1:   SCOPE
                002354  005002                  CLR     R2
                002356  012737  002402  001110  MOV     #T1,$LPERR      ;SET RETURN ADRES FOR LUP
                                                                        ;ON ERROR (SW9)
                002364  012737  002456  000004  MOV     #TIMOUT,@#4     ;SET UP ADDRESS FOR TIMEOUT VECTOR
                002372  012700  177771          MOV     #-7,R0          ;INITIALIZE R0 TO KEEP TRACK OF REGIS REFERENCED
                002376  012701  001244          MOV     #RKDS,R1        ;INITIALIZE R1 WITH RKDS ADDRESS
                002402  005731          T1:     TST     @(R1)+          ;REFERENCE THE REGISTER,
                                                                        ;IF IT CAN'T BE, TIMEOUT TRAP WILL OCCUR
                                                                        ;SHIFT THE POINTER
                002404  005200                  INC     R0              ;ALL REGISTERS REFERENCED?
                002406  001375                  BNE     T1              ;IF NOT, LOOP BACK & REFERENCE NEXT REGISTER
                002410  012737  002116  000004  MOV     #BADTMO,@#4
                002416  005702                  TST     R2              ;WAS THERE AN ERROR?
                002420  001415                  BEQ     1$              ;NO,BRANCH
                002422  104401  002430          TYPE    ,65$            ;;TYPE ASCIZ STRING
                002426  000410                  BR      64$             ;;GET OVER THE ASCIZ
                                        ;;65$:  .ASCIZ  <15><12>/PROG ABORTED/
                002450                  64$:
                002450  000137  005360          JMP     $GET42          ;IF YES, ABORT THIS ENTIRE TEST
                002454                  1$:
                002454  000407                  BR      TST2            ;;EXIT
                002456  022626          TIMOUT: CMP     (SP)+,(SP)+
                002460  014137  001162          MOV     -(R1),$REGD     ;GET ADDRESS OF REGISTER THAT TIMED OUT
                002464  104001                  ERROR   1               ;TIMED OUT WHEN REFERENCING RK11
                                                                        ;REGISTERS
                002466  005721                  TST     (R1)+           ;REPOSITION POINTER TO THE NEXT REGISTER ADDRESS
                002470  005202                  INC     R2              ;SET FLAG INDICATING ERROR
                002472  000744                  BR      T1+2            ;BRANCH BACK & REFERENCE THE NXT REGISTER
                                                ;

                                        ;;**********************************************************************
                                        ;*TEST 2         CHECK RK11 INITIALIZATION
                                        ;*THIS TEST CHECKS THAT THE CONTROLLER LOGIC IS INITIALIZED
                                        ;*CORRECTLY, RKWC, RKDA, RKDA, RKDB SHOULD BE CLEAR AND
                                        ;*RKCS SHOULD HAVE 'CNTRL RDY' BIT SET.
                                        ;;**********************************************************************
                002474  000004          TST2:   SCOPE
                002476  000005                  RESET                   ;ISSUE A BUS INIT
                002500  012700  177772          MOV     #-6,R0          ;SET COUNT FOR 6 REGISTERS
                002504  012701  001246          MOV     #RKER,R1        ;INITIALIZE ADRES
                002510  023711  001250  1$:     CMP     RKCS,@R1        ;IS IT RKCS?
                002514  001005                  BNE     2$              ;NO
                002516  022771  000200  000000  CMP     #200,@(R1)      ;ONLY BIT 7 OF RKCS SHOULD BE SET!
```

```
                                        BNE     3$              ;BRANCH IF ERROR
                                        BR      4$
002530  005771  000000        2$:     TST     @(R1)           ;CHECK THAT REST OF REGISTERS
                                                                ;ARE CLEAR
002534  001406                        BEQ     4$              ;BRANCH IF REGISTER IS CLR
002536  011137  001162        3$:     MOV     @R1,$REGO       ;GET ADRES OF REGISTER
002542  017137  000000  001164        MOV     @(R1),$REG1     ;GET CONTENTS OF REGISTER
002550  104002                        ERROR   2               ;RK11 REGISTER WAS FOUND TO B NOT CLEAR
002552  005721                4$:     TST     (R1)+           ;INCREMNT POINTER TO NXT REG
002554  005200                        INC     R0              ;CHKD ALL REGS?
002556  001354                        BNE     1$              ;IF NOT LUP BAK
                                ;
                                ;;*****************************************************************
                                ;*TEST 3         TEST RKCS FUNCTION BITS - 1,2,3
                                        ;*THIS TEST CHECKS IF THE FUNCTION BIT-1,2,3 IN RKCS CAN BE WRITTEN &
                                        ;*READ BACK.  R0 CONTAINS THE BIT THAT WAS WRITTEN.  R1 CONTAINS THE
                                        ;*BITS THAT WAS/WERE READ BACK.
                                ;;*****************************************************************
002560  000004                TST3:   SCOPE
002562  012737  002574  001110        MOV     #1$,$LPERR      ;SET RETURN ADRES FOR LOPING ON
                                                                ;ERROR (SW 9)
002570  012700  000002                MOV     #2,R0           ;INITIALIZE BIT TO BE WRITTEN IN RKCS
002574  010077  176450        1$:     MOV     R0,@RKCS        ;WRITE THAT BIT IN RKCS
002600  017701  176444                MOV     @RKCS,R1        ;GET RKCS
002604  042701  000200                BIC     #200,R1         ;MASK OUT CNTRL RDY BIT
002610  020001                        CMP     R0,R1           ;WAS RECVD BIT SAME AS WRITTEN BIT
002612  001406                        BEQ     2$              ;YES, BRANCH OTHERWISE REPORT ERROR
002614  010037  001162                MOV     R0,$REGO        ;GET EXPCTD RKCS, BIT THAT WAS WRITTEN
002620  017737  176424  001164        MOV     @RKCS,$REG1     ;GET RECVD RKCS, BIT READ
002626  104003                        ERROR   3               ;BIT THAT WAS WRITTEN WAS NOT READ BACK
002630  006300                2$:     ASL     R0              ;SHIFT TO WRITE NEXT BIT
002632  020027  000020                CMP     R0,#20          ;HAVE U CHKD ALL 3 BITS 1, 2, 3
002636  001356                        BNE     1$              ;IF NOT, LOOP BACK & CHECK THE NXT BIT
                                ;
                                ;;*****************************************************************
                                ;*TEST 4         TEST RKCS EXTENDED MEMORY BITS - 4,5
                                        ;*THIS TEST CHECKS IF THE 'EXTENDED MEMORY' BITS CAN BE WRITTEN
                                        ;*AND READ BACK CORRECTLY.
                                ;;*****************************************************************
002640  000004                TST4:   SCOPE
002642  012737  002654  001110        MOV     #1$,$LPERR      ;SET RETURN ADRES FOR LUPING
                                                                ;ON EROR (SW 9)
002650  012700  000020                MOV     #20,R0          ;INITIALIZE BIT TO BE WRITTEN IN RKCS
002654  010077  176370        1$:     MOV     R0,@RKCS        ;WRITE THAT BIT IN RKCS
002660  017701  176364                MOV     @RKCS,R1        ;GET RKCS
002664  042701  000200                BIC     #200,R1         ;MASK OUT CNTRL RDY BIT
002670  020001                        CMP     R0,R1           ;WAS RECVD BIT SAME AS WRITTEN BIT
002672  001406                        BEQ     2$              ;YES, BRANCH
002674  010037  001162                MOV     R0,$REGO        ;GET EXPCTD RKCS, BIT THAT WAS WRITTEN
002700  017737  176344  001164        MOV     @RKCS,$REG1     ;GET RKCD RECVD, BIT THAT WAS READ
002706  104003                        ERROR   3               ;BIT THAT WAS WRITTEN WAS NOT READ BACK
002710  006300                2$:     ASL     R0              ;SHIFT TO WRITE NEXT BIT
002712  022700  000100                CMP     #100,R0         ;HAVE U CHKD BOTH BITS, 4, 5
002716  001356                        BNE     1$              ;IF NOT, LOOP BACK & CHECK THE NXT BIT
                                ;
                                ;;*****************************************************************
```

```
                                ;*TEST 5        CHECK RKCS IDE BIT - 6
                                ;*THIS TEST CHECKS IF IDE BIT CAN BE WRITTEN & READ BACK.  THE PROCESSOR
                                ;*STATUS IS SET AT PRIORITY 7 SO THAT UNWANTED INTERRUPTS ARE LOCKED OUT.
                                ;*THEN IDE BIT IS CLEARED AND PROCESSOR PRIORITY IS LOWERED TO INSURE THAT
                                ;*NO INTERRUPTS OCCUR.
                                ;;*****************************************************************
002720  000004         TST5:   SCOPE
002722  012746  000340          MOV     #340,-(SP)
002726  012746  002734          MOV     #64$,-(SP)
002732  000002                  RTI
002734                  64$:
002734  013700  001250          MOV     RKCS,R0
002740  012710  000100          MOV     #100,@R0        ;SET THE IDE BIT
002744  022710  000300          CMP     #300,@R0        ;WAS IT WRITTEN CORRECTLY
002750  001406                  BEQ     1$              ;YES, BRANCH OTHERWISE REPORT ERROR
002752  012737  000300  001162  MOV     #300,$REG0      ;GET EXPCT RKCS
002760  011037  001164          MOV     @R0,$REG1       ;GET RKCS RECVD
002764  104003                  ERROR   3               ;IDE BIT WAS WRITTEN, BUT WAS NOT
                                                        ;READ BACK
002766  104412          1$:     CNT.RESET               ;CONTROL RESET, CLEAR IDE
                                                        ;THIS IS A CALL FOR THE 'CNTRL-
                                                        ;RESET' ROUTINE.  A CONTROL RESET
                                                        ;IS ISSUED AND AFTER A CERTAIN TIME
                                                        ;IF THE 'CNTRL RDY' DOES NOT SET
                                                        ;AN ERROR IS REPORTED. NOTE THAT
                                                        ;THE PC IN ERROR MESSAGE IS THE
                                                        ;PC WHERE 'CNT.RESET' IS LOCATED.
002770  022710  000200          CMP     #200,@R0        ;DID IDE BIT GET CLRD'
002774  001406                  BEQ     2$              ;YES, BRANCH
002776  010037  001162          MOV     R0,$REG0        ;GET ADRES OF RKCS
003002  011037  001164          MOV     @R0,$REG1       ;GET RKCS
003006  104010                  ERROR   10              ;IDE BIT COULD NOT B CLRD
003010  000430                  BR      TST6            ;;EXIT
003012  104414  000010  2$:     DELAY   ,10             ;WAIT FOR AT LEAST 60 US
                                                        ;ON 11/20  12 US FOR 11/45
003016  012777  003066  176244  MOV     #3$,@RKVEC      ;SET RK11 INTERUPT VECTOR TO
                                                        ;WHICH RK11 CAN INTERUPT IF THERE
                                                        ;IS FAULTY LOGIC
003024  013746  001266          MOV     RKPRI,-(SP)     ;LOWER CPU PRIORITY SO THAT
003030  012746  003036          MOV     #4$,-(SP)       ;RK11 CAN POSSIBLY INTERRUPT IF
003034  000002                  RTI                     ;THER IS MALFUNCTIONINIG LOGIC
003036  000240          4$:     NOP                     ;THE INTERRUPT WOULD OCCUR
003040  000240                  NOP
003042  000240                  NOP
003044  012746  000340          MOV     #340,-(SP)
003050  012746  003056          MOV     #65$,-(SP)
003054  000002                  RTI
003056                  65$:
                                                        ;PRIORITY
003056  012777  002202  176204  MOV     #BADINT,@RKVEC  ;SET UP UNEXPCTD INTRUPT VECTOR
003064  000402                  BR      TST6            ;;EXIT
003066  022626          3$:     CMP     (SP)+,(SP)+     ;RESTORE STACK
003070  104012                  ERROR   12              ;AN UNEXPECTED RK11 INTERRUPT
                                                        ;OCCURED PROBABLY DUE TO FAULTY LOGIC
                        ;
```

```
                                ;;*****************************************************************
                                ;*TEST 6         CHECK RKCS SSE,EXB,FMT,IBA BITS - 8,9,10,11
                                        ;*THIS TEST CHECKS IF THE SSE, EXB, FMT & IBA BITS CAN BE WRITTEN
                                        ;*AND READ BACK CORRECTLY
                                ;;*****************************************************************
003072  000004                TST6:   SCOPE
003074  012701  000400                MOV     #400,R1         ;INITIALIZE BIT TO BE WRITTEN IN RKCS
003100  013702  001250                MOV     RKCS,R2
003104  010112                1$:     MOV     R1,@R2          ;WRITE THAT BIT IN RKCS
003106  011200                        MOV     @R2,R0          ;GET RKCS
003110  042700  000200                BIC     #200,R0         ;MASK CNTRL RDY BIT
003114  020100                        CMP     R1,R0           ;WAS THE READ BIT SAME AS THE
                                                            ;WRITTEN BIT
003116  001405                        BEQ     2$              ;YES BRANCH, OTHERWISE REPORT ERROR
003120  010137  001162                MOV     R1,$REG0        ;GET EXPTD RKCS
003124  011237  001164                MOV     @R2,$REG1       ;GET RECVD RKCS
003130  104003                        ERROR   3               ;BIT THAT WAS WRITTEN (AS IN $REG0)
                                                            ;WAS NOT READ BACK
003132  006301                2$:     ASL     R1              ;SHIFT TO WRITE NEXT BIT
003134  022701  010000                CMP     #10000,R1       ;HAVE U CHECKED ALL BITS 8, 9, 10, 11
003140  001361                        BNE     1$              ;IF NOT, LOOP BACK & CHECK THE NXT BIT

                                ;
                                ;;*****************************************************************
                                ;*TEST 7         CHECK READ ONLY BITS OF RKCS
                                        ;*THIS TEST CHECKS THAT TRYING TO SET THE UNUSED BIT OR THE READ ONLY
                                        ;*BITS DOES NOT SET THEM OR AFFECT ANY OTHER BITS IN RKCS
                                ;;*****************************************************************
003142  000004                TST7:   SCOPE
003144  013700  001250                MOV     RKCS,R0
003150  012710  170200                MOV     #170200,@R0     ;TRY SETTING THE UNUSED BIT & RD
                                                            ;ONLY BITS
003154  011001                        MOV     @R0,R1          ;GET RKCS
003156  042701  010000                BIC     #10000,R1       ;MASK BIT 12
003162  022701  000200                CMP     #200,R1         ;IS 'RDY' BIT SET? NO OTHER
                                                            ;BIT SHOULD BE SET.
003166  001406                        BEQ     TST10           ;;OK, EXIT
003170  012737  000200  001162        MOV     #200,$REG0      ;GET EXPCTD RKCS
003176  011037  001164                MOV     @R0,$REG1       ;GET RKCS RECVD
003202  104004                        ERROR   4               ;TRIED TO SET UNUSED & RD ONLY BITS
                                                            ;OF RKCS
                                                            ;SHOULD NOT HAVE AFFECTED ANY BITS

                                ;
                                ;;*****************************************************************
                                ;*TEST 10        CHECK THAT 'GO' BIT (0) CAN BE SET
                                        ;*THIS TEST CHECKS THAT THE 'GO' BIT CAN BE SET, BY PERFORMING
                                        ;*CONTROL RESET & SEEING THAT THE EXB & IBA SET PREVIOUSLY
                                        ;*WERE CLEARED.
                                ;;*****************************************************************
003204  000004                TST10:  SCOPE
003206  012746  000340                MOV     #340,-(SP)
003212  012746  003220                MOV     #64$,-(SP)
003216  000002                        RTI
003220                          64$:
003220  013701  001250                MOV     RKCS,R1
```

```
                                        MOV     #7576,@R1       ;SET ALL BITS EXCEPT GO
                                        CLR     R0              ;
                                        RESET                   ;ISSUE BUS INIT
                                        CMP     #200,@R1        ;CHECK IF RKCS WAS CLEARED?
                                        BEQ     1$              ;YES, BRANCH OTHERWISE REPORT ERROR
                                        MOV     @R1,,$REGO      ;GET RKCS
                                        ERROR   5               ;BUS INIT DID NOT CLEAR RKCS
                                1$:     MOV     #5000,@R1       ;SET IBA & EXB IN RKCS
                                        INC     @R1             ;SET GO, CONTROL RESET
                                2$:     INC     R0              ;KEEP TIME
                                        TSTB    R0              ;HAVE U WAITED LONG FOR CNTRL RDY
                                                                ;TO SET?
                                        BMI     3$              ;IF YES, BRANCH & REPORT ERROR
                                        TSTB    @R1             ;WAS CNTRL RDY SET?
                                        BPL     2$              ;IF NOT LOOP BACK & WAIT FOR IT
                                        CMP     #200,@R1        ;IF CNTRL RDY WAS SET, CHK IF 'CNTRL
                                                                ;RESET' CLEARED IBA & EXB BITS
                                        BEQ     TST11           ;;IF YES, EXIT. OTHERWISE ERROR
                                        MOV     @R1,$REGO       ;GET RKCS
                                        ERROR   6               ;GO BIT COULD NOT BE SET OR FAULT IN
                                                                ;THE 'INIT L' GENERATING LOGIC
                                        BR      TST11           ;;EXIT
                                3$:     JSR     PC,GT3RG        ;GO, GET RKCS, ER, DS
                                        ERROR   7               ;CONTROL READY DID NOT SET AFTER
                                                                ;CONTROL RESET
                                ;
                                ;;*****************************************************************
                                ;*TEST 11       CHECK RKCS WITH A COUNT PATTERN
                                        ;*THIS TEST CHECKS THAT RKCS CAN BE CLEARED FROM 7576 THEN A COUNT
                                        ;*PATTERN FROM 2 TO 7777 IS RUN. NOTE: ALL PATTERNS WITH BIT 0 SET
                                        ;*(GO BIT) ARE AVOIDED SO THAT RK11 MAY NOT START AN UNDESIRED OPERATION.
                                        ;*R1 CONTAINS THE COUNT PATTERN THAT WAS WRITTEN
                                ;;*****************************************************************
003314  000004                TST11:  SCOPE
003316  012737  000010  001206        MOV     #10,$TIMES      ;;DO 10 ITERATIONS
003324  012746  000340                MOV     #340,-(SP)
003330  012746  003336                MOV     #64$,-(SP)
003334  000002                        RTI
003336                          64$:
003336  013700  001250                MOV     RKCS,R0
003342  012710  007576                MOV     #7576,@R0       ;SET ALL BITS IN RKCS EXCEPT GO
003346  005010                        CLR     @R0             ;CLEAR RKCS
003350  022710  000200                CMP     #200,@R0        ;WAS IT CLEARED
003354  001405                        BEQ     1$              ;YES,BRANCH
003356  010037  001162                MOV     R0,$REGO        ;GET ADRES OF RKCS
003362  011037  001164                MOV     @R0,$REG1       ;NO, GET RKCS
003366  104010                        ERROR   10              ;RKCS COULD NOT BE CLEARED
003370  012701  000002        1$:     MOV     #2,R1           ;WRITE THIS BIT IN RKCS
003374  012737  003406  001110        MOV     #2$,$LPERR
003402  012705  177773                MOV     #-5,R5
003406  010110                2$:     MOV     R1,@R0          ;WRITE IT
003410  010102                        MOV     R1,R2           ;GET BIT THAT WAS WRITTEN
003412  052702  000200                BIS     #200,R2         ;SET CNTRL RDY BIT
003416  011003                        MOV     @R0,R3
003420  020203                        CMP     R2,R3           ;WAS THAT BIT WRITTEN CORRECTLY?
003422  001407                        BEQ     3$              ;YES, BRANCH
```

```
                                ;CHECK RKCS WITH A COUNT PATTERN
003424  010237  001162          MOV     R2,$REGO        ;GET EXPCTD WORD
003430  010337  001164          MOV     R3,$REG1        ;GET RKCS RECVD
003434  104003                  ERROR   3               ;DID NOT READ BAK THE BIT THAT
                                                        ;WAS WRITTEN
003436  005205                  INC     R5
003440  001405                  BEQ     TST12           ;;EXIT
003442  062701  000002  3$:     ADD     #2,R1           ;GENERATE NXT PATTERN TO BE WRITTEN
003446  022701  010000          CMP     #10000,R1       ;ALL PATTERNS WRITTEN?
003452  001355                  BNE     2$              ;IF NOT, LUP BAK & CHK NXT PATTERN
                                ;
                                ;;*****************************************************************
                                ;*TEST 12       CHECK THAT RKWC BIT 0-15 CAN BE SET
                                ;*THIS TEST FLOATS A '1' THROUGH RKWC BIT 0-15 AND CHECKS THAT IT
                                ;*CAN BE READ BACK CORRECTLY. R0 CONTAINS THE WORD THAT IS WRITTEN
                                ;;*****************************************************************
003454  000004          TST12:  SCOPE
003456  012700  000001          MOV     #1,R0           ;INITIALIZE R0 FOR THE BIT TO BE WRITTEN IN RKWC
003462  013701  001252          MOV     RKWC,R1
003466  012737  003474  001110  MOV     #1$,$LPERR      ;SET UP RETURN ADRES FOR
                                                        ;LUPING ON ERROR (SW 9)
003474  010011          1$:     MOV     R0,@R1          ;WRITE THAT BIT IN RKWC
003476  011102                  MOV     @R1,R2
003500  020002                  CMP     R0,R2           ;WAS IT WRITTEN CORRECTLY
003502  001405                  BEQ     2$              ;YES, BRANCH. OTHERWISE, ERROR
003504  010037  001162          MOV     R0,$REGO        ;GET EXPCTD RKWC BIT THAT WAS WRITTEN
003510  010237  001164          MOV     R2,$REG1        ;GET RKWC (THAT WAS READ BACK)
003514  104011                  ERROR   11              ;DID NOT READ BACK THE BIT THAT
                                                        ;WAS WRITTEN IN RKWC
003516  006300          2$:     ASL     R0              ;SHIFT TO WRITE THE NXT BIT
003520  001365                  BNE     1$              ;IF ALL TME BITS HAVE NOT BEEN
                                                        ;DONE, LOOP BACK
                                ;
                                ;;*****************************************************************
                                ;*TEST 13       CHECK RKWC WITH A COUNT PATTERN
                                ;*THIS TEST CHECKS THAT RKWC CAN BE CLEARED, THEN A COUNT PATTERN
                                ;*FROM 0 TO 177777 IS WRITTEN & CHECKED IF IT WAS WRITTEN CORRECTLY
                                ;;*****************************************************************
003522  000004          TST13:  SCOPE
003524  012737  000010  001206  MOV     #10,$TIMES      ;;DO 10 ITERATIONS
003532  013700  001252          MOV     RKWC,R0
003536  012710  177777          MOV     #177777,@R0     ;SET ALL BITS IN RKWC
003542  005010                  CLR     @R0             ;CLEAR RKWC
003544  005710                  TST     @R0             ;WAS IT CLEARED?
003546  001405                  BEQ     1$              ;YES, BRANCH
003550  010037  001162          MOV     R0,$REGO        ;GET ADRES OF RKWC
003554  011037  001164          MOV     @R0,$REG1       ;NO, GET RKWC
003560  104010                  ERROR   10              ;RKWC COULD NOT BE CLEARED
                                ;
003562  005001          1$:     CLR     R1              ;INITIALIZE COUNT PATTERN
003564  012705  177773          MOV     #-5,R5
003570  012737  003576  001110  MOV     #2$,$LPERR
003576  010110          2$:     MOV     R1,@R0          ;WRITE THE PATTERN IN THE REGISTER
003600  011002                  MOV     @R0,R2
003602  020102                  CMP     R1,R2           ;WAS IT WRIITEN CORRRECTLY?
003604  001407                  BEQ     3$              ;YES, BRANCH
003606  010137  001162          MOV     R1,$REGO        ;GET EXPECTED WORD
```

```
000612  010237  001164          MOV     R2,$REG1        ;GET WORD THAT WAS RECVD
000616  104011                  ERROR   11              ;DID NOT READ BACK THE PATTERN THAT
                                                        ;WAS WRITTEN INTO THE REGISTER
000620  005205                  INC     R5
000622  001402                  BEQ     TST14           ;;EXIT
000624  005201          3$:     INC     R1              ;INCREMENT COUNT PATTERN
000626  001363                  BNE     2$              ;LUP BAK & WRITE NXT PATTERN IF NOT
                                                        ;DONE WITH ALL
                        ;
                        ;;******************************************************************
                        ;*TEST 14       CHECK THAT RKBA CAN BE SET
                                ;*THIS TEST FLOATS A '1' THROUGH RKBA BITS 0-15 AND CHECKS THAT
                                ;*IT CAN BE READ BACK CORRECTLY. R0 CONTAINS THE WORD THAT WAS
                                ;*WRITTEN.
                        ;;******************************************************************
000630  000004          TST14:  SCOPE
000632  012700  000001          MOV     #1,R0           ;INITIALIZE R0 FOR BIT TO BE
                                                        ;WRITTEN IN RKBA
000636  013701  001254          MOV     RKBA,R1
000642  012737  003650  001110  MOV     #1$,$LPERR      ;SET UP RETURN ADRES FOR
                                                        ;LUPING ON EROR (SW 12)
003650  010011          1$:     MOV     R0,@R1          ;WRITE THAT BIT IN RKBA
003652  011102                  MOV     @R1,R2
003654  020002                  CMP     R0,R2           ;WAS IT WRITTEN CORRECTLY?
003656  001405                  BEQ     2$              ;YES, BRANCH
003660  010037  001162          MOV     R0,$REG0        ;GET EXPCTD RKBA (BIT WRITTEN)
003664  010237  001164          MOV     R2,$REG1        ;GET RKBA (BIT READ BACK)
003670  104013                  ERROR   13              ;DID NOT READ BACK TME BIT THAT
                                                        ;WAS WRITTEN IN RKBA
003672  006300          2$:     ASL     R0              ;SHIFT R0, TO WRITE NEXT BIT
003674  001365                  BNE     1$              ;IF ALL BITS ARE NOT CHKD, LOOP
                                                        ;BACK & WRITE NXT BIT
                        ;
                        ;;******************************************************************
                        ;*TEST 15       CHECK RKBA WITH A COUNT PATTERN
                                ;*THIS TEST CHECKS THAT RKBA CAN BE CLEARED, THEN IT RUNS A COUNT
                                ;*PATTERN FROM 0 TO 177777. R1 CONTAINS THE COUNT PATTERN TO BE WRITTEN.
                        ;;******************************************************************
003676  000004          TST15:  SCOPE
003700  012737  000010  001206  MOV     #10,$TIMES      ;;DO 10 ITERATIONS
003706  013700  001254          MOV     RKBA,R0
003712  012710  177777          MOV     #177777,@R0     ;SET ALL BITS IN RKBA
003716  005010                  CLR     @R0             ;CLEAR RKBA
003720  005710                  TST     @R0             ;WAS IT CLEARED?
003722  001405                  BEQ     1$              ;YES, BRANCH
003724  010037  001162          MOV     R0,$REG0        ;GET ADRES OF RKBA
003730  011037  001164          MOV     @R0,$REG1       ;NO, GET RKBA
003734  104010                  ERROR   10              ;RKBA COULD NOT BE CLEARED

003736  005001          1$:     CLR     R1              ;INITIALIZE COUNT PATTERN
003740  012705  177773          MOV     #-5,R5
003744  012737  003752  001110  MOV     #2$,$LPERR
003752  010110          2$:     MOV     R1,@R0          ;WRITE THE PATTERN IN THE
                                                        ;REGISTER
003754  011002                  MOV     @R0,R2
003756  020102                  CMP     R1,R2           ;WAS IT WRIITEN CORRRECTLY?
```

```
MAINDEC-11-DZRKJ-D     MACY11 27(1006)  04-OCT-76  13:08  PAGE 20
DZRKJD.P11    30-AUG-76 14:48           T15     CHECK RKBA WITH A COUNT PATTERN                                  SEQ 0033

  998  003760  001407                          BEQ     3$              ;YES, BRANCH
  999  003762  010137  001162                  MOV     R1,$REG0        ;GET EXPECTED WORD
 1000  003766  011037  001164                  MOV     @R0,$REG1       ;GET WORD THAT WAS RECVD
 1001  003772  104013                          ERROR   13              ;DID NOT READ BACK THE PATTERN THA
 1002                                                                  ;WAS WRITTEN INTO THE REGISTER
 1003  003774  005205                          INC     R5
 1004  003776  001402                          BEQ     TST16           ;EXIT
 1005  004000  005201                  3$:     INC     R1              ;INCREMENT COUNT PATTERN
 1006  004002  001363                          BNE     2$              ;LUP BAK & WRITE NXT PATTERN IF NOT
 1007                                                                  ;DONE WITH ALL
 1008                                  ;
 1009                                  ;;*****************************************************************
 1010                                  ;*TEST 16        CHECK THAT RKDA CAN BE SET
 1011                                          ;*THIS TEST FLOATS A '1' THROUGH RKDA AND CHECKS THAT THE WORD WHICH WAS
 1012                                          ;*WRITTEN IS READ BACK CORRECTLY
 1013                                  ;;*****************************************************************
 1014  004004  000004                  TST16:  SCOPE
 1015  004006  005077  175236                  CLR     @RKCS           ;CLEAR RKCS
 1016  004012  012700  000001                  MOV     #1,R0           ;INITIALIZE R0 FOR BIT TO BE WRITTEN IN RKDA
 1017  004016  013701  001256                  MOV     RKDA,R1
 1018  004022  012737  004030  001110          MOV     #1$,$LPERR      ;SET UP RETURN ADRES
 1019                                                                  ;FOR LUPING ON EROR
 1020  004030  010011                  1$:     MOV     R0,@R1          ;WRITE THAT BIT IN RKDA
 1021  004032  011102                          MOV     @R1,R2
 1022  004034  020002                          CMP     R0,R2           ;WAS IT WRITTEN CORRECTLY
 1023  004036  001405                          BEQ     2$              ;YES, BRANCH
 1024  004040  010037  001162                  MOV     R0,$REG0        ;NO, GET EXPCTD RKDA (BIT WRITTEN)
 1025  004044  010237  001164                  MOV     R2,$REG1        ;GET RKDA (BIT READ BACK)
 1026  004050  104015                          ERROR   15              ;DID NOT READ BACK THE BIT THAT
 1027                                                                  ;WAS WRITTEN IN RKDA
 1028  004052  006300                  2$:     ASL     R0              ;SHIFT R0, TOWRITE NXT BIT
 1029  004054  001365                          BNE     1$              ;IF ALL BITS ARE NOT CHKD, LOOP
 1030                                                                  ;BACK & WRITE NXT BIT
 1031                                  ;
 1032                                  ;;*****************************************************************
 1033                                  ;*TEST 17        CHECK RKDA WITH A COUNT PATTERN
 1034                                          ;*THIS TEST CHECKS THAT RKDA CAN BE CLEARED, THEN A COUNT PATTERN
 1035                                          ;*FROM 0 TO 177777 IS WRITTEN AND CHECKED. R1 CONTAINS THE COUNT
 1036                                          ;*PATTERN TO BE WRITTEN
 1037                                  ;;*****************************************************************
 1038  004056  000004                  TST17:  SCOPE
 1039  004060  012737  000010  001206          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
 1040  004066  013700  001256                  MOV     RKDA,R0
 1041  004072  012710  177777                  MOV     #177777,@R0     ;SET ALL BITS IN RKDA
 1042  004076  005010                          CLR     @R0             ;CLEAR RKDA
 1043  004100  005710                          TST     @R0             ;WAS IT CLEARED?
 1044  004102  001405                          BEQ     1$              ;YES, BRANCH
 1045  004104  010037  001162                  MOV     R0,$REG0        ;GET ADRES OF RKDA
 1046  004110  011037  001164                  MOV     @R0,$REG1       ;GO, GET RKDA
 1047  004114  104010                          ERROR   10              ;RKDA COULD NOT BE CLEARED
 1048
 1049  004116  005001                  1$:     CLR     R1              ;INITIALIZE COUNT PATTERN
 1050  004120  012705  177773                  MOV     #-5,R5
 1051  004124  012737  004132  001110          MOV     #2$,$LPERR
 1052  004132  010110                  2$:     MOV     R1,@R0          ;WRITE THE PATTERN IN THE REGISTER
 1053  004134  011002                          MOV     @R0,R2
```

```
1054  004136  020102                        CMP     R1,R2           ;WAS IT WRITTEN CORRECT_Y?
1055  004140  001407                        BEQ     3$              ;YES, BRANCH
1056  004142  010137  001162                MOV     R1,$REG0        ;GET EXPECTED WORD
1057  004146  010237  001164                MOV     R2,$REG1        ;GET WORD THAT WAS RECVD
1058  004152  104015                        ERROR   15              ;DID NOT READ BACK THE PATTERN THAT
1059                                                                ;WAS WRITTEN INTO THE REGISTER
1060  004154  005205                        INC     R5
1061  004156  001402                        BEQ     TST20           ;;EXIT
1062  004160  005201              3$:       INC     R1              ;INCREMENT COUNT PATTERN
1063  004162  001363                        BNE     2$              ;LUP BAK & WRITE NXT PATTERN IF NOT
1064                                                                ;DONE WITH ALL
1065                                ;
1066                                ;;*****************************************************************
1067                                ;*TEST 20       CHECK THAT RKWC,RKBA,RKDA CAN BE CLEARED BY RESET
1068                                        ;*THIS TEST CHECKS THAT RKWC, RKBA AND RKDA CAN BE CLEARED BY BUS INIT.
1069                                        ;*RESET INSTRUCTION IS USED
1070                                ;;*****************************************************************
1071  004164  000004                TST20:  SCOPE
1072  004166  013700  001252                MOV     RKWC,R0         ;INITIALIZE R0 TO PRINT TO RKWC
1073  004172  010002                        MOV     R0,R2
1074  004174  012701  177775                MOV     #-3,R1          ;SET UP COUNT FOR 3 REGISTERS TO BE CHKD
1075  004200  010103                        MOV     R1,R3
1076  004202  012720  177777        1$:     MOV     #177777,(R0)+   ;SET ALL BITS IN RKWC
1077  004206  005201                        INC     R1              ;                   RKBA
1078  004210  001374                        BNE     1$              ;                   RKDA
1079  004212  000005                        RESET                   ;ISSUE A BUS INIT
1080  004214  005712                2$:     TST     (R2)            ;WAS THE REGISTER (PTD TO BY R2) CLEARED?
1081  004216  001405                        BEQ     3$              ;YES, BRANCH
1082  004220  010237  001162                MOV     R2,$REG0        ;NO, GET ADRES OF REG IS THAT WAS NOT CLEARED
1083  004224  011237  001164                MOV     @R2,$REG1       ;GET CONTENTS OF THAT REGISTER
1084  004230  104017                        ERROR   17              ;RK11 REGISTER (ADRES IN R0) COULD
1085                                                                ;NOT BE CLEARED BY BUS INIT
1086  004232  005722                3$:     TST     (R2)+           ;INCREMENT POINTER TO NXT REGISTER
1087  004234  005203                        INC     R3              ;HAVE U CHKD ALL 3 REGISTERS?
1088  004236  001366                        BNE     2$              ;IF NOT, LOOP BACK & CHK NXT
1089
1090                                ;;*****************************************************************
1091                                ;*TEST 21       CHECKTHAT RKCS, RKWC, RKBA, RKDA CAN BE CLEARED BY CONTROL RESET
1092                                        ;*RKCS IS SET TO 7560, RKWC, RKBA, RKDA ARE ALL SET
1093                                        ;*TO 177777.  CONTROL RESET IS DONE AND IT IS CHECKED
1094                                        ;*IF ALL THESE REGISTERS ARE CLEARED.
1095                                ;;*****************************************************************
1096  004240  000004                TST21:  SCOPE
1097  004242  012746  000340                MOV     #340,-(SP)
1098  004246  012746  004254                MOV     #64$,-(SP)
1099  004252  000002                        RTI
1100  004254                        64$:
1101  004254  013700  001250                MOV     RKCS,R0
1102  004260  012710  007560                MOV     #7560,@R0       ;SET ALL WRITABLE BITS IN RKCS
1103  004264  005210                        INC     @R0             ;SET GO, CONTROL RESET
1104  004266  005005                        CLR     R5
1105  004270  105710                1$:     TSTB    @R0             ;DID CNTRL RDY SET?
1106  004272  100405                        BMI     2$              ;YES, BRANCH
1107  004274  005205                        INC     R5              ;WAITED LONG?
1108  004276  001374                        BNE     1$              ;IF NOT LUP BAK & WAIT
1109  004300  004737  005424                JSR     PC,GT3RG        ;GET RKCS,ER,DS
```

```
1110  004304  104007                    ERROR   7               ;CNTRL RDY DID NOT SET
1111                                                            ;AFTER CNTRL RESET
1112  004306  022710  000200    2$:     CMP     #200,@R0        ;DID CNTRL RESET CLEAR RKCS
1113  004312  001405                    BEQ     3$              ;YES, BRANCH
1114  004314  010037  001162            MOV     R0,$REG0        ;GET ADRES OF RKCS
1115  004320  011037  001164            MOV     @R0,$REG1       ;GET CONTENTS OF RKCS
1116  004324  104014                    ERROR   14              ;CONTROL RESET DID NOT CLEAR RKCS
1117  004326  013702  001252    3$:     MOV     RKWC,R2
1118  004332  010204                    MOV     R2,R4
1119  004334  012701  177777            MOV     #177777,R1
1120  004340  010122                    MOV     R1,(R2)+        ;SET ALL BITS IN RKWC
1121  004342  010122                    MOV     R1,(R2)+        ;                  RKBA
1122  004344  010122                    MOV     R1,(R2)+        ;                  RKDA
1123  004346  104412                    CNT.RESET               ;GO, DO CONTROL RESET
1124                                                            ;THIS IS A CALL FOR THE 'CNTRL-
1125                                                            ;RESET' ROUTINE. A CONTROL RESET IS
1126                                                            ;DONE & AFTER A CERTAIN TIME IF
1127                                                            ;'CNTRL RDY' DOES NOT SET AN ERROR IS
1128                                                            ;REPORTED. NOTE THAT THE PC IN ERROR
1129                                                            ;IS THE PC WHERE CNT.RESET IS
1130                                                            ;LOCATED. THIS IS A VERY BASIC ERROR &
1131                                                            ;IF IT OCCURS GO BACK TO TEST 10.
1132  004350  012703  177775            MOV     #-3,R3
1133  004354  005714            4$:     TST     (R4)            ;WAS THE REGISTER CLEARED?
1134  004356  001405                    BEQ     5$              ;YES, BRANCH
1135  004360  010437  001162            MOV     R4,$REG0        ;GET ADRES OF REGISTER IN ERROR
1136  004364  011437  001164            MOV     @R4,$REG1       ;GET CONTENTS OF THAT REGISTER
1137  004370  104014                    ERROR   14              ;CONTROL RESET DID NOT CLEAR THE
1138                                                            ;REGISTER WHOOSE ADRES IS IN R4
1139  004372  005724            5$:     TST     (R4)+           ;INCREMENT POINTER TO NXT REGISTER
1140  004374  005203                    INC     R3              ;CHKD ALL REGS?
1141  004376  001366                    BNE     4$              ;IF NOT, LUP BAK & CHK THE NXT REG
1142
1143                                    ;;**************************************************************
1144                                    ;*TEST 22       CHECK THAT EACH RK11 REGISTER IS UNIQUELY ADRESSED
1145                                    ;*THIS TEST CHECKS THAT EACH RK11 REGISTER CAN BE UNIQUELY ADDRESSED
1146                                    ;*1) RKCS RKWC, RKBA, RKDA ARE FIRST SET TO 177777 (17576 FOR RKCS)
1147                                    ;*2) REGISTER WHOOSE ADDRESS IS IN R0 IS CLEARED
1148                                    ;*3) EVERY OTHER REGISTER IS CHECKED FOR ERRONEOUS CLEARING BECAUSE OF
1149                                    ;*ADDRESSING ERROR. IF SO THE MULTIPLE ADRESSING ERROR IS REPORTED
1150                                    ;*4) ADDRESS IN R0 IS CHANGED TO THE NEXT REGISTER & THE PROCESS IS
1151                                    ;*   REPEATED.
1152                                    ;;**************************************************************
1153  004400  000004            T$T22:  SCOPE
1154  004402  012746  000340            MOV     #340,-(SP)
1155  004406  012746  004414            MOV     #64$,-(SP)
1156  004412  000002                    RTI
1157  004414                    64$:
1158  004414  012705  177771            MOV     #-7,R5          ;SET UP COUNT FOR THE # OF
1159                                                            ;REGISTERS TO BE UNIQELY ADDRSD
1160  004420  013700  001244            MOV     RKDS,R0         ;INITIALIZE POINTER TO REGIS TO BE SET
1161  004424  012701  177774    1$:     MOV     #-4,R1
1162  004430  013702  001250            MOV     RKCS,R2
1163  004434  012722  017576            MOV     #17576,(R2)+    ;SET BITS IN RKCS
1164  004440  012722  177777    2$:     MOV     #177777,(R2)+   ;SET BITS IN RKWC
1165  004444  005201                    INC     R1              ;             RKBA
```

```
1166  004446  001374                        BNE     2$              ;            RKDA
1167                                                                  ;          RKMR IF RK11C
1168  004450  005010                        CLR     @R0             ;CLEAR REGISTER (WHOSE ADDR IS IN R0)
1169  004452  013702  001250                MOV     RKCS,R2
1170  004456  020002                        CMP     R0,R2           ;WAS THE CLEARED REGISTER RKCS?
1171  004460  001406                        BEQ     3$              ;YES
1172                                                                  ;NO-CHECK IF IT WAS INADVERTENTLY
1173  004462  011203                        MOV     @R2,R3          ;CLEARED BECAUSE OF ADDRESSING
1174  004464  042703  170200                BIC     #170200,R3      ;ERROR
1175  004470  022703  007576                CMP     #7576,R3
1176  004474  001020                        BNE     6$              ;IF SO, REPORT ERROR
1177  004476  005722                3$:     TST     (R2)+           ;INCREMENT POINTER TO NEXT REGISTER
1178  004500  012701  177775                MOV     #-3,R1          ;SET COUNT FOR 3 REGISTERS
1179  004504  020200                4$:     CMP     R2,R0           ;WAS THE CLEARED REGISTER SAME AS
1180                                                                  ;THE REGISTER POINTED TO BY R2
1181  004506  001404                        BEQ     5$              ;YES
1182                                                                  ;NO, CHECK IF THE REGIS UNDER TEST
1183                                                                  ;(POINTED BY R2) WAS INADVERTENTLY
1184                                                                  ;CLEARED WHEN CLEARING THE  REGIS
1185                                                                  ;POINTED TO BY R0, DUE TO
1186                                                                  ;ADDRESSING ERROR
1187  004510  011203                        MOV     @R2,R3          ;GET CONTENTS OF REGIS BEING CHECKED
1188  004512  010304                        MOV     R3,R4           ;FOR INADVERTENT CLEARING
1189  004514  005104                        COM     R4              ;CHECK IF ANY BIT WAS ERRORNEOUSLY CLEARED
1190  004516  001007                        BNE     6$              ;IF SO, REPORT ERROR
1191  004520  005722                5$:     TST     (R2)+           ;INCRMENT PTR TO NEXT REGISTER
1192  004522  005201                        INC     R1              ;INCRMENT COUNT
1193  004524  001367                        BNE     4$              ;CHECK THE REST
1194                                                                  ;
1195  004526  005720                        TST     (R0)+           ;INCREMENT PTR TO THE NXT REGIS TO  BE CLEARED
1196  004530  005205                        INC     R5              ;HAVE ALL THE REGIS BEEN CHECKED?
1197  004532  001334                        BNE     1$              ;IF NOT, LOOP BACK
1198  004534  000415                        BR      TST23           ;;EXIT, IF DONE
1199  004536  010037  001162        6$:     MOV     R0,$REG0        ;GET ADRES OF REGISTER THAT WAS
1200                                                                  ;TRIED TO REFERENCE
1201  004542  010237  001164                MOV     R2,$REG1        ;GET ADRES OF REGISTER THAT GOT
1202                                                                  ;REFERENCED INSTEAD
1203  004546  011037  001166                MOV     @R0,$REG2       ;GET CONTENTS OF REG THAT WAS
1204                                                                  ;ADDRESED & MEANT TO BE CLEARED
1205  004552  010337  001170                MOV     R3,$REG3        ;GET CONTENTS OF REGISTER THAT GOT
1206                                                                  ;CHANGED INSTEAD
1207  004556  104020                        ERROR   20              ;POSSIBLE ADRESING ERROR. TRIED
1208                                                                  ;ADRESING RK11 REGISTER, ANOTHER ONE
1209                                                                  ;GOT ADRESED. REGIS IN R0 WAS THE
1210                                                                  ;ADRESED ONE, REG IN R2
1211                                                                  ;WAS THE ONE THAT GOT ADRESED INSTEAD
1212  004560  023702  001250                CMP     RKCS,R2
1213  004564  001744                        BEQ     3$              ;RETURN TO THE
1214  004566  000754                        BR      5$              ;RIGHT POINT
1215                                ;;**************************************************************
1216                                ;*TEST 23        CHECK THAT HI & LO BYTES OF RKCS CAN BE ADDRESSED
1217                                        ;*THIS TEST CHECKS THAT THE HI & LO BYTES OF RKCS CAN BE
1218                                        ;*ADDRESSED CORRECTLY.
1219                                        ;*BITS IN ALL REGISTERS THAT CAN BE WRITTEN ARE SET. THEN EACH
1220                                        ;*BYTE OF RKCS IS REFERENCED BY 'CLRB' & IT IS CHECKED THAT ONLY
1221                                        ;*THAT BYTE & NO OTHER REGISTER BYTE GETS CLEARED
```

```
                        ;;*****************************************************************
004570  000004         TST23:  SCOPE
004572  012702 177776          MOV     #-2,R2          ;SET COUNT FOR 2BYTES-HI & LO
004576  012700 177775          MOV     #-3,R0          ;SET COUNT
004602  012746 000340          MOV     #340,-(SP)
004606  012746 004614          MOV     #64$,-(SP)
004612  000002                RTI
004614                  64$:
004614  013701 001250          MOV     RKCS,R1         ;INITIALIZE PTR. TO RKCS
004620  012721 007576          MOV     #7576,(R1)+     ;SET ALL BITS IN RKCS
004624  012721 177777  1$:     MOV     #177777,(R1)+   ;SET ALL BITS IN RKWC
004630  005200                INC     R0              ;                 RKBA
004632  001374                BNE     1$              ;                 RKDA
004634  013701 001250          MOV     RKCS,R1         ;INITIALIZE PTR TO RKCS LO BYTE
004640  105011          2$:     CLRB    @R1             ;CLER RKCS LO OR HI BYTE
004642  010104                MOV     R1,R4
004644  017703 174400          MOV     @RKCS,R3        ;GET RKCS WORD
004650  022702 177776          CMP     #-2,R2          ;R U CHKING HI OR LO BYTE?
004654  001011                BNE     3$              ;BRANCH IF HI BYTE
004656  042703 177600          BIC     #177600,R3      ;MASK HI BYTE
004662  001417                BEQ     4$              ;OK IF LO BYTE WAS CLEARED
004664  005037 001162          CLR     $REG0           ;GET EXPCTD RKCS.
004670  010337 001164          MOV     R3,$REG1        ;GET RKCS RECVD, LO BYTE
004674  104022                ERROR   22              ;ALL RK11 REGISTER'S WERE LOADED WITH
                                                        ;1'S THEN TRIED TO ADRES & CLR RKCS
                                                        ;LO BYTE, IT COULD NOT BE CLEARED.
004676  000411                BR      4$
004700  042703 000377  3$:     BIC     #377,R3         ;MASK LO BYTE
004704  001406                BEQ     4$              ;OK IF HI BYTE WAS CLEARED
004706  005037 001162          CLR     $REG0           ;GET EXPCTD RKCS, HI BYTE
                                                        ;GET WHAT WAS ACTUALLY RECVD
004712  000303                SWAB    R3
004714  010337 001164          MOV     R3,$REG1
004720  104023                ERROR   23              ;ALL RK11 REGISTER'S WERE LOADED WITH
                                                        ;1'S THEN TRIED TO ADRES & CLR RKCS
                                                        ;HI BYTE IT COULD NOT BE CLEARED.
004722  012700 177774  4$:     MOV     #-4,R0          ;INITIALIZE COUNT FOR REST OF REGISTERS
004726  013705 001250          MOV     RKCS,R5         ;INITIALIZE POINTER TO RKCS
004732  005725          5$:     TST     (R5)+           ;INCREMENT PTR TO NXT REGIS
004734  005200                INC     R0              ;ALL REGS DONE?
004736  001417                BEQ     6$              ;IF YES, GO & ADRES TO CLEAR RKCS HI BYTE
004740  011503                MOV     @R5,R3          ;IF NOT, GET CONTENTS OF THE REGIS
                                                        ;BEING CHKD
004742  005103                COM     R3              ;COMPLEMENT THE CONTENTS, SHOULD BE
                                                        ;0 FOR IT WAS
004744  001772                BEQ     5$              ;PREVIOUSLY SET TO ALL 1'S IF IT'S
                                                        ;0 BRANCH, IF NOT REPORT ERROR
004746  010137 001162          MOV     R1,$REG0        ;GET RKCS-BYTE-ADRES WHICH WAS TRIED
                                                        ;TO ADRES
004752  010537 001164          MOV     R5,$REG1        ;GET ADRES OF REGIS WHICH GOT ADRESED
                                                        ;INSTEAD
004756  012737 177777 001166    MOV     #177777,$REG2   ;GET EXPCTD CONTENTS OF REGISTER
                                                        ;THAT GO ADRESED
004764  005103                COM     R3              ;GET CONTENTS RECVD FROM THAT REGIS
004766  010337 001170          MOV     R3,$REG3
004772  104024                ERROR   24              ;ALL RK11 REGISTERS WERE LOADED
```

```
                                                ;WITH 1'S.  RKCS
                                                ;BYTE (ADRES UNDER 'BYTE' IN ER
                                                ;MSGE) WAS ADRESED
                                                ;USING 'CLRB', BUT REGISTER (ADRES
                                                ;UNDER 'REGIS' IN
                                                ;ER MSG) GOT CHANGED AS A RESULT.
004774  000756          BR      5$
004776  005201  6$:     INC     R1              ;POSITION PTR TO RKCS HI BYTE
005000  005202          INC     R2              ;CHK IF BOTH HI & LO BYTES (RKCS)
                                                ;WERE CLEARED
005002  001316          BNE     2$              ;IF NOT BRANCH BACK
                ;
                ;;*****************************************************************
                ;*TEST 24        CHECK THAT HI & LO BYTES OF RKWC,BA,DA CAN BE ADDRESSED
                        ;*THIS TEST CHECKS THAT BYTE OPERATIONS ON RKWC, RKBA & RKDA CAN BE DONE
                        ;*CORRECTLY. FIRST RKWC, RKCS, RKBA, RKDA ARE SET TO 177777.
                        ;*1) REGISTER BYTE POINTED TO BY R2 IS CLEARED USING 'CLRB'
                        ;*2) IT IS CHECKED THAT ONLY THAT BYTE AND NO OTHER BYTES GET CLEARED
                        ;*3) POINTER R2 IS INCREMENTED TO THE NEXT REGISTER-BYTE & THE PROCESS
                        ;*IS REPEATED. LO BYTE IS DONE FIRST, THEN HI-BYTE IS DONE.
                ;;*****************************************************************
004  000004  TST24:  SCOPE
006  012746  000340          MOV     #340,-(SP)
012  012746  005020          MOV     #64$,-(SP)
016  000002                  RTI
020             64$:
020  012704  177772          MOV     #-6,R4          ;SET UP COUNT FOR 6 REG-BYTES TO BE ADRESED
024  013702  001252          MOV     RKWC,R2         ;INITIALIZE POINTER TO RKWC
030  012700  177775  1$:     MOV     #-3,R0
034  013701  001250          MOV     RKCS,R1
040  012721  007576          MOV     #7576,(R1)+     ;SET RKCS BITS
044  012721  177777  2$:     MOV     #177777,(R1)+   ;    RKWC
050  005200                  INC     R0              ;       RKBA
052  001374                  BNE     2$              ;       RKDA

054  105012                  CLRB    @R2             ;ADDRESS & CLEAR REGIS BYTE UNDER TEST

056  010200                  MOV     R2,R0
060  042700  000001          BIC     #1,R0           ;CONVERT THE BYTE ADRES INTO WORD
                                                ;ADRES THAT IT BELONGS TO
064  011001                  MOV     @R0,R1          ;GET THE ENTIRE REGIS WRD
066  032702  000001          BIT     #1,R2           ;WAS THE CLRD BYTE HI OR LO?
072  001003                  BNE     3$              ;WAS HI, BRANCH
074  042701  177400          BIC     #177400,R1      ;WAS LO-MASK HI BYTE
100  000402                  BR      4$
102  042701  000377  3$:     BIC     #377,R1         ;MASK LO BYTE
106  001411  4$:     BEQ     5$              ;WAS THE ADDRESSED BYTE CLEARED? BR IF YES
110  010237  001162          MOV     R2,$REG0        ;GET ADRES OF REG-BYTE THAT WAS
                                                ;TRIED TO B ADRESED & CLEARED
114  032702  000001          BIT     #1,R2
120  001401                  BEQ     11$
122  000301                  SWAB    R1
124  010137  001164  11$:    MOV     R1,$REG1        ;GET CONTENTS OF REG-BYTE
130  104025                  ERROR   25              ;TRIED TO ADRES & CLR A REGISTER BYTE
                                                ;(ADRES UNDER 'REG-BYTE' IN ER MSGE), COULD
                                                ;NOT CLEAR IT.
```

```
1334  005132  017701  174112          5$:     MOV     @RKCS,R1        ;
1335  005136  022701  007776                  CMP     #7776,R1        ;WAS RKCS ERRONEOUSLY CLRD?
1336  005142  001410                          BEQ     6$              ;NO, BRANCH
1337  005144  010237  001162                  MOV     R2,$PEGO        ;GET ADRES OF REG-BYTE THAT WAS
1338                                                                  ;TRIED TO B ADRESED & CLEARED.
1339  005150  012737  007776  001164          MOV     #7776,$REG1     ;GET EXPCTD RKCS
1340  005156  010137  001166                  MOV     R1,$REG2        ;GET RKCS RECVD
1341  005162  104016                          ERROR   16              ;ALL RK11 REGISTRS WERE LOADED WITH 1'S
1342                                                                  ;TRIED TO ADRES & CLR 'REGIS-BYTE' (IN ER MSGE)
1343                                                                  ;RKCS GOT CHANGED AS A RESULT.  '(RKCS)EXP'
1344                                                                  ;CONTAINS EXPCTD RKCS.  'RKCS (RECVD)' CONTAINS
1345                                                                  ;RKCS RECVD.
1346  005164  012700  177772          6$:     MOV     #-6,R0          ;SET COUNT FOR BYTES
1347  005170  013701  001252                  MOV     RKWC,R1         ;INITIALIZE PTR TO RKWC
1348  005174  020102                  7$:     CMP     R1,R2           ;WAS THE CLEARED BYTE (PTD TO BY R2)
1349                                                                  ;SAME AS BYTE TO BE CHKD (PTD TO BY R1)?
1350  005176  001433                          BEQ     10$             ;IF YES, DO NOT CHK THIS BYTE
1351  005200  010105                          MOV     R1,R5
1352  005202  042705  000001                  BIC     #1,R5           ;STRIP WORD ADDRESS FROM BYTE ADDRS
1353  005206  011503                          MOV     @R5,R3
1354  005210  032701  000001                  BIT     #1,R1           ;IS THE BYTE TO B CHKD LO OR HI BYTE?
1355  005214  001003                          BNE     8$              ;HI BYTE-BRANCH
1356  005216  042703  177400                  BIC     #177400,R3      ;MASK HI BYTE
1357  005222  000402                          BR      9$
1358  005224  042703  000377          8$:     BIC     #377,R3         ;MASK LO BYTE
1359  005230  001016          9$:     BNE     10$
1360  005232  010237  001162                  MOV     R2,$REGO        ;GET ADRES OF REG-BYTE THAT WAS
1361                                                                  ;TRIED TO B ADRESED & CLEARED
1362  005236  010137  001164                  MOV     R1,$REG1        ;GET ADRS OF REG-BYTE THAT GOT
1363                                                                  ;ADRESED INSTEAD
1364  005242  012737  000377  001166          MOV     #377,$REG2      ;GET EXPCTD CONTENTS OF REG-BYTE
1365                                                                  ;THAT GOT ADRESED
1366  005250  032701  000001                  BIT     #1,R1
1367  005254  001401                          BEQ     12$
1368  005256  000303                          SWAB    R3
1369  005260  010337  001170          12$:    MOV     R3,$REG3        ;GET CONTENTS RECVD FOR THAT REG-BYTE
1370  005264  104021                          ERROR   21              ;ALL RK11 REGISTRS WERE LOADED WITH 1'S.
1371                                                                  ;TRIED TO ADRES & CLR 'REG-BYT1' (IN ER MSGE)
1372                                                                  ;'REG-BYT2' GOT CHANGD AS A RESULT.
1373                                                                  ;'BYT2-EXPCT' (IN ER MSGE) IS THE EXPCTD CONTENTS
1374                                                                  ;OF REG-BYT2.  'BYT2-RECVD' IS THE CONTENTS RECVD.
1375  005266  005201          10$:    INC     R1              ;INCREMENT PTR TO NXT REG BYTE
1376  005270  005200                  INC     R0              ;ALL BYTES CHKD?
1377  005272  001340                  BNE     7$              ;NO, LOOP BACK
1378  005274  005202                  INC     R2              ;INCREMENT PTR TO NXT REG BYTE TO B CLRD
1379  005276  005204                  INC     R4              ;ALL BYTES CLRD?
1380  005300  001253                  BNE     1$              ;LOOP BACK
1381
1382
1383
1384
1385                          .SBTTL  END OF PASS ROUTINE
1386
1387                          ;;*****************************************************************
1388                          ;*INCREMENT THE PASS NUMBER ($PASS)
1389                          ;*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 START1
                                        $EOP:
                                                SCOPE
005302 000004                                   CLR     $TSTNM          ;;ZERO THE TEST NUMBER
005304 005037 001102                            CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
005310 005237 001100                            INC     $PASS           ;;INCREMENT THE PASS NUMBER
005314 042737 100000 001100                     BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
005320 005327                                   DEC     (PC)+           ;;LOOP?
005326 000001                           $EOPCT: .WORD   1
005330 003022                                   BGT     $DOAGN          ;;YES
005332 012737                                   MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
005334 000001                           $ENDCT: .WORD   1
005336 005330                                   $EOPCT
005340 104401 005407                            TYPE    ,$ENDMG         ;;TYPE "END PASS #"
005344 013746 001100                            MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
005350 104405                                   TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
005352 104401 005404                            TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
005356 013700 000042                    $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
005362 001405                                   BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
005364 000005                                   RESET                   ;;CLEAR THE WORLD
005366 004710                           $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
005370 000240                                   NOP                     ;;SAVE ROOM
005372 000240                                   NOP                     ;;FOR
005374 000240                                   NOP                     ;;ACT11
005376                                  $DOAGN:
005376 000137                                   JMP     @(PC)+          ;;RETURN
005400 002100                           $RTNAD: .WORD   START1
005402     377      377      000       $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
005405     015 042412 042116           $ENDMG: .ASCIZ  <15><12>/END PASS #/
005413 050040 051501 020123
005421 000043
```

```
                                                .SBTTL  GT3RG:  ROUTINE FOR GETTING RKCS, RKER, RKDS

                                        ;GT3RG
                                        ;SUBROUTINE FOR TRANSFERRING THE CONTENTS OF RKCS, RKER, RKDS
                                        ;TO  $REG0, $REG1, $REG2 RESPECTIVELY BEFORE TYPING OUT AN
                                        ;ERROR MESSAGE.
                                        ;CALL:  JSR     PC,GT3RG

005423 017737 173620 001162     GT3RG:  MOV     @RKCS,$REG0     ;GET RKCS
005431 017737 173610 001164             MOV     @RKER,$REG1     ;GET RKER
005437 017737 173600 001166             MOV     @RKDS,$REG2     ;GET RKDS
005445 000207                           RTS     PC              ;EXIT FROM THIS SUBROUTINE
```

```
                                        .SBTTL  GT4RG:  ROUTINE FOR GETTING RKCS, RKER, RKDS, RKDA

                                ;GT4RG
                                ;SUBROUTINE FOR TRANSFERRING CONTENTS OF RKCS, RKER, RKDS
                                ;RKDA TO $REG0, $REG1, $REG2, $REG3 RESPECTIVELY BEFORE
                                ;TYPING OUT AN ERROR MESSAGE.
                                ;CALL:  JSR     PC,GT4RG
005450  004737  005424          GT4RG:  JSR     PC,GT3RG
005454  017737  173576  001170          MOV     @RKDA,$REG3
005462  000207                          RTS     PC


                                        .SBTTL  DELAY:  TIME DELAY ROUTINE

                                ;DELAY
                                ;THIS ROUTINE PROVIDES  A VARIABLE TIME DELAY. THE CALL FOR THIS
                                ;ROUTINE IS AN ENCODED 'TRAP' INSTRUCTION.
                                ;CALL:          DELAY   ,N      N IS ANY OCTAL NO. FROM 1 TO 177777
                                ;THE DELAY PROVIDED IS 7.5N US (CONVERT N TO DECIMAL) FOR 11/20
                                ;1.5N US FOR 11/45
                                ;IF THE USER WANTS TO CHANGE THE DELAY TIME (EXMP: SHORTER DELAY TO
                                ;GET A TIGHTER SCOPE LOOP) THE VARIABLE 'N' FOLLOWING 'DELAY' SHOULD
                                ;BE CHANGED TO SUIT THE INDIVIDUAL NEED.

005464  017637  000000  001264  DELA.Y: MOV     @(SP),TIMER     ;GET 'AMOUNT' (N) FOR WHICH
005472  062716  000002                  ADD     #2,(SP)         ;DELAY IS TO BE PROVIDED
                                                                ;ADJUST STACK POINTER TO SKIP OVER 'N'
005476  005337  001264          1$:     DEC     TIMER           ;COUNT DOWN TO 0
005502  001375                          BNE     1$
005504  000002                          RTI                     ;RETURN TO MAIN PROGRAM


                                        .SBTTL  CON.RESET:      CONTROL REST ROUTINE

                                ;CON.RESET
                                ;THIS ROUTINE ISSUES A CONTROL RESET AND WAITS FOR
                                ;THE 'CNTRL RDY' FLAG TO SET.  WHEN THE FLAG SETS
                                ;AN EXIT IS MADE OUT OF THE ROUTINE.  IF 'CNTRL-RDY'
                                ;DOES NOT SET WITHIN A CERTAIN TIME AN ERROR MESSAGE
                                ;       CNT RDY DIDN'T SET
                                ;       PC=XXXXXX RKCS=YYYYYY
                                ;IS GIVEN. NOTE THAT XXXXXX IS THE PC WHERE 'CNT.RESET' OR 'CNT.RDY'
                                ;IS CALLED.

                                ;CALL:  CNT.RESET


                                        .SBTTL  CNT.RDY:        WAIT FOR CONTROL READY ROUTINE

                                ;CN.RDY
                                ;THIS ROUTINE WAITS FOR THE CONTROL READY BIT TO SET AND WHEN IT
```

```
1502                                    ;SETS EXITS OUT. IF WITHIN A CERTAIN TIME CNTRL RDY DOES
1503                                    ;;CT SET AN ERROR IS REPORTED. WAITING TIME IS 883 MS FOR 11/20
1504                                    ;175 MS FOR 11/45 WITH BIPOLAR MEMORY.
1505                                    ;CALL:  CNT.RDY
1506  005506  012777  000001  173534    CN.RS*: MOV     #1,@RKCS        ;ISSUE A CONTROL RESET
1507  005514  012737  177500  001170            MOV     #-300,$REG3     ;SET UP COUNT
1508  005522  000402                            BR      CN.RDY+4                ;SKIP OVER CN.RDY
1509  005524  005037  001170            CN.RDY: CLR     $REG3
1510  005530  105777  173514            1$:     TSTB    @RKCS           ;DID CNTRL-RDY SET?
1511  005534  100435                            BMI     3$              ;YES, EXIT
1512  005536  005237  001170                    INC     $REG3           ;WAITED LONG?
1513  005542  001372                            BNE     1$              ;IF NOT, GO BAK & WAIT
1514  005544  032777  020000  173366  2$:      BIT     #SW13,@SWR      ;INHIBIT TYPEOUT?
1515  005552  001026                            BNE     3$              ;IF YES, SKIP TYPEOUT
1516  005554  104401                            TYPE
1517  005556  001216                            MSG3
1518  005560  104401  005566                    TYPE    ,65$            ;;TYPE ASCIZ STRING
1519  005564  000403                            BR      64$             ;;GET OVER THE ASCIZ
1520                                  ;;65$:    .ASCIZ  <15><12>/PC=/
1521  005574                          64$:
1522  005574  011646                            MOV     (SP),-(SP)
1523  005576  162716  000002                    SUB     #2,(SP)
1524  005602  104402                            TYPOC                   ;GO TYPE PC IN THE MAIN PROGRAM,
1525                                                                    ; WHERE ERROR OCCURRED
1526  005604  104401  005612                    TYPE    ,67$            ;;TYPE ASCIZ STRING
1527  005610  000404                            BR      66$             ;;GET OVER THE ASCIZ
1528                                  ;;67$:    .ASCIZ  / RKCS=/
1529  005622                          66$:
1530  005622  017746  173422                    MOV     @RKCS,-(SP)     ;GET RKCS
1531  005626  104402                            TYPOC                   ;GO TYPE IT
1532
1533  005630  000002                  3$:       RTI                     ;RETURN FROM THIS
1534                                                                    ;ROUTINE TO THE MAIN
1535                                                                    ;PROGRAM
1536
1537
1538                                  ;THIS PART OF THE PROGRAM CONTAINS THE COMMON ROUTINES CALLED
1539                                  ;FROM THE SYSMAC.SML PACKAGE
1540                                  ;
1541
1542                                  .SBTTL  SCOPE HANDLER ROUTINE
1543
1544                                  ;;*****************************************************************
1545                                  ;*THIS ROUTINE CONTROLS THE LOOPING OF SUBTESTS. IT WILL INCREMENT
1546                                  ;*AND LOAD THE TEST NUMBER($TSTNM) INTO THE DISPLAY REG.(DISPLAY<7:0>)
1547                                  ;*AND LOAD THE ERROR FLAG ($ERFLG) INTO DISPLAY<15:08>
1548                                  ;*THE SWITCH OPTIONS PROVIDED BY THIS ROUTINE ARE:
1549                                  ;*SW14=1         LOOP ON TEST
1550                                  ;*SW11=1         INHIBIT ITERATIONS
1551                                  ;*SW09=1         LOOP ON ERROR
1552                                  ;*SW08=1         LOOP ON TEST IN SWR<7:0>
1553                                  ;*CALL
1554                                  ;*      SCOPE           ;;SCOPE=IOT
1555
1556  005632                          $SCOPE:
1557  005632  104407                            CKSWR                   ;;TEST  FOR CHANGE IN SOFT-SWR
```

```
1558  005634  032777  040000  173276  1$:     BIT     #BIT14,@SWR     ;;LOOP ON PRESENT TEST?
1559  005642  001111                          BNE     $OVER           ;;YES IF SW14=1
1560                                  ;*****START OF CODE FOR THE XOR TESTER*****
1561  005644  000416                  $XTSTR: BR      6$              ;;IF RUNNING ON THE "XOR" TESTER CHANGE
1562                                                                  ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
1563  005646  013746  000004                  MOV     @#ERRVEC,-(SP)  ;;SAVE THE CONTENTS OF THE ERROR VECTOR
1564  005652  012737  005672  000004          MOV     #5$,@#ERRVEC    ;;SET FOR TIMEOUT
1565  005660  005737  177060                  TST     @#177060        ;;TIME OUT ON XOR?
1566  005664  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
1567  005670  000463                          BR      $SVLAD          ;;GO TO THE NEXT TEST
1568  005672  022626                  5$:     CMP     (SP)+,(SP)+     ;;CLEAR THE STACK AFTER A TIME OUT
1569  005674  012637  000004                  MOV     (SP)+,@#ERRVEC  ;;RESTORE THE ERROR VECTOR
1570  005700  000423                          BR      7$              ;;LOOP ON THE PRESENT TEST
1571  005702                          6$:;*****END OF CODE FOR THE XOR TESTER*****
1572  005702  032777  000400  173230          BIT     #BIT08,@SWR     ;;LOOP ON SPEC. TEST?
1573  005710  001404                          BEQ     2$              ;;BR IF NO
1574  005712  127737  173222  001102          CMPB    @SWR,$TSTNM     ;;ON THE RIGHT TEST?    SWR<7:0>
1575  005720  001462                          BEQ     $OVER           ;;BR IF YES
1576  005722  105737  001103          2$:     TSTB    $ERFLG          ;;HAS AN ERROR OCCURRED?
1577  005726  001421                          BEQ     3$              ;;BR IF NO
1578  005730  123737  001115  001103          CMPB    $ERMAX,$ERFLG   ;;MAX. ERRORS FOR THIS TEST OCCURRED?
1579  005736  101015                          BHI     3$              ;;BR IF NO
1580  005740  032777  001000  173172          BIT     #BIT09,@SWR     ;;LOOP ON ERROR?
1581  005746  001404                          BEQ     4$              ;;BR IF NO
1582  005750  013737  001110  001106  7$:     MOV     $LPERR,$LPADR   ;;SET LOOP ADDRESS TO LAST SCOPE
1583  005756  000443                          BR      $OVER
1584  005760  105037  001103          4$:     CLRB    $ERFLG          ;;ZERO THE ERROR FLAG
1585  005764  005037  001206                  CLR     $TIMES          ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
1586  005770  000415                          BR      1$              ;;ESCAPE TO THE NEXT TEST
1587  005772  032777  004000  173140  3$:     BIT     #BIT11,@SWR     ;;INHIBIT ITERATIONS?
1588  006000  001011                          BNE     1$              ;;BR IF YES
1589  006002  005737  001100                  TST     $PASS           ;;IF FIRST PASS OF PROGRAM
1590  006006  001406                          BEQ     1$              ;;      INHIBIT ITERATIONS
1591  006010  005237  001104                  INC     $ICNT           ;;INCREMENT ITERATION COUNT
1592  006014  023737  001206  001104          CMP     $TIMES,$ICNT    ;;CHECK THE NUMBER OF ITERATIONS MADE
1593  006022  002021                          BGE     $OVER           ;;BR IF MORE ITERATION REQUIRED
1594  006024  012737  000001  001104  1$:     MOV     #1,$ICNT        ;;REINITIALIZE THE ITERATION COUNTER
1595  006032  013737  006102  001206          MOV     $MXCNT,$TIMES   ;;SET NUMBER OF ITERATIONS TO DO
1596  006040  105237  001102          $SVLAD: INCB    $TSTNM          ;;COUNT TEST NUMBERS
1597  006044  011637  001106                  MOV     (SP),$LPADR     ;;SAVE SCOPE LOOP ADDRESS
1598  006050  011637  001110                  MOV     (SP),$LPERR     ;;SAVE ERROR LOOP ADDRESS
1599  006054  005037  001210                  CLR     $ESCAPE         ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
1600  006060  112737  000001  001115          MOVB    #1,$ERMAX       ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
1601  006066  013777  001102  173046  $OVER:  MOV     $TSTNM,@DISPLAY ;;DISPLAY TEST NUMBER
1602  006074  013716  001106                  MOV     $LPADR,(SP)     ;;FUDGE RETURN ADDRESS
1603  006100  000002                          RTI                     ;;FIXES PS
1604  006102  000050                  $MXCNT: 50                      ;;MAX. NUMBER OF ITERATIONS
1605
1606
1607                                  ;;**********************************************************************
1608
1609                                  .SBTTL  ERROR HANDLER ROUTINE
1610
1611                                  ;*SW15=1        HALT ON ERROR
1612                                  ;*SW13=1        INHIBIT ERROR TYPEOUTS
1613                                  ;*SW10=1        BELL ON ERROR
```

```
                                        ;*SW09=1        LOOP ON ERROR
                                        ;*SW12=1        CYCLE ON ERROR TO PREVIOUS 'SCOPE'
                                        ;*GO TO $ERRTYP ON ERROR

006104  104407                  $ERROR: CKSWR                           ;CHECK FOR SOFTWARE SWITCH ENTRY REQUEST
006106  105237  001103          7$:     INCB    $ERFLG                  ;SET THE ERROR FLAG
006112  001775                          BEQ     7$                      ;DON'T LET THE FLAG GO TO ZERO
006114  013777  001102  173020          MOV     $TSTNM,@DISPLAY         ;DISPLAY TEST NUMBER AND ERROR FLAG
006122  005237  001112          1$:     INC     $ERTTL                  ;COUNT THE NUMBER OF ERRORS
006126  011637  001116                  MOV     (SP),$ERRPC             ;GET ADDRESS OF ERROR INSTRUCTION
006132  162737  000002  001116          SUB     #2,$ERRPC
006140  117737  172752  001114          MOVB    @$ERRPC,$ITEMB          ;STRIP AND SAVE THE ERROR ITEM CODE
006146  032777  020000  172764          BIT     #SW13,@SWR              ;SKIP TYPEOUT IF SET
006154  001004                          BNE     2$                      ;SKIP TYPEOUTS
006156  004737  006244                  JSR     PC,@#$ERRTYP            ;GO TO USER ERROR ROUTINE
006162  104401  001213                  TYPE    ,$CRLF
006166  005777  172746          2$:     TST     @SWR                    ;HALT ON ERROR
006172  100002                          BPL     3$                      ;SKIP IF CONTINUE
006174  000000                          HALT                            ;HALT ON ERROR!
006176  104407                          CKSWR                           ;CHECK FOR SOFTWARE SWITCH ENTRY REQUEST
006200  032777  010000  172732  3$:     BIT     #SW12,@SWR              ;SW 12 SET?
006206  001402                          BEQ     .+6                     ;NO, BRANCH
006210  013716  001106                  MOV     $LPADR,(SP)             ;ADJUST RETURN ADRES FOR SW12
006214  032777  001000  172716          BIT     #SW09,@SWR              ;LOOP ON ERROR SWITCH SET?
006222  001402                          BEQ     4$                      ;BR IF NO
006224  013716  001110                  MOV     $LPERR,(SP)             ;FUDGE RETURN FOR LOOPING
006230  005737  001210          4$:     TST     $ESCAPE                 ;CHECK FOR AN ESCAPE ADDRESS
006234  001402                          BEQ     5$                      ;BR IF NONE
006236  013716  001210                  MOV     $ESCAPE,(SP)            ;FUDGE RETURN ADDRESS FOR ESCAPE
006242  000002                  5$:     RTI                             ;RETURN

                                        .SBTTL  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.

006244                          $ERRTYP:
006244  104401  001213                  TYPE    ,$CRLF                  ;;"CARRIAGE RETURN" & "LINE FEED"
006250  010046                          MOV     R0,-(SP)                ;;SAVE R0
006252  005000                          CLR     R0                      ;;PICKUP THE ITEM INDEX
006254  153700  001114                  BISB    @#$ITEMB,R0
006260  001004                          BNE     1$                      ;;IF ITEM NUMBER IS ZERO, JUST
                                                                        ;;TYPE THE PC OF THE ERROR
006262  013746  001116                  MOV     $ERRPC,-(SP)            ;;SAVE $ERRPC FOR TYPEOUT
                                                                        ;;ERROR ADDRESS
006266  104402                          TYPOC                           ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
006270  000426                          BR      6$                      ;;GET OUT
006272  005300                  1$:     DEC     R0                      ;;ADJUST THE INDEX SO THAT IT WILL
006274  006300                          ASL     R0                      ;;      WORK FOR THE ERROR TABLE
006276  006300                          ASL     R0
006300  006300                          ASL     R0
006302  062700  001272                  ADD     #$ERRTB,R0              ;;FORM TABLE POINTER
006306  012037  006316                  MOV     (R0)+,2$                ;;PICKUP "ERROR MESSAGE" POINTER
006312  001404                          BEQ     3$                      ;;SKIP TYPEOUT IF NO POINTER
```

```
1670  006314  104401                        TYPE                    ;;TYPE THE "ERROR MESSAGE"
1671  006316  000000                2$:     .WORD   0               ;;"ERROR MESSAGE" POINTER GOES HERE
1672  006320  104401  001213                TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
1673  006324  012037  006334        3$:     MOV     (R..+,4$        ;;PICKUP "DATA HEADER" POINTER
1674  006330  001404                        BEQ     5$              ;;SKIP TYPEOUT IF 0
1675  006332  104401                        TYPE                    ;;TYPE THE "DATA HEADER"
1676  006334  000000                4$:     .WORD   0               ;;"DATA HEADER" POINTER GOES HERE
1677  006336  104401  001213                TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
1678  006342  011000                5$:     MOV     (R0),R0         ;;PICKUP "DATA TABLE" POINTER
1679  006344  001004                        BNE     7$              ;;GO TYPE THE DATA
1680  006346  012600                6$:     MOV     (SP)+,R0        ;;RESTORE R0
1681  006350  104401  001213                TYPE    ,$CRLF          ;;"CARRIAGE RETURN" & "LINE FEED"
1682  006354  000207                        RTS     PC              ;;RETURN
1683  006356                        7$:
1684  006356  013046                        MOV     @(R0)+,-(SP)    ;;SAVE @(R0)+ FOR TYPEOUT
1685  006360  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1686  006362  005710                        TST     (R0)            ;;IS THERE ANOTHER NUMBER?
1687  006364  001770                        BEQ     6$              ;;BR IF NO
1688  006366  104401  006374                TYPE    ,8$             ;;TYPE TWO(2) SPACES
1689  006372  000771                        BR      7$              ;;LOOP
1690  006374  020040      000       8$:     .ASCIZ  /  /            ;;TWO(2) SPACES
1691            006400                      .EVEN
1692
1693                                  .SBTTL  TTY INPUT ROUTINE
1694
1695                                  ;;**************************************************************
1696                                  .ENABL  LSB
1697
1698                                  ;;**************************************************************
1699                                  ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
1700                                  ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
1701                                  ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
1702                                  ;*WHEN OPERATING IN TTY FLAG MODE.
1703  006400  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR      ;;IS THE SOFT-SWR SELECTED?
1704  006406  001074                        BNE     15$             ;;BRANCH IF NO
1705  006410  105777  172530                TSTB    @$TKS           ;;CHAR THERE?
1706  006414  100071                        BPL     15$             ;;IF NO, DON'T WAIT AROUND
1707  006416  117746  172524                MOVB    @$TKB,-(SP)     ;;SAVE THE CHAR
1708  006422  042716  177600                BIC     #^C177,(SP)     ;;STRIP-OFF THE ASCII
1709  006426  022726  000007                CMP     #7,(SP)+        ;;IS IT A CONTROL G?
1710  006432  001062                        BNE     15$             ;;NO, RETURN TO USER
1711  006434  123727  001134  000001        CMPB    $AUTOB,#1       ;;ARE WE RUNNING IN AUTO-MODE?
1712  006442  001456                        BEQ     15$             ;;BRANCH IF YES
1713
1714  006444  104401  007125                TYPE    ,$CNTLG         ;;ECHO THE CONTROL-G (^G)
1715  006450  104401  007132        $GTSWR: TYPE    ,$MSWR          ;;TYPE CURRENT CONTENTS
1716  006454  013746  000176                MOV     $WREG,-(SP)     ;;SAVE SWREG FOR TYPEOUT
1717  006460  104402                        TYPOC                   ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
1718  006462  104401  007143                TYPE    ,$MNEW          ;;PROMPT FOR NEW SWR
1719  006466  005046                19$:     CLR     -(SP)           ;;CLEAR COUNTER
1720  006470  005046                        CLR     -(SP)           ;;THE NEW SWR
1721  006472  105777  172446        7$:     TSTB    @$TKS           ;;CHAR THERE?
1722  006476  100375                        BPL     7$              ;;IF NOT TRY AGAIN
1723
1724  006500  117746  172442                MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
1725  006504  042716  177600                BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
```

```
1726
1727
1728
1729  006510  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
1730  006514  001005                          BNE     10$             ;;BRANCH IF NOT
1731  006516  104401  007120                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (↑U)
1732  006522  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
1733  006526  000757                          BR      19$             ;;LET'S TRY IT AGAIN
1734
1735
1736  006530  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
1737  006534  001022                          BNE     16$             ;;BRANCH IF NO
1738  006536  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
1739  006542  001403                          BEQ     11$             ;;BRANCH IF YES
1740  006544  016677  000002  172366          MOV     2(SP),@SWR      ;;SAVE NEW SWR
1741  006552  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
1742  006556  104401  001213          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
1743  006562  123727  001135  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
1744  006570  001003                          BNE     15$             ;;BRANCH IF NOT
1745  006572  012777  000100  172344          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
1746  006600  000002          15$:    RTI                             ;;RETURN
1747  006602  004737  007324          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
1748  006606  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
1749  006612  002420                          BLT     18$             ;;BRANCH IF YES
1750  006614  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
1751  006620  003015                          BGT     18$             ;;BRANCH IF YES
1752  006622  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
1753  006626  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
1754  006632  001403                          BEQ     17$             ;;BRANCH IF YES
1755  006634  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
1756  006636  006316                          ASL     (SP)            ;;   CHAR OVER TO MAKE
1757  006640  006316                          ASL     (SP)            ;;   ROOM FOR NEW ONE.
1758  006642  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
1759  006646  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
1760  006652  000707                          BR      7$              ;;GET THE NEXT ONE
1761  006654  104401  001212          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
1762  006660  000720                          BR      20$             ;;SIMULATE CONTROL-U
1763                                  .DSABL  LSB
1764
1765
1766                                  ;;**************************************************************
1767                                  ;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
1768                                  ;*CALL:
1769                                  ;*      RDCHR                   ;;INPUT A SINGLE CHARACTER FROM THE TTY
1770                                  ;*      RETURN HERE             ;;CHARACTER IS ON THE STACK
1771                                  ;*                              ;;WITH PARITY BIT STRIPPED OFF
1772                                  ;
1773
1774  006662  011646          $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
1775  006664  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
1776  006672  105777  172246          1$:     TSTB    @$TKS           ;;WAIT FOR
1777  006676  100375                          BPL     1$              ;;A CHARACTER
1778  006700  117766  172242  000004          MOVB    @$TKB,4(SP)     ;;READ THE TTY
1779  006706  042766  177600  000004          BIC     #↑C<177>,4(SP)  ;;GET RID OF JUNK IF ANY
1780  006714  026627  000004  000023          CMP     4(SP),#23       ;;IS IT A CONTROL-S?
1781  006722  001013                          BNE     3$              ;;BRANCH IF NO
```

```
1782  006724  105777  172214          2$:     TSTB    @$TKS           ;;WAIT FOR A CHARACTER
1783  006730  100375                          BPL     2$              ;;LOOP UNTIL ITS THERE
1784  006732  117746  172210                  MOVB    @$TKB,-(SP)     ;;GET CHARACTER
1785  006736  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII
1786  006742  022627  000021                  CMP     (SP)+,#21       ;;IS IT A CONTROL-Q?
1787  006746  001366                          BNE     2$              ;;IF NOT DISCARD IT
1788  006750  000750                          BR      1$              ;;YES, RESUME
1789  006752  026627  000004  000140  3$:     CMP     4(SP),#140      ;;IS IT UPPER CASE?
1790  006760  002407                          BLT     4$              ;;BRANCH IF YES
1791  006762  026627  000004  000175          CMP     4(SP),#175      ;;IS IT A SPECIAL CHAR?
1792  006770  003003                          BGT     4$              ;;BRANCH IF YES
1793  006772  042766  000040  000004          BIC     #40,4(SP)       ;;MAKE IT UPPER CASE
1794  007000  000002                  4$:     RTI                     ;;GO BACK TO USER
1795                                          ;;**************************************************************
1796                                          ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
1797                                          ;*CALL:
1798                                          ;*      RDLIN                   ;;INPUT A STRING FROM THE TTY
1799                                          ;*      RETURN HERE             ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
1800                                          ;*                              ;;TERMINATOR WILL BE A BYTE OF ALL 0'S
1801
1802  007002  010346                  $RDLIN: MOV     R3,-(SP)        ;;SAVE R3
1803  007004  012703  007110          1$:     MOV     #$TTYIN,R3      ;;GET ADDRESS
1804  007010  022703  007120          2$:     CMP     #$TTYIN+8.,R3   ;;BUFFER FULL?
1805  007014  101405                          BLOS    4$              ;;BR IF YES
1806  007016  104410                          RDCHR                   ;;GO READ ONE CHARACTER FROM THE TTY
1807  007020  112613                          MOVB    (SP)+,(R3)      ;;GET CHARACTER
1808  007022  122713  000177          10$:    CMPB    #177,(R3)       ;;IS IT A RUBOUT
1809  007026  001003                          BNE     3$              ;;SKIP IF NOT
1810  007030  104401  001212          4$:     TYPE    ,$QUES          ;;TYPE A '?'
1811  007034  000763                          BR      1$              ;;CLEAR THE BUFFER AND LOOP
1812  007036  111337  007106          3$:     MOVB    (R3),9$         ;;ECHO THE CHARACTER
1813  007042  104401  007106                  TYPE    ,9$
1814  007046  122723  000015                  CMPB    #15,(R3)+       ;;CHECK FOR RETURN
1815  007052  001356                          BNE     2$              ;;LOOP IF NOT RETURN
1816  007054  105063  177777                  CLRB    -1(R3)          ;;CLEAR RETURN (THE 15)
1817  007060  104401  001214                  TYPE    ,$LF            ;;TYPE A LINE FEED
1818  007064  012603                          MOV     (SP)+,R3        ;;RESTORE R3
1819  007066  011646                          MOV     (SP),-(SP)      ;;ADJUST THE STACK AND PUT ADDRESS OF THE
1820  007070  016666  000004  000002          MOV     4(SP),2(SP)     ;;      FIRST ASCII CHARACTER ON IT
1821  007076  012766  007110  000004          MOV     #$TTYIN,4(SP)
1822  007104  000002                          RTI                     ;;RETURN
1823  007106     000                  9$:     .BYTE   0               ;;STORAGE FOR ASCII CHAR. TO TYPE
1824  007107     000                          .BYTE   0               ;;TERMINATOR
1825  007110  000010                  $TTYIN: .BLKB   8.              ;;RESERVE 8 BYTES FOR TTY INPUT
1826  007120  052536  005015     000  $CNTLU: .ASCIZ  /^U/<15><12>    ;;CONTROL "U"
1827  007125     136  006507  000012  $CNTLG: .ASCIZ  /^G/<15><12>    ;;CONTROL "G"
1828  007132  005015  053523  020122  $MSWR:  .ASCIZ  <15><12>/SWR = /
1829  007140  020075     000
1830  007143     040  047040  053505  $MNEW:  .ASCIZ  /  NEW = /
1831  007150  036440  000040
1832
1833                                          .SBTTL  TYPE ROUTINE
1834
1835                                          ;;**************************************************************
1836                                          ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
1837                                          ;*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
                                        ;*
007154  105737  001157          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
007160  100002                         BPL     1$              ;;BR IF YES
007162  000000                         HALT                    ;;HALT HERE IF NO TERMINAL
007164  000407                         BR      3$              ;;LEAVE
007166  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
007170  017600  000002                 MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
007174  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
007176  001005                         BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
007200  005726                         TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
007202  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
007204  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
007210  000002                         RTI                     ;;RETURN
007212  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
007216  001430                         BEQ     8$
007220  122716  000200                 CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CR_F>
007224  001006                         BNE     5$
007226  005726                         TST     (SP)+           ;;POP  <CR><LF> EQUIV
007230  104401                         TYPE                    ;;TYPE A CR AND LF
007232  001213                         $CRLF
007234  105037  007370                 CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
007240  000755                         BR      2$              ;;GET NEXT CHARACTER
007242  004737  007324          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
007246  123726  001156          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
007252  001350                         BNE     2$              ;;IF NO GO GET NEXT CHAR.
007254  013746  001154                 MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                        ;;AND THE NULL CHAR.
007260  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
007264  002770                         BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
007266  004737  007324                 JSR     PC,$TYPEC       ;;GO TYPE A NULL
007272  105337  007370                 DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
007276  000770                         BR      7$              ;;LOOP

                                        ;HORIZONTAL TAB PROCESSOR

007300  112716  000040          8$:     MOVB    #' ,(SP)        ;;REPLACE TAB WITH SPACE
007304  004737  007324          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
007310  132737  000007  007370         BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
007316  001372                         BNE     9$              ;;TAB STOP
007320  005726                         TST     (SP)+           ;;POP SPACE OFF STACK
007322  000724                         BR      2$              ;;GET NEXT CHARACTER
007324  105777  171620          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
007330  100375                         BPL     $TYPEC
007332  116677  000002  171612         MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
007340  122766  000015  000002         CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
```

```
1894  007346  001003                        BNE     1$              ;;BRANCH IF NO
1895  007350  105037  007370                CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
1896  007354  000406                        BR      $TYPEX          ;;EXIT
1897  007356  122766  000012  000002  1$:   CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
1898  007364  001402                        BEQ     $TYPEX          ;;BRANCH IF YES
1899  007366  105227                        INCB    (PC)+           ;;COUNT THE CHARACTER
1900  007370  000000                $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
1901  007372  000207                $TYPEX: RTS     PC
1902
1903
1904                                  .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
1905
1906                                  ;;**************************************************************
1907                                  ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
1908                                  ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
1909                                  ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
1910                                  ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
1911                                  ;*REPLACED WITH SPACES.
1912                                  ;*CALL:
1913                                  ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
1914                                  ;*      TYPDS                   ;;GO TO THE ROUTINE
1915
1916  007374                        $TYPDS:
1917  007374  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
1918  007376  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
1919  007400  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
1920  007402  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
1921  007404  010546                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
1922  007406  012746  020200                MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
1923  007412  016605  000020                MOV     20(SP),R5       ;;GET THE INPUT NUMBER
1924  007416  100004                        BPL     1$              ;;BR IF INPUT IS POS.
1925  007420  005405                        NEG     R5              ;;MAKE THE BINARY NUMBER POS.
1926  007422  112766  000055  000001        MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
1927  007430  005000                1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
1928  007432  012703  007610                MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
1929  007436  112723  000040                MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
1930  007442  005002                2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
1931  007444  016001  007600                MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
1932  007450  160105                3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
1933  007452  002402                        BLT     4$              ;;BR IF DONE
1934  007454  005202                        INC     R2              ;;INCREASE THE BCD DIGIT BY 1
1935  007456  000774                        BR      3$
1936  007460  060105                4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
1937  007462  005702                        TST     R2              ;;CHECK IF BCD DIGIT=0
1938  007464  001002                        BNE     5$              ;;FALL THROUGH IF 0
1939  007466  105716                        TSTB    (SP)            ;;STILL DOING LEADING 0'S?
1940  007470  100407                        BMI     7$              ;;BR IF YES
1941  007472  106316                5$:     ASLB    (SP)            ;;MSD?
1942  007474  103003                        BCC     6$              ;;BR IF NO
1943  007476  116663  000001  177777        MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
1944  007504  052702  000060        6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
1945  007510  052702  000040        7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
1946  007514  110223                        MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
1947  007516  005720                        TST     (R0)+           ;;JUST INCREMENTING
1948  007520  020027  000010                CMP     R0,#10          ;;CHECK THE TABLE INDEX
1949  007524  002746                        BLT     2$              ;;GO DO THE NEXT DIGIT
```

```
                        007526  003002                  BGT     8$              ;;GO TO EXIT
                        007530  010502                  MOV     R5,R2           ;;GET THE LSD
                        007532  000764                  BR      6$              ;;GO CHANGE TO ASCII
                        007534  105726          8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
                        007536  100003                  BPL     9$              ;;BR IF NO
                        007540  116663  177777  177776  MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
                        007546  105013          9$:     CLRB    (R3)            ;;SET THE TERMINATOR
                        007550  012605                  MOV     (SP)+,R5        ;;POP STACK INTO R5
                        007552  012603                  MOV     (SP)+,R3        ;;POP STACK INTO R3
                        007554  012602                  MOV     (SP)+,R2        ;;POP STACK INTO R2
                        007556  012601                  MOV     (SP)+,R1        ;;POP STACK INTO R1
                        007560  012600                  MOV     (SP)+,R0        ;;POP STACK INTO R0
                        007562  104401  007610          TYPE    ,$DBLK          ;;NOW TYPE THE NUMBER
                        007566  016666  000002  000004  MOV     2(SP),4(SP)     ;;ADJUST THE STACK
                        007574  012616                  MOV     (SP)+,(SP)
                        007576  000002                  RTI                     ;;RETURN TO USER
                        007600  023420          $DTBL:  10000.
                        007602  001750                  1000.
                        007604  000144                  100.
                        007606  000012                  10.
                        007610  000004          $DBLK:  .BLKW   4

                                                .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

                        007620  017646  000000  $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
                        007624  116637  000001  010043  MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
                        007632  112637  010045          MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
                        007636  062716  000002          ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
                        007642  000406                  BR      $TYPON
                        007644  112737  000001  010043  $TYPOC: MOVB    #1,$OFILL       ;;SET THE ZERO FILL SWITCH
                        007652  112737  000006  010045          MOVB    #6,$OMODE+1     ;;SET FOR SIX(6) DIGITS
                        007660  112737  000005  010042  $TYPON: MOVB    #5,$OCNT        ;;SET THE ITERATION COUNT
                        007666  010346                  MOV     R3,-(SP)        ;;SAVE R3
```

```
2006  007670  010446                        MOV     R4,-(SP)        ;;SAVE R4
2007  007672  010546                        MOV     R5,-(SP)        ;;SAVE R5
2008  007674  113704  010045                MOVB    $OMODE+1,R4     ;;GET THE NUMBER OF DIGITS TO TYPE
2009  007700  005404                        NEG     R4
2010  007702  062704  000006                ADD     #6,R4           ;;SUBTRACT IT FOR MAX. ALLOWED
2011  007706  110437  010044                MOVB    R4,$OMODE       ;;SAVE IT FOR USE
2012  007712  113704  010043                MOVB    $OFILL,R4       ;;GET THE ZERO FILL SWITCH
2013  007716  016605  000012                MOV     12'SP),R5       ;;PICKUP THE INPUT NUMBER
2014  007722  005003                        CLR     R3              ;;CLEAR THE OUTPUT WORD
2015  007724  006105              1$:       ROL     R5              ;;ROTATE MSB INTO "C"
2016  007726  000404                        BR      3$              ;;GO DO MSB
2017  007730  006105              2$:       ROL     R5              ;;FORM THIS DIGIT
2018  007732  006105                        ROL     R5
2019  007734  006105                        ROL     R5
2020  007736  010503                        MOV     R5,R3
2021  007740  006103              3$:       ROL     R3              ;;GET LSB OF THIS DIGIT
2022  007742  105337  010044                DECB    $OMODE          ;;TYPE THIS DIGIT?
2023  007746  100016                        BPL     7$              ;;BR IF NO
2024  007750  042703  177770                BIC     #177770,R3      ;;GET RID OF JUNK
2025  007754  001002                        BNE     4$              ;;TEST FOR 0
2026  007756  005704                        TST     R4              ;;SUPPRESS THIS 0?
2027  007760  001403                        BEQ     5$              ;;BR IF YES
2028  007762  005204              4$:       INC     R4              ;;DON'T SUPPRESS ANYMORE 0'S
2029  007764  052703  000060                BIS     #'0,R3          ;;MAKE THIS DIGIT ASCII
2030  007770  052703  000040      5$:       BIS     #' ,R3          ;;MAKE ASCII IF NOT ALREADY
2031  007774  110337  010040                MOVB    R3,8$           ;;SAVE FOR TYPING
2032  010000  104401  010040                TYPE    ,8$             ;;GO TYPE THIS DIGIT
2033  010004  105337  010042      7$:       DECB    $OCNT           ;;COUNT BY 1
2034  010010  003347                        BGT     2$              ;;BR IF MORE TO DO
2035  010012  002402                        BLT     6$              ;;BR IF DONE
2036  010014  005204                        INC     R4              ;;INSURE LAST DIGIT ISN'T A BLANK
2037  010016  000744                        BR      2$              ;;GO DO THE LAST DIGIT
2038  010020  012605              6$:       MOV     (SP)+,R5        ;;RESTORE R5
2039  010022  012604                        MOV     (SP)+,R4        ;;RESTORE R4
2040  010024  012603                        MOV     (SP)+,R3        ;;RESTORE R3
2041  010026  016666  000002  000004        MOV     2(SP),4(SP)     ;;SET THE STACK FOR RETURNING
2042  010034  012616                        MOV     (SP)+,(SP)
2043  010036  000002                        RTI                     ;;RETURN
2044  010040     000              8$:       .BYTE   0               ;;STORAGE FOR ASCII DIGIT
2045  010041     000                        .BYTE   0               ;;TERMINATOR FOR TYPE ROUTINE
2046  010042     000              $OCNT:    .BYTE   0               ;;OCTAL DIGIT COUNTER
2047  010043     000              $OFILL:   .BYTE   0               ;;ZERO FILL SWITCH
2048  010044  000000              $OMODE:   .WORD   0               ;;NUMBER OF DIGITS TO TYPE
2049
2050                                        .SBTTL  TRAP DECODER
2051
2052                                        ;;**************************************************************
2053                                        ;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
2054                                        ;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
2055                                        ;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
2056                                        ;*GO TO THAT ROUTINE.
2057
2058  010046  010046              $TRAP:    MOV     R0,-(SP)        ;;SAVE R0
2059  010050  016600  000002                MOV     2(SP),R0        ;;GET TRAP ADDRESS
2060  010054  005740                        TST     -(R0)           ;;BACKUP BY 2
2061  010056  111000                        MOVB    (R0),R0         ;;GET RIGHT BYTE OF TRAP
```

```
2062  010060  006300                        ASL     R0              ;;POSITION FOR INDEXING
2063  010062  016000  010102                MOV     $TRPAD(R0),R0   ;;INDEX TO TABLE
2064  010066  000200                        RTS     R0              ;;GO TO ROUTINE
2065
2066
2067                                  ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO
2068
2069  010070  011646                $TRAP2: MOV     (SP),-(SP)      ;;MOVE THE PC DOWN
2070  010072  016666  000004  000002        MOV     4(SP),2(SP)     ;;MOVE THE PSW DOWN
2071  010100  000002                        RTI                     ;;RESTORE THE PSW
2072
2073                                  .SBTTL  TRAP TABLE
2074
2075                                  ;*THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
2076                                  ;*BY THE "TRAP" INSTRUCTION.
2077
2078                                  ;       ROUTINE
2079                                  ;       -------
2080  010102  010070                $TRPAD: .WORD   $TRAP2
2081  010104  007154                        $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
2082  010106  007644                        $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
2083  010110  007620                        $TYPOS  ;;CALL=TYPOS    TRAP+3(104403   TYPE OCTAL NUMBER (NO LEADING ZEROS)
2084  010112  007660                        $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
2085  010114  007374                        $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
2086
2087  010116  006450                        $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING
2088
2089  010120  006400                        $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
2090  010122  006662                        $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
2091  010124  007002                        $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE
2092
2093  010126  005506                        CN.RST  ;;CALL=CNT.RESET        TRAP+12(104412) CONTROL RESET ROUTINE
2094
2095  010130  005524                        CN.RDY  ;;CALL=CNT.RDY  TRAP+13(104413) WAIT FOR CNTRL RDY TO SET
2096
2097  010132  005464                        DELA.Y  ;;CALL=DELAY    TRAP+14(104414) TIME DELAY ROUTINE
2098
2099
2100                                  .SBTTL  POWER DOWN AND UP ROUTINES
2101
2102                                  ;;*****************************************************************
2103                                  ;POWER DOWN ROUTINE
2104  010134  012737  010300  000024 $PWRDN: MOV     #$I_LUP,@#PWRVEC ;;SET FOR FAST UP
2105  010142  012737  000340  000026        MOV     #340,@#PWRVEC+2 ;;PRIO:7
2106  010150  010046                        MOV     R0,-(SP)        ;;PUSH R0 ON STACK
2107  010152  010146                        MOV     R1,-(SP)        ;;PUSH R1 ON STACK
2108  010154  010246                        MOV     R2,-(SP)        ;;PUSH R2 ON STACK
2109  010156  010346                        MOV     R3,-(SP)        ;;PUSH R3 ON STACK
2110  010160  010446                        MOV     R4,-(SP)        ;;PUSH R4 ON STACK
2111  010162  010546                        MOV     R5,-(SP)        ;;PUSH R5 ON STACK
2112  010164  017746  170750                MOV     @SWR,-(SP)      ;;PUSH @SWR ON STACK
2113  010170  010637  010304                MOV     SP,$SAVR6       ;;SAVE SP
2114  010174  012737  010206  000024        MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
2115  010202  000000                        HALT
2116  010204  000776                        BR      .-2             ;;HANG UP
2117
```

```
                                        ;;*****************************************************************
                                        ;POWER UP ROUTINE
010206  012737  010300  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
010214  013706  010304          $PWRUP: MOV     $SAVR6,SP        ;;GET SP
010220  005037  010304                  CLR     $SAVR6           ;;WAIT LOOP FOR THE TTY
010224  005237  010304          1$:     INC     $SAVR6           ;;WAIT FOR THE INC
010230  001375                          BNE     1$               ;;OF  WORD
010232  012677  170702                  MOV     (SP)+,@#SWR      ;;POP STACK INTO @#SWR
010236  012605                          MOV     (SP)+,R5         ;;POP STACK INTO R5
010240  012604                          MOV     (SP)+,R4         ;;POP STACK INTO R4
010242  012603                          MOV     (SP)+,R3         ;;POP STACK INTO R3
010244  012602                          MOV     (SP)+,R2         ;;POP STACK INTO R2
010246  012601                          MOV     (SP)+,R1         ;;POP STACK INTO R1
010250  012600                          MOV     (SP)+,R0         ;;POP STACK INTO R0
010252  012737  010134  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
010260  012737  000340  000026          MOV     #340,@#PWRVEC+2  ;;PRIO:7
010266  104401                          TYPE                     ;REPORT THE POWER FAILURE
010270  010306          $PWRMG: .WORD   $POWER                   ;;POWER FAIL MESSAGE POINTER
010272  012716                  MOV     (PC)+,(SP)               ;;RESTART AT PFSTRT
010274  002306          $PWRAD: .WORD   PFSTRT                   ;;RESTART ADDRESS
010276  000002                  RTI
010300  000000          $ILLUP: HALT                             ;;THE POWER UP SEQUENCE WAS STARTED
010302  000776                  BR      .-2                      ;; BEFORE THE POWER DOWN WAS COMPLETE
010304  000000          $SAVR6: 0                                ;;PUT THE SP HERE
010306  005015  047520  042527  $POWER: .ASCIZ  <15><12>"POWER"
010314  000122
                                        .EVEN

                                        ;ERROR MESSAGES

                                        .SBTTL  ERROR MESSAGES

010316  044524  042515  047440  EM1:    .ASCIZ  /TIME OUT ON RK11 REGISTER/
010324  052125  047440  020116
010332  045522  030461  051040
010340  043505  051511  042524
010346  000122

010350  042522  044507  052123  EM2:    .ASCIZ  /REGISTER NOT CLEARED/
010356  051105  047040  052117
010364  041440  042514  051101
010372  042105     000

010375     122  041513  020123  EM3:    .ASCIZ  /RKCS ERROR/
010402  051105  047522  000122

010410  045522  051503  042440  EM4:    .ASCIZ  /RKCS ERROR-ON WRITING READ ONLY BITS/
010416  051122  051117  047455
010424  020116  051127  052111
010432  047111  020107  042522
010440  042101  047440  046116
010446  020131  044502  051524
010454     000

010455     102  051525  044440  EM5:    .ASCIZ  /BUS INIT DID NOT CLEAR RKCS/
010462  044516  020124  044504
```

```
2174  010470  020104  047516  020124
2175  010476  046103  040505  020122
2176  010504  045522  051503     000
2177
2178  010511     103  052116  046122  EM6:    .ASCIZ  /CNTRL RESET DIDN'T CLEAR RKCS, ON SETING 'GO'/
2179  010516  051040  051505  052105
2180  010524  042040  042111  023516
2181  010532  020124  046103  040505
2182  010540  020122  045522  051503
2183  010546  020054  047117  051440
2184  010554  052105  047111  020107
2185  010562  043447  023517     000
2186
2187  010567     103  052116  046122  EM7:    .ASCIZ  /CNTRL RDY DIDN'T SET AFTER CNTRL RESET/
2188  010574  051040  054504  042040
2189  010602  042111  023516  020124
2190  010610  042523  020124  043101
2191  010616  042524  020122  047103
2192  010624  051124  020114  042522
2193  010632  042523  000124
2194
2195  010636  042522  044507  052123  EM10:   .ASCIZ  /REGISTER NOT CLEARED/
2196  010644  051105  047040  052117
2197  010652  041440  042514  051101
2198  010660  042105     000
2199
2200  010663     122  053513  020103  EM11:   .ASCIZ  /RKWC ERROR/
2201  010670  051105  047522  000122
2202
2203
2204  010676  045522  040502  042440  EM13:   .ASCIZ  /RKBA ERROR/
2205  010704  051122  051117     000
2206
2207  010711     103  052116  046122  EM14:   .ASCIZ  /CNTRL RESET DIDN'T CLEAR REGISTER/
2208  010716  051040  051505  052105
2209  010724  042040  042111  023516
2210  010732  020124  046103  040505
2211  010740  020122  042522  044507
2212  010746  052123  051105     000
2213
2214  010753     122  042113  020101  EM15:   .ASCIZ  /RKDA ERROR/
2215  010760  051105  047522  000122
2216
2217  010766  052502  020123  047111  EM17:   .ASCIZ  /BUS INIT DIDN'T CLR REGISTR/
2218  010774  052111  042040  042111
2219  011002  023516  020124  046103
2220  011010  020122  042522  044507
2221  011016  052123  000122
2222
2223  011022  042101  051104  051505  EM20:   .ASCIZ  /ADDRESSING ERROR-TRIED TO ADDRESS REG1, GOT REG2/
2224  011030  044523  043516  042440
2225  011036  051122  051117  052055
2226  011044  044522  042105  052040
2227  011052  020117  042101  051104
2228  011060  051505  020123  042522
2229  011066  030507  020054  047507
```

```
2230  011074  020124  042522  031107
2231  011102     000
2232
2233  011103     104  042111  023516  EM22:   .ASCIZ  /DIDN'T CLEAR RKCS LOW BYTE/
2234  011110  020124  046103  040505
2235  011116  020122  045522  051503
2236  011124  046040  053517  041040
2237  011132  052131  000105
2238
2239  011136  044504  047104  052047  EM23:   .ASCIZ  /DIDN'T CLEAR RKCS HIGH BYTE/
2240  011144  041440  042514  051101
2241  011152  051040  041513  020123
2242  011160  044510  044107  041040
2243  011166  052131  000105
2244
2245  011172  051124  042511  020104  EM24:   .ASCIZ  /TRIED TO CLEAR RKCS 'BYTE', CHANGED 'REGIS'/
2246  011200  047524  041440  042514
2247  011206  051101  051040  041513
2248  011214  020123  041047  052131
2249  011222  023505  020054  044103
2250  011230  047101  042507  020104
2251  011236  051047  043505  051511
2252  011244  000047
2253
2254  011246  040506  046111  042105  EM25:   .ASCIZ  /FAILED TO CLEAR 'REG-BYTE'/
2255  011254  052040  020117  046103
2256  011262  040505  020122  051047
2257  011270  043505  041055  052131
2258  011276  023505     000
2259
2260  011301     122  041513  020123  EM26:   .ASCIZ  /RKCS ALTERED ON CLEARING 'REG-BYTE'/
2261  011306  046101  042524  042522
2262  011314  020104  047117  041440
2263  011322  042514  051101  047111
2264  011330  020107  051047  043505
2265  011336  041055  052131  023505
2266  011344     000
2267
2268  011345     124  044522  042105  EM27:   .ASCIZ  /TRIED TO CLEAR 'REG-BYT1', CHANGED 'REG-BYT2'/
2269  011352  052040  020117  046103
2270  011360  040505  020122  051047
2271  011366  043505  041055  052131
2272  011374  023461  020054  044103
2273  011402  047101  042507  020104
2274  011410  051047  043505  041055
2275  011416  052131  023462     000
2276
2277  011423     125  042516  050130  EM43:   .ASCIZ  /UNEXPECTED PK11 INTERRUPT/
2278  011430  041505  042524  020104
2279  011436  045522  030461  044440
2280  011444  052116  051105  052522
2281  011452  052120     000
2282
2283          011456                          .EVEN
2284
2285                                          .SBTTL  ERROR DATA POINTERS
```

```
2286
2287  011456  001116  001162  000000  DT1:    .WORD   $ERRPC,$REG0,0
2288
2289  011464  001116  001162  001164  DT2:    .WORD   $ERRPC,$REG0,$REG1,0
2290  011472  000000
2291
2292  011474  001116  001162  001164  DT20:   .WORD   $ERRPC,$REG0,$REG1,$REG2,$REG3,0
2293  011502  001166  001170  000000
2294
2295  011510  001116  000000          DT21:   .WORD   $ERRPC,0
2296
2297  011514  001116  001162  001164  DT26:   .WORD   $ERRPC,$REG0,$REG1,$REG2,0
2298  011522  001166  000000
2299
2300
2301
2302                                          .SBTTL  ERROR HEADERS
2303
2304  011526  020040  041520  020040  DH1:    .ASCIZ  /  PC    REG-ADDR/
2305  011534  051040  043505  040455
2306  011542  042104  000122
2307
2308  011546  020040  041520  020040  DH2:    .ASCIZ  /  PC    REGADD     RECVD/
2309  011554  051040  043505  042101
2310  011562  020104  020040  051040
2311  011570  041505  042126     000
2312
2313  011575     040  050040  020103  DH4:    .ASCIZ  /  PC     EXPCT    RECVD/
2314  011602  020040  042440  050130
2315  011610  052103  020040  051040
2316  011616  041505  042126     000
2317
2318  011623     040  050040  020103  DH3:    .ASCIZ  /  PC     WROTE    READ/
2319  011630  020040  053440  047522
2320  011636  042524  020040  051040
2321  011644  040505  000104
2322
2323  011650  020040  041520  020040  DH5:    .ASCIZ  /  PC     RECVD/
2324  011656  020040  042522  053103
2325  011664  000104
2326
2327  011666  020040  041520  020040  DH11:   .ASCIZ  /  PC     WROTE    READ/
2328  011674  020040  051127  052117
2329  011702  020105  020040  042522
2330  011710  042101     000
2331
2332  011713     040  050040  000103  DH21:   .ASCIZ  /  PC/
2333
2334  011720  020040  041520  020040  DH20:   .ASCIZ  /  PC     REG1      REG2    (REG1)   (REG2)/
2335  011726  020040  042522  030507
2336  011734  020040  020040  051040
2337  011742  043505  020062  020040
2338  011750  051050  043505  024461
2339  011756  020040  051050  043505
2340  011764  024462     000
2341
```

```
2342  011767    040  050040  020103  DH24:   .ASCIZ  /  PC     BYTE   REGIS (REG)EXP (REG)RECVD/
2343  011774  020040  020040  054502
2344  012002  042524  020040  051040
2345  012010  043505  051511  024040
2346  012016  042522  024507  054105
2347  012024  020120  051050  043505
2348  012032  051051  041505  042126
2349  012040     000
2350
2351  012041    040  050040  020103  DH25:   .ASCIZ  /  PC   REG-BYTE RECVD/
2352  012046  020040  042522  026507
2353  012054  054502  042524  051040
2354  012062  041505  042126     000
2355
2356  012067    040  050040  020103  DH26:   .ASCIZ  /  PC   REG-BYT  (CS)EXP (CS)RECVD/
2357  012074  020040  042522  026507
2358  012102  054502  020124  024040
2359  012110  051503  042451  050130
2360  012116  024040  051503  051051
2361  012124  041505  042126     000
2362
2363  012131    040  050040  020103  DH27:   .ASCIZ  /  PC    R-BYT1  R-BYT2  2-EXPCT 2-RECVD/
2364  012136  020040  051040  041055
2365  012144  052131  020061  051040
2366  012152  041055  052131  020062
2367  012160  031040  042455  050130
2368  012166  052103  031040  051055
2369  012174  041505  042126     000
2370
2371  012201    040  050040  020103  DH30:   .ASCIZ  /  PC     RKCS    RKER    RKDS/
2372  012206  020040  020040  045522
2373  012214  051503  020040  020040
2374  012222  045522  051105  020040
2375  012230  020040  045522  051504
2376  012236     000
2377
2378
2379
2380          000001                        .END
```

```
BADINT  002202
BADTMO  002116
BIT0  = 000001
BIT00 = 000001
BIT01 = 000002
BIT02 = 000004
BIT03 = 000010
BIT04 = 000020
BIT05 = 000040
BIT06 = 000100
BIT07 = 000200
BIT08 = 000400
BIT09 = 001000
BIT1  = 000002
BIT10 = 002000
BIT11 = 004000
BIT12 = 010000
BIT13 = 020000
BIT14 = 040000
BIT15 = 100000
BIT2  = 000004
BIT3  = 000010
BIT4  = 000020
BIT5  = 000040
BIT6  = 000100
BIT7  = 000200
BIT8  = 000400
BIT9  = 001000
BPTVEC= 000014
CKSWR = 104407
CNT.RD= 104413
CNT.RE= 104412
CN.RDY  005524
CN.RST  005506
CR    = 000015
CRLF  = 000200
DCISP = 177570
DELAY = 104414
DELA.Y  005464
DH1     011526
DH11    011666
DH2     011546
DH20    011720
DH21    011713
DH24    011767
DH25    012041
DH26    012067
DH27    012131
DH3     011623
DH30    012201
DH4     011575
DH5     011650
DISPLA  001142
DISPRE  000174
DSWR  = 177570
DT1     011456
DT2     011464
DT20    011474
DT21    011510
DT26    011514
EMTVEC= 000030
EM1     010316
EM10    010636
EM11    010663
EM13    010676
EM14    010711
EM15    010753
EM17    010766
EM2     010350
EM20    011022
EM22    011103
EM23    011136
EM24    011172
EM25    011246
EM26    011301
EM27    011345
EM3     010375
EM4     010410
EM43    011423
EM5     010455
EM6     010511
EM7     010567
ERRVEC= 000004
FTITLE  001262
GTSWR = 104406
GT3RG   005424
GT4RG   005450
HT    = 000011
IOTVEC= 000020
LF    = 000012
MSG3    001216
PFSTRT  002306
PIRQ  = 177772
PIRQVE= 000240
PR0   = 000000
PR1   = 000040
PR2   = 000100
PR3   = 000140
PR4   = 000200
PR5   = 000240
PR6   = 000300
PR7   = 000340
PS    = 177776
PSW   = 177776
PWRVEC= 000024
RDCHR = 104410
RDLIN = 104411
RESVEC= 000010
RKBA    001254
RKCS    001250
RKDA    001256
RKDB    001260
RKDS    001244
RKER    001246
RKPRI   001266
RKVEC   001270
RKWC    001252
R6    =%000006
R7    =%000007
STACK = 001100
START   001542
START1  002100
STKLMT= 177774
SWR     001140
SWREG   000176
SW0   = 000001
SW00  = 000001
SW01  = 000002
SW02  = 000004
SW03  = 000010
SW04  = 000020
SW05  = 000040
SW06  = 000100
SW07  = 000200
SW08  = 000400
SW09  = 001000
SW1   = 000002
SW10  = 002000
SW11  = 004000
SW12  = 010000
SW13  = 020000
SW14  = 040000
SW15  = 100000
SW2   = 000004
SW3   = 000010
SW4   = 000020
SW5   = 000040
SW6   = 000100
SW7   = 000200
SW8   = 000400
SW9   = 001000
TBITVE= 000014
TIMER   001264
TIMOUT  002456
TKVEC = 000060
TPVEC = 000064
TRAPVE= 000034
TRTVEC= 000014
TST1    002352
TST10   003204
TST11   003314
TST12   003454
TST13   003522
TST14   003630
TST15   003676
TST16   004004
TST17   004056
TST2    002474
TST20   004164
TST21   004240
TST22   004400
TST23   004570
TST24   005004
TST3    002560
TST4    002640
TST5    002720
TST6    003072
TST7    003142
TYPDS = 104405
TYPE  = 104401
TYPOC = 104402
TYPON = 104404
TYPOS = 104403
T1      002402
$AUTOB  001134
$BDADR  001122
$BDDAT  001126
$CHARC  007370
$CKSWR  006400
$CMTAG  001100
$CM1  = 000012
$CM2  = 000024
$CM3  = 000012
$CNTLG  007125
$CNTLU  007120
$CRLF   001213
$DBLK   007610
$DOAGN  005400
$DTBL   007600
$ENDAD  005370
$ENDCT  005336
$ENDMG  005407
$ENULL  005404
$EOP    005302
$EOPCT  005330
$ERFLG  001103
$ERMAX  001115
$ERROR  006104
$ERRPC  001116
$ERRTB  001272
$ERRTY  006244
$ERTTL  001112
$ESCAP  001210
$FILLC  001156
$FILLS  001155
$GDADR  001120
$GDDAT  001124
$GET42  005360
$GTSWR  006450
$HD   = 000000
$ICNT   001104
$ILLUP  010300
$INTAG  001135
$ITEMB  001114
$LF     001214
$LPADR  001106
$LPERR  001110
$MNEW   007143
$MSWR   007132
$MXCNT  006102
$NULL   001154
$NWTST= 000001
$OCNT   010042
$OMODE  010044
$OVER   006066
$PASS   001100
$POWER  010306
$PWRAD  010274
$PWRDN  010134
$PWRMG  010270
$PWRUP  010206
$QUES   001212
$RDCHR  006662
$RDLIN  007002
$RDSZ = 000010
$REGAD  001160
$REG0   001162
$REG1   001164
$REG10  001202
$REG11  001204
$REG2   001166
$REG3   001170
$REG4   001172
$REG5   001174
$REG6   001176
$REG7   001200
$RTNAD  005402
$SAVR6  010304
$SCOPE  005632
$SETUP= 000117
$STUP = 177777
$SVLAD  006040
$SVPC = 000204
$SWR  = 165400
$SWRMK= 000000
```

```
MAINDEC-11-DZRKJ-D     MACY11 27(1006)  04-OCT-76  13:08  PAGE 47
DZRKJD.P11    30-AUG-76 14:48          SYMBOL TABLE                                                    SEQ 0059

$TIMES  001206          $TPFLG  001157          $TRPAD  010102          $TYPEC  007324          $XTSTR  005644
$TKB    001146          $TPS    001150          $TSTNM  001102          $TYPEX  007372          $$GET4= 000000
$TKS    001144          $TRAP   010046          $TTYIN  007110          $TYPOC  007644          $OFILL  010043
$TN   = 000025          $TRAP2  010070          $TYPDS  007374          $TYPON  007660          .       = 012237
$TPB    001152          $TRP  = 000015          $TYPE   007154          $TYPOS  007620

. ABS.  012237     000


 ERRORS DETECTED:  0
 DEFAULT GLOBALS GENERATED:  0

 DZRKJD,DZRKJD/LI:ME/NL:MC:MD:CND/SOL/NSQ←DZRKJD.P11
 RUN-TIME: 43 24 .9 SECONDS
 RUN-TIME RATIO: 253/69=3.6
 CORE USED:  23K  (45 PAGES)
```

Spooler runtime 9 Seconds, 43 KCS, 252 disk reads, 4 disk writes, 59 pages

[redacted]