RK11/05F/J

RK11 BASIC LOGIC TEST 1

MD-11-DZRKJ-E

EP-DZRKJ-E-DL-A

COPYRIGHT © 75-77

FICHE 1 OF 1

AUG 1977

digital

MADE IN USA

EOF1D2RKHGSEQ	00010000	770712	PDP10 411	HDR1DZRKJESEQ	00010000	770712

.REM %

IDENTIFICATION

PRODUCT CODE:	MAINDEC-11-DZRKJ-E-D
PRODUCT NAME:	RK11 BASIC LOGIC TEST I
DATE CREATED:	APRIL, 1977
MAINTAINER:	DIAGNOSTIC GROUP
AUTHOR:	JIM KAPADIA
REVISED BY:	PERVEZ ZAKI TOM SAWYER CHUCK HESS

THE INFORMATION IN THIS DOCUMENT IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION. DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR ANY ERRORS THAT MAY APPEAR IN THIS MANUAL.

THE SOFTWARE DESCRIBED IN THIS DOCUMENT IS FURNISHED TO THE PURCHASER UNDER A LICENSE FOR USE ON A SINGLE COMPUTER SYSTEM AND CAN BE COPIED (WITH INCLUSION OF DIGITAL'S COPYRIGHT NOTICE) ONLY FOR USE IN SUCH SYSTEM, EXCEPT AS MAY OTHERWISE BE PROVIDED IN WRITING BY DIGITAL.

DIGITAL EQUIPMENT CORPORATION ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL.

COPYRIGHT (C) 1975,1977 BY DIGITAL EQUIPMENT CORPORATION

QUICK LOOK-UP OPERATING INSTRUCTIONS

FOR A QUICK REFERENCE, LOOK UP THE FOLLOWING SECTIONS:
1.0 ABSTRACT
2.0 REQUIREMENTS
4.1 LOADING AND OPERATOR ACTION
5.0 SWITCH OPTIONS
FOR A MORE COMPLETE EXPLANATION REFER TO THE TABLE OF CONTENTS BELOW AND THE FOLLOWING DOCUMENT.

TABLE OF CONTENTS

1.0 ABSTRACT

2.0 REQUIREMENTS
2.1 EQUIPMENT
2.2 PRELIMINARY PROGRAMS
2.3 EXECUTION TIME

3.0 STARTING ADDRESS

4.0 PROGRAM CONTROL MODES & OPERATOR ACTION
4.1 PAPER TAPE
4.2 RKDP DUMP MODE
4.3 RKDP CHAIN MODE
4.4 ACT11

5.0 SWITCH OPTIONS

6.0 SCOPE LOOPS

7.0 PROGRAM STRUCTURE

8.0 ERROR REPORTING

9.0 ERROR INTERPRETATION

10.0 HANDLERS AND COMMON ROUTINES
10.1 TRAP HANDLER
10.2 SCOPE HANDLER
10.3 ERROR HANDLER
10.4 CONTROL RESET ROUTINE
10.5 CONTROL READY ROUTINE
10.6 TIME DELAY ROUTINE
10.7 OTHER ROUTINES
TTY HANDLER (I/O), ERROR TYPEOUT ROUTINE
POWER DOWN/POWER UP ROUTINE

11.0 UNEXPECTED TIMEOUTS & RK11 INTERRUPTS
12.0 QUICK VERYFYING MODE

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.

PRESS START.

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

RK11 LOGIC TEST I
MAINDEC-11-DZRKJ-E

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 I
MAINDEC-11-DZRKJ-E
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, IT PROCEEDS WITH THE EXECUTION OF THE TESTS AS BEFORE, BUT THE TITLE IS NOT TYPED OUT.

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 INTERRUPTIOCBURS 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  MD-11-DZRKJ-E, RK11 BASIC LOGIC TEST 1
                                ;*COPYRIGHT (C) 1974,1977
                                ;*DIGITAL EQUIPMENT CORP.
                                ;*MAYNARD, MASS. 01754
                                ;*
                                ;*PROGRAM BY JIM KAPADIA
                                ;*
                                ;*THIS PROGRAM WAS ASSEMBLED USING THE PDP-11 MAINDEC SYSMAC
                                ;*PACKAGE (MAINDEC-11-DZQAC-C3), JAN 19, 1977.
                                ;*
                                ;*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
```

```
838          000046                    .=46
839  000046  005400                    $ENDAD          ;;1)SET LOC.46 TO ADDRESS OF $ENDAD IN .$EOP
840          000052                    .=52
841  000052  000000                    .WORD   0       ;;2)SET LOC.52 TO ZERO
842          000204                    .=$SVPC         ;; RESTORE PC
843
```

```
                                        .SBTTL  COMMON TAGS

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

                 001100                         .=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/
```

```
900  001224  042122  020131  044504
901  001232  047104  052047  051440
902  001240  052105     000
903          001244                          .EVEN
904
905                                  ;RK11 REGISTERS
906                                  ;IF FOR ANY REASON THE REGISTER ADDRESSES ARE DIFFERENT FROM THESE
907                                  ;(GIVEN BELOW), THE CONTENTS OF THE APPROPRIATE POINTERS SHOULD BE
908                                  ;MODIFIED SO THAT THE CORRECT ADDRESS IS USED.
909                                  ;
910                                          .EVEN
911  001244  177400                  RKDS:   177400
912  001246  177402                  RKER:   177402
913  001250  177404                  RKCS:   177404
914  001252  177406                  RKWC:   177406
915  001254  177410                  RKBA:   177410
916  001256  177412                  RKDA:   177412
917  001260  177416                  RKDB:   177416
918
919
920                                  ;TAGS AND GENERAL DATA AREA
921                                  ;
922                                  ;
923
924  001262  000000                  FTITLE: 0               ;FLAG FOR PRINTING PROGRAM TITLE
925  001264  000000                  TIMER:  0               ;TIMER REGISTER
926  001266  000200                  RKPRI:  200             ;CONTAINS THE CPU LEVEL AT WHICH
927                                                          ;RK11 NORMALLY INTERRUPTS. THIS WORD
928                                                          ;SHOULD BE CHANGED IF RK11 IS DESINGATED
929                                                          ;A BR LEVEL OTHER THAN 5. E.G. IF IT IS CHANGED
930                                                          ;TO 6, THIS WORD SHOULD BE CHANGED TO 240.
931  001270  000220                  RKVEC:  220             ;CONTAINS THE NORMAL VECTOR ADDRESS TO WHICH
932                                                          ;RK11 INTERRUPTS. IF THIS IS NOT SO, CHANGE
933                                                          ;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  010336                         EM1     ;TIME OUT ON RK11 REG
 001274  011546                         DH1     ;PC     REG
 001276  011476                         DT1     ;$ERRPC $REG0
 001300  000000                         0

                                ;ITEM   2

 001302  010370                         EM2     ;REGISTER NOT CLEARED
 001304  011566                         DH2     ;PC     REGADD  RECVD
 001306  011504                         DT2     ;$ERRPC $REG0   $REG1
 001310  000000                         0

                                ;ITEM   3

 001312  010415                         EM3     ;RKCS ERROR
 001314  011643                         DH3     ;PC     WROTE   READ
 001316  011504                         DT2     ;$ERRPC $REG0   $REG1
 001320  000000                         0

                                ;ITEM   4

 001322  010430                         EM4     ;RKCS ERROR-ON WRITING READ ONLY BITS
 001324  011615                         DH4     ;PC     EXPCT   RECVD
 001326  011504                         DT2     ;$ERRPC $REG0   $REG1
 001330  000000                         0

                                ;ITEM   5

 001332  010475                         EM5     ;BUS INIT DID NOT CLEAR RKCS
 001334  011670                         DH5     ;PC     RECVD
 001336  011476                         DT1     ;$ERRPC $REG0
 001340  000000                         0

                                ;ITEM   6

 001342  010531                         EM6     ;'CNTRL RESET' DIDN'T CLEAR RKCS, ON SETING GO
 001344  011670                         DH5     ;PC     RECVD
 001346  011476                         DT1     ;$ERRPC $REG0
 001350  000000                         0

                                ;ITEM   7

 001352  010607                         EM7     ;'CNTRL RDY' DIDN'T SET AFTER CONTROL RESET
 001354  012221                         DH30    ;PC     RKCS    RKER    RKDS
 001356  011534                         DT26    ;$ERRPC $REG0   $REG1   $REG2
 001360  000000                         0

                                ;ITEM   10

 001362  010656                         EM10    ;REGISTER NOT CLEARED
 001364  011566                         DH2     ;PC     REGADD  RECVD
 001366  011504                         DT2     ;$ERRPC $REG0   $REG1
 001370  000000                         0
```

```
                                ;ITEM   11

001372  010703                          EM11    ;RKWC ERROR
001374  011706                          DH11    ;PC     WROTE   READ
001376  011504                          DT2     ;$ERRPC $REG0   $REG1
001400  000000                          0

                                ;ITEM   12

001402  011443                          EM43    ;UNEXPECTED RK11 INTERRUPT
001404  011733                          DH21    ;PC
001406  011530                          DT21    ;$ERRPC
001410  000000                          0

                                ;ITEM   13

001412  010716                          EM13    ;RKBA ERROR
001414  011706                          DH11    ;PC     WROT    READ
001416  011504                          DT2     ;$ERRPC $REG0   $REG1
001420  000000                          0

                                ;ITEM   14

001422  010731                          EM14    ;CNTRL RESET DID NOT CLEAR REGISTER
001424  011566                          DH2     ;PC     REGADD  RECVD
001426  011504                          DT2     ;$ERRPC $REG0   $REG1
001430  000000                          0

                                ;ITEM   15

001432  010773                          EM15    ;RKDA ERROR
001434  011706                          DH11    ;PC     WROTE   READ
001436  011504                          DT2     ;$ERRPC $REG0   $REG1
001440  000000                          0

                                ;ITEM   16

001442  011321                          EM26    ;RKCS ALTERERED ON CLEARING 'REG-BYTE'
001444  012107                          DH26    ;PC     REG-BYTE (RKCS)EXP (RKCS)RECVD
001446  011534                          DT26    ;$ERRPC $REG0   $REG1   $REG2
001450  000000                          0

                                ;ITEM   17

001452  011006                          EM17    ;BUS INIT DIDN'T CLEAR REGISTER
001454  011566                          DH2     ;PC     REGADD  RECVD
001456  011504                          DT2     ;$ERRPC $REG0   $REG1
001460  000000                          0

                                ;ITEM   20

001462  011042                          EM20    ;ADDRESSING ERROR-TRIED TO ADDRESS REG1, GOT REG2
001464  011740                          DH20    ;PC     REG1    REG2    (REG1)  (REG2)
001466  011514                          DT20    ;$ERRPC $REG0   $REG1   $REG2   $REG3
001470  000000                          0
```

```
                                ;ITEM   21

001472  011365                          EM27    ;TRIED TO CLEAR 'REG-BYTE', CHANGED 'REG-BYT2'
001474  012151                          DH27    ;PC     REG-BYT1        REG-BYT2        BYT2-EXPCT      BYT2-RECVD
001476  011514                          DT20    ;$ERRPC $REG0           $REG1           $REG2           $REG3
001500  000000                          0

                                ;ITEM   22

001502  011123                          EM22    ;DID NOT CLEAR RKCS LO BYTE
001504  011615                          DH4     ;PC     EXPCT   RECVD
001506  011504                          DT2     ;$ERRPC $REG0   $REG1
001510  000000                          0

                                ;IEM    23

001512  011156                          EM23    ;DID NOT CLEAR RKCS HI BYTE
001514  011615                          DH4     ;PC     EXPCT   RECVD
001516  011504                          DT2     ;$ERRPC $REG0   $REG1
001520  000000                          0

                                ;ITEM   24

001522  011212                          EM24    ;TRIED TO CLEAR RKCS 'BYTE', CHANGED 'REGIS'
001524  012007                          DH24    ;PC     BYTE    REGIS   (REG)EXP        (REG)RECVD
001526  011514                          DT20    ;$ERRPC $REG0   $REG1   $REG2   $REG3
001530  000000                          0

                                ;ITEM   25

001532  011266                          EM25    ;FAILED TO CLEAR 'REG-BYTE'
001534  012061                          DH25    ;PC     REG-BYTE RECVD
001536  011504                          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                          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  005642  000020          MOV     #$SCOPE,@#IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
001572  012737  000340  000022          MOV     #340,@#IOTVEC+2 ;;LEVEL 7
001600  012737  006114  000030          MOV     #$ERROR,@#EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
001606  012737  000340  000032          MOV     #340,@#EMTVEC+2 ;;LEVEL 7
001614  012737  010066  000034          MOV     #$TRAP,@#TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
001622  012737  000340  000036          MOV     #340,@#TRAPVEC+2;LEVEL 7
001630  012737  010154  000024          MOV     #$PWRDN,@#PWRVEC ;;POWER FAILURE VECTOR
001636  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;LEVEL 7
001644  005037  001206                  CLR     $TIMES          ;;INITIALIZE NUMBER OF ITERATIONS
```

```
001650  005037  001210                  CLR     $ESCAPE          ;;CLEAR THE ESCAPE ON ERROR ADDRESS
001654  112737  000001  001115          MOVB    #1,$ERMAX        ;;ALLOW ONE ERROR PER TEST
001662  012737  001662  001106          MOV     #.,$LPADR        ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
001670  012737  001670  001110          MOV     #.,$LPERR        ;;SETUP THE ERROR LOOP ADDRESS
                                ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
                                ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
001676  013746  000004                  MOV     @#ERRVEC,-(SP)   ;;SAVE ERROR VECTOR
001702  012737  001736  000004          MOV     #64$,@#ERRVEC    ;;SET UP ERROR VECTOR
001710  012737  177570  001140          MOV     #DSWR,SWR        ;;SETUP FOR A HARDWARE SWICH REGISTER
001716  012737  177570  001142          MOV     #DDISP,DISPLAY   ;;AND A HARDWARE DISPLAY REGISTER
001724  022777  177777  177206          CMP     #-1,@SWR         ;;TRY TO REFERENCE HARDWARE SWR
001732  001012                          BNE     66$              ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
                                                                 ;;AND  THE HARDWARE SWR IS NOT = -1
001734  000403                          BR      65$              ;;BRANCH IF NO TIMEOUT
001736  012716  001744          64$:    MOV     #65$,(SP)        ;;SET UP FOR TRAP RETURN
001742  000002                          RTI
001744  012737  000176  001140  65$:    MOV     #SWREG,SWR       ;;POINT TO SOFTWARE SWR
001752  012737  000174  001142          MOV     #DISPREG,DISPLAY
001760  012637  000004          66$:    MOV     (SP)+,@#ERRVEC   ;;RESTORE ERROR VECTOR

001764  023737  000042  000046          CMP     @#42,@#46        ;ARE WE IN ACT11 AUTOMATIC MODE?
001772  001416                          BEQ     69$              ;IF YES, SKIP TITLE
                                .SBTTL  TYPE PROGRAM NAME
                                ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
001774  005227  177777                  INC     #-1              ;;FIRST TIME?
002000  001043                          BNE     67$              ;;BRANCH IF NO
002002  104401  002040                  TYPE    ,68$             ;;TYPE ASCIZ STRING
                                .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
002006  005737  000042                  TST     @#42             ;;ARE WE RUNNING UNDER XXDP/ACT?
002012  001006                          BNE     69$              ;;BRANCH IF YES
002014  023727  001140  000176          CMP     SWR,#SWREG       ;;SOFTWARE SWITCH REG SELECTED?
002022  001005                          BNE     70$              ;;BRANCH IF NO
002024  104406                          GTSWR                    ;;GET SOFT-SWR SETTINGS
002026  000403                          BR      70$
002030  112737  000001  001134  69$:    MOVB    #1,$AUTOB        ;;SET AUTO-MODE INDICATOR
002036                          70$:
002036  000424                          BR      67$              ;;GET OVER THE ASCIZ
                                ;;68$:  .ASCIZ  <CRLF>/RK11 LOGIC TEST I/<15><12>/MAINDEC-11-DZRKJ-E/<CRLF>
002110                          67$:
                                ;
002110  012737  002126  000004  START1: MOV     #BADTMO,@#4      ;SET TIME OUT VECTOR FOR UNEXPECTED
                                                                 ;TIME OUTS
002116  012777  002212  177144          MOV     #BADINT,@RKVEC   ;SET UP RK11 INTERRUPT VECTOR FOR
                                                                 ;UNEXPECTED INTERRUPTS FROM RK11
002124  000516                          BR      TST1             ;GO TO TEST 1


                                        ;THIS ROUTINE HANDLES UNEXPECTED TIME OUTS

002126  011600                  BADTMO: MOV     (SP),R0 ;SAVE PC WHERE TIME OUT OCCURED
002130  005740                          TST     -(R0)
002132  022626                          CMP     (SP)+,(SP)+      ;RESTORE STACK POINTER
002134  104401  002142                  TYPE    ,65$             ;;TYPE ASCIZ STRING
002140  000417                          BR      64$              ;;GET OVER THE ASCIZ
```

```
                                        ;;65$:  .ASCIZ  <15><12>/UNEXPECTED TIME OUT AT PC=/
002200                                  64$:
002200  010046                                  MOV     R0,-(SP)        ;SET UP FOR TYPING OUT PC
002202  104402                                  TYPOC                   ;GO TYPE OUT OCTAL PC
002204  000000                                  HALT
002206  000137  001542                          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.

002212  011600                          BADINT: MOV     (SP),R0         ;SAVE PC WHERE INTERRUPT OCCURED
002214  005740                                  TST     -(R0)
002216  032777  020000  176714                  BIT     #20000,@SWR     ;INHIBIT ERROR TYPEOUT?
002224  001015                                  BNE     1$              ;YES, DON'T TYPE OUT
002226  104401                                  TYPE
002230  001213                                  $CRLF
002232  104401                                  TYPE
002234  011443                                  EM43                    ;TYPE 'UNEXPEXTED RK11 INTERRUPT'
                                                                        ;TYPE ' AT PC='
002236  104401  002244                          TYPE    ,65$            ;;TYPE ASCIZ STRING
002242  000404                                  BR      64$             ;;GET OVER THE ASCIZ
                                        ;;65$:  .ASCIZ  / AT PC=/
002254                                  64$:
002254  010046                                  MOV     R0,-(SP)        ;SET UP FOR TYPING OUT PC
002256  104402                                  TYPOC                   ;GO TYPE OCTAL PC WHERE BAD
                                                                        ;INTERUPT OCCURED
002260  032777  001000  176652          1$:     BIT     #1000,@SWR      ;LOOP ON ERROR?
002266  001403                                  BEQ     2$              ;NO, BRANCH
002270  022626                                  CMP     (SP)+,(SP)+     ;YES, REPOSITION STACK
002272  000177  176610                          JMP     @$LPADR         ;GO TO THE STARTING ADDRESS OF
                                                                        ;THE TEST THAT GAVE UNEXPECTED INTERRUPT
002276  032777  040000  176634          2$:     BIT     #40000,@SWR     ;LOOP ON TEST?
002304  001401                                  BEQ     3$              ;NO, BRANCH
002306  000002                                  RTI                     ;YES, LOOP. GO BACK WHER U INTERRUPTED FROM.
002310  000000                          3$:     HALT                    ;UNEXPECTED INTERRUPT OCCURED AS
                                                                        ;INDICATED IN THE TYPE OUT.U CAN LOOP
                                                                        ;ON ERROR, TEST,OR INHIBIT TYPEOUT BY
                                                                        ;SETTING APPROPRIATE SWITCHES.
002312  000137  001542                          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.

002316                                  PFSTRT:
002316  104401  002324                          TYPE    ,65$            ;;TYPE ASCIZ STRING
002322  000411                                  BR      64$             ;;GET OVER THE ASCIZ
                                        ;;65$:  .ASCIZ  <15><12>/PWR UP,RESTART/
002346                                  64$:
002346  005000                                  CLR     R0
```

```
002350  005001                          CLR     R1
002352  005201                  1$:     INC     R1
002354  001376                          BNE     1$
002356  105200                          INCB    R0
002360  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
                                ;;*****************************************************************
002362  000004                  TST1:   SCOPE
002364  005002                          CLR     R2
002366  012737  002412  001110          MOV     #T1,$LPERR      ;SET RETURN ADRES FOR LUP
                                                                ;ON EROR (SW9)
002374  012737  002466  000004          MOV     #TIMOUT,@#4     ;SET UP ADDRESS FOR TIMEOUT VECTOR
002402  012700  177771                  MOV     #-7,R0          ;INITIALIZE R0 TO KEEP TRACK OF REGIS REFERENCED
002406  012701  001244                  MOV     #RKDS,R1        ;INITIALIZE R1 WITH RKDS ADDRESS
002412  005731                  T1:     TST     @(R1)+          ;REFERENCE THE REGISTER,
                                                                ;IF IT CAN'T BE, TIMEOUT TRAP WILL OCCUR
                                                                ;SHIFT THE POINTER
002414  005200                          INC     R0              ;ALL REGISTERS REFERENCED?
002416  001375                          BNE     T1              ;IF NOT, LOOP BACK & REFERENCE NEXT REGISTER
002420  012737  002126  000004          MOV     #BADTMO,@#4
002426  005702                          TST     R2              ;WAS THERE AN ERROR?
002430  001415                          BEQ     1$              ;NO,BRANCH
002432  104401  002440                  TYPE    ,65$            ;;TYPE ASCIZ STRING
002436  000410                          BR      64$             ;;GET OVER THE ASCIZ
                                ;;65$:  .ASCIZ  <15><12>/PROG ABORTED/
002460                          64$:
002460  000137  005370                  JMP     $GET42          ;IF YES, ABORT THIS ENTIRE TEST
002464                          1$:
002464  000407                          BR      TST2            ;;EXIT
002466  022626                  TIMOUT: CMP     (SP)+,(SP)+
002470  014137  001162                  MOV     -(R1),$REG0     ;GET ADDRESS OF REGISTER THAT TIMED OUT
002474  104001                          ERROR   1               ;TIMED OUT WHEN REFERENCING RK11
                                                                ;REGISTERS
002476  005721                          TST     (R1)+           ;REPOSITION POINTER TO THE NEXT REGISTER ADDRESS
002500  005202                          INC     R2              ;SET FLAG INDICATING ERROR
002502  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.
                                ;;*****************************************************************
002504  000004                  TST2:   SCOPE
002506  000005                          RESET                   ;ISSUE A BUS INIT
002510  012700  177772                  MOV     #-6,R0          ;SET COUNT FOR 6 REGISTERS
002514  012701  001246                  MOV     #RKER,R1        ;INITIALIZE ADRES
002520  023711  001250          1$:     CMP     RKCS,@R1        ;IS IT RKCS?
```

```
002524  001005                          BNE     2$              ;NO
002526  022771  000200  000000          CMP     #200,@(R1)      ;ONLY BIT 7 OF RKCS SHOULD BE SET!
002534  001004                          BNE     3$              ;BRANCH IF ERROR
002536  000411                          BR      4$
002540  005771  000000          2$:     TST     @(R1)           ;CHECK THAT REST OF REGISTERS
                                                                ;ARE CLEAR
002544  001406                          BEQ     4$              ;BRANCH IF REGISTER IS CLR
002546  011137  001162          3$:     MOV     @R1,$REG0       ;GET ADRES OF REGISTER
002552  017137  000000  001164          MOV     @(R1),$REG1     ;GET CONTENTS OF REGISTER
002560  104002                          ERROR   2               ;RK11 REGISTER WAS FOUND TO B NOT CLEAR
002562  005721                  4$:     TST     (R1)+           ;INCREMNT POINTER TO NXT REG
002564  005200                          INC     R0              ;CHKD ALL REGS?
002566  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.
                                ;;*****************************************************************
002570  000004                  TST3:   SCOPE
002572  012737  002604  001110          MOV     #1$,$LPERR      ;SET RETURN ADRES FOR LOPING ON
                                                                ;ERROR (SW 9)
002600  012700  000002                  MOV     #2,R0           ;INITIALIZE BIT TO BE WRITTEN IN RKCS
002604  010077  176440          1$:     MOV     R0,@RKCS        ;WRITE THAT BIT IN RKCS
002610  017701  176434                  MOV     @RKCS,R1        ;GET RKCS
002614  042701  000200                  BIC     #200,R1         ;MASK OUT CNTRL RDY BIT
002620  020001                          CMP     R0,R1           ;WAS RECVD BIT SAME AS WRITTEN BIT
002622  001406                          BEQ     2$              ;YES, BRANCH OTHERWISE REPORT ERROR
002624  010037  001162                  MOV     R0,$REG0        ;GET EXPCTD RKCS, BIT THAT WAS WRITTEN
002630  017737  176414  001164          MOV     @RKCS,$REG1     ;GET RECVD RKCS, BIT READ
002636  104003                          ERROR   3               ;BIT THAT WAS WRITTEN WAS NOT READ BACK
002640  006300                  2$:     ASL     R0              ;SHIFT TO WRITE NEXT BIT
002642  020027  000020                  CMP     R0,#20          ;HAVE U CHKD ALL 3 BITS 1, 2, 3
002646  001356                          BNE     1$              ;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.
                                ;;*****************************************************************
002650  000004                  TST4:   SCOPE
002652  012737  002664  001110          MOV     #1$,$LPERR      ;SET RETURN ADRES FOR LUPING
                                                                ;ON EROR (SW 9)
002660  012700  000020                  MOV     #20,R0          ;INITIALIZE BIT TO BE WRITTEN IN RKCS
002664  010077  176360          1$:     MOV     R0,@RKCS        ;WRITE THAT BIT IN RKCS
002670  017701  176354                  MOV     @RKCS,R1        ;GET RKCS
002674  042701  000200                  BIC     #200,R1         ;MASK OUT CNTRL RDY BIT
002700  020001                          CMP     R0,R1           ;WAS RECVD BIT SAME AS WRITTEN BIT
002702  001406                          BEQ     2$              ;YES, BRANCH
002704  010037  001162                  MOV     R0,$REG0        ;GET EXPCTD RKCS, BIT THAT WAS WRITTEN
002710  017737  176334  001164          MOV     @RKCS,$REG1     ;GET RKCD RECVD, BIT THAT WAS READ
002716  104003                          ERROR   3               ;BIT THAT WAS WRITTEN WAS NOT READ BACK
002720  006300                  2$:     ASL     R0              ;SHIFT TO WRITE NEXT BIT
002722  022700  000100                  CMP     #100,R0         ;HAVE U CHKD BOTH BITS, 4, 5
002726  001356                          BNE     1$              ;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.
                                        ;;*****************************************************************
002730  000004                          TST5:   SCOPE
002732  012746  000340                          MOV     #340,-(SP)
002736  012746  002744                          MOV     #64$,-(SP)
002742  000002                                  RTI
002744                                  64$:
002744  013700  001250                          MOV     RKCS,R0
002750  012710  000100                          MOV     #100,@R0        ;SET THE IDE BIT
002754  022710  000300                          CMP     #300,@R0        ;WAS IT WRITTEN CORRECTLY
002760  001406                                  BEQ     1$              ;YES, BRANCH OTHERWISE REPORT ERROR
002762  012737  000300  001162                  MOV     #300,$REG0      ;GET EXPCT RKCS
002770  011037  001164                          MOV     @R0,$REG1       ;GET RKCS RECVD
002774  104003                                  ERROR   3               ;IDE BIT WAS WRITTEN, BUT WAS NOT
                                                                        ;READ BACK
002776  104412                          1$:     CNT.RESET               ;CONTROL RESET, CLEAR IDE
                                                                        ;THIS IS A CALL FOR THE 'CNTRL-
                                                                        ;RESET' ROUTINE.  A CONTROL RESET
                                                                        ;IS ISSUED AND AFTER A CERTAIN TIME
                                                                        ;IF THE 'CNTRL RDY' DOES NOT SET
                                                                        ;AN ERROR IS REPORTED. NOTE THAT
                                                                        ;THE PC IN ERROR MESSAGE IS THE
                                                                        ;PC WHERE 'CNT.RESET' IS LOCATED.
003000  022710  000200                          CMP     #200,@R0        ;DID IDE BIT GET CLRD?
003004  001406                                  BEQ     2$              ;YES, BRANCH
003006  010037  001162                          MOV     R0,$REG0        ;GET ADRES OF RKCS
003012  011037  001164                          MOV     @R0,$REG1       ;GET RKCS
003016  104010                                  ERROR   10              ;IDE BIT COULD NOT B CLRD
003020  000430                                  BR      TST6            ;;EXIT
003022  104414  000010                  2$:     DELAY   ,10             ;WAIT FOR AT LEAST 60 US
                                                                        ;ON 11/20  12 US FOR 11/45
003026  012777  003076  176234                  MOV     #3$,@RKVEC      ;SET RK11 INTERUPT VECTOR TO
                                                                        ;WHICH RK11 CAN INTERUPT IF THERE
                                                                        ;IS FAULTY LOGIC
003034  013746  001266                          MOV     RKPRI,-(SP)     ;LOWER CPU PRIORITY SO THAT
003040  012746  003046                          MOV     #4$,-(SP)       ;RK11 CAN POSSIBLY INTERRUPT IF
003044  000002                                  RTI                     ;THER IS MALFUNCTIONINIG LOGIC
003046  000240                          4$:     NOP                     ;THE INTERRUPT WOULD OCCUR
003050  000240                                  NOP
003052  000240                                  NOP
003054  012746  000340                          MOV     #340,-(SP)
003060  012746  003066                          MOV     #65$,-(SP)
003064  000002                                  RTI
003066                                  65$:
                                                                        ;PRIORITY
003066  012777  002212  176174                  MOV     #BADINT,@RKVEC  ;SET UP UNEXPCTD INTRUPT VECTOR
003074  000402                                  BR      TST6            ;;EXIT
003076  022626                          3$:     CMP     (SP)+,(SP)+     ;RESTORE STACK
003100  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
                                        ;;*****************************************************************
003102  000004                          TST6:   SCOPE
003104  012701  000400                          MOV     #400,R1         ;INITIALIZE BIT TO BE WRITTEN IN RKCS
003110  013702  001250                          MOV     RKCS,R2
003114  010112                          1$:     MOV     R1,@R2          ;WRITE THAT BIT IN RKCS
003116  011200                                  MOV     @R2,R0          ;GET RKCS
003120  042700  000200                          BIC     #200,R0         ;MASK CNTRL RDY BIT
003124  020100                                  CMP     R1,R0           ;WAS THE READ BIT SAME AS THE
                                                                        ;WRITTEN BIT
003126  001405                                  BEQ     2$              ;YES BRANCH, OTHERWISE REPORT ERROR
003130  010137  001162                          MOV     R1,$REG0        ;GET EXPTD RKCS
003134  011237  001164                          MOV     @R2,$REG1       ;GET RECVD RKCS
003140  104003                                  ERROR   3               ;BIT THAT WAS WRITTEN (AS IN $REG0)
                                                                        ;WAS NOT READ BACK
003142  006301                          2$:     ASL     R1              ;SHIFT TO WRITE NEXT BIT
003144  022701  010000                          CMP     #10000,R1       ;HAVE U CHECKED ALL BITS 8, 9, 10, 11
003150  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
                                        ;;*****************************************************************
003152  000004                          TST7:   SCOPE
003154  013700  001250                          MOV     RKCS,R0
003160  012710  170200                          MOV     #170200,@R0     ;TRY SETTING THE UNUSED BIT & RD
                                                                        ;ONLY BITS
003164  011001                                  MOV     @R0,R1          ;GET RKCS
003166  042701  010000                          BIC     #10000,R1       ;MASK BIT 12
003172  022701  000200                          CMP     #200,R1         ;IS 'RDY' BIT SET? NO OTHER
                                                                        ;BIT SHOULD BE SET.
003176  001406                                  BEQ     TST10           ;;OK, EXIT
003200  012737  000200  001162                  MOV     #200,$REG0      ;GET EXPCTD RKCS
003206  011037  001164                          MOV     @R0,$REG1       ;GET RKCS RECVD
003212  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.
                                        ;;*****************************************************************
003214  000004                          TST10:  SCOPE
003216  012746  000340                          MOV     #340,-(SP)
003222  012746  003230                          MOV     #64$,-(SP)
003226  000002                                  RTI
```

```
003230                          64$:
003230  013701  001250                  MOV     RKCS,R1
003234  012711  007576                  MOV     #7576,@R1       ;SET ALL BITS EXCEPT GO
003240  005000                          CLR     R0              ;
003242  000005                          RESET                   ;ISSUE BUS INIT
003244  022711  000200                  CMP     #200,@R1        ;CHECK IF RKCS WAS CLEARED?
003250  001403                          BEQ     1$              ;YES, BRANCH OTHERWISE REPORT ERROR
003252  011137  001162                  MOV     @R1,$REG0       ;GET RKCS
003256  104005                          ERROR   5               ;BUS INIT DID NOT CLEAR RKCS
003260  012711  005000          1$:     MOV     #5000,@R1       ;SET IBA & EXB IN RKCS
003264  005211                          INC     @R1             ;SET GO, CONTROL RESET
003266  005200                  2$:     INC     R0              ;KEEP TIME
003270  105700                          TSTB    R0              ;HAVE U WAITED LONG FOR CNTRL RDY
                                                                ;TO SET?
003272  100411                          BMI     3$              ;IF YES, BRANCH & REPORT ERROR
003274  105711                          TSTB    @R1             ;WAS CNTRL RDY SET?
003276  100373                          BPL     2$              ;IF NOT LOOP BACK & WAIT FOR IT
003300  022711  000200                  CMP     #200,@R1        ;IF CNTRL RDY WAS SET, CHK IF 'CNTRL
                                                                ;RESET' CLEARED IBA & EXB BITS
003304  001407                          BEQ     TST11           ;;IF YES, EXIT. OTHERWISE ERROR
003306  011137  001162                  MOV     @R1,$REG0       ;GET RKCS
003312  104006                          ERROR   6               ;GO BIT COULD NOT BE SET OR FAULT IN
                                                                ;THE 'INIT L' GENERATING LOGIC
003314  000403                          BR      TST11           ;;EXIT
003316  004737  005434          3$:     JSR     PC,GT3RG        ;GO, GET RKCS, ER, DS
003322  104007                          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
                                ;;*****************************************************************
003324  000004                  TST11:  SCOPE
003326  012737  000010  001206          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
003334  012746  000340                  MOV     #340,-(SP)
003340  012746  003346                  MOV     #64$,-(SP)
003344  000002                          RTI
003346                          64$:
003346  013700  001250                  MOV     RKCS,R0
003352  012710  007576                  MOV     #7576,@R0       ;SET ALL BITS IN RKCS EXCEPT GO
003356  005010                          CLR     @R0             ;CLEAR RKCS
003360  022710  000200                  CMP     #200,@R0        ;WAS IT CLEARED
003364  001405                          BEQ     1$              ;YES,BRANCH
003366  010037  001162                  MOV     R0,$REG0        ;GET ADRES OF RKCS
003372  011037  001164                  MOV     @R0,$REG1       ;NO, GET RKCS
003376  104010                          ERROR   10              ;RKCS COULD NOT BE CLEARED
003400  012701  000002          1$:     MOV     #2,R1           ;WRITE THIS BIT IN RKCS
003404  012737  003416  001110          MOV     #2$,$LPERR
003412  012705  177773                  MOV     #-5,R5
003416  010110                  2$:     MOV     R1,@R0          ;WRITE IT
003420  010102                          MOV     R1,R2           ;GET BIT THAT WAS WRITTEN
003422  052702  000200                  BIS     #200,R2         ;SET CNTRL RDY BIT
003426  011003                          MOV     @R0,R3
```

```
003430  020203                          CMP     R2,R3           ;WAS THAT BIT WRITTEN CORRECTLY?
003432  001407                          BEQ     3$              ;YES, BRANCH
003434  010237  001162                  MOV     R2,$REG0        ;GET EXPCTD WORD
003440  010337  001164                  MOV     R3,$REG1        ;GET RKCS RECVD
003444  104003                          ERROR   3               ;DID NOT READ BAK THE BIT THAT
                                                                ;WAS WRITTEN
003446  005205                          INC     R5
003450  001405                          BEQ     TST12           ;;EXIT
003452  062701  000002          3$:     ADD     #2,R1           ;GENERATE NXT PATTERN TO BE WRITTEN
003456  022701  010000                  CMP     #10000,R1       ;ALL PATTERNS WRITTEN?
003462  001355                          BNE     2$              ;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
                                ;;*****************************************************************
003464  000004                  TST12:  SCOPE
003466  012700  000001                  MOV     #1,R0           ;INITIALIZE R0 FOR THE BIT TO BE WRITTEN IN RKWC
003472  013701  001252                  MOV     RKWC,R1
003476  012737  003504  001110          MOV     #1$,$LPERR      ;SET UP RETURN ADRES FOR
                                                                ;LUPING ON ERROR (SW 9)
003504  010011                  1$:     MOV     R0,@R1          ;WRITE THAT BIT IN RKWC
003506  011102                          MOV     @R1,R2
003510  020002                          CMP     R0,R2           ;WAS IT WRITTEN CORRECTLY
003512  001405                          BEQ     2$              ;YES, BRANCH. OTHERWISE, ERROR
003514  010037  001162                  MOV     R0,$REG0        ;GET EXPCTD RKWC BIT THAT WAS WRITTEN
003520  010237  001164                  MOV     R2,$REG1        ;GET RKWC (THAT WAS READ BACK)
003524  104011                          ERROR   11              ;DID NOT READ BACK THE BIT THAT
                                                                ;WAS WRITTEN IN RKWC
003526  006300                  2$:     ASL     R0              ;SHIFT TO WRITE THE NXT BIT
003530  001365                          BNE     1$              ;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
                                ;;*****************************************************************
003532  000004                  TST13:  SCOPE
003534  012737  000010  001206          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
003542  013700  001252                  MOV     RKWC,R0
003546  012710  177777                  MOV     #177777,@R0     ;SET ALL BITS IN RKWC
003552  005010                          CLR     @R0             ;CLEAR RKWC
003554  005710                          TST     @R0             ;WAS IT CLEARED?
003556  001405                          BEQ     1$              ;YES, BRANCH
003560  010037  001162                  MOV     R0,$REG0        ;GET ADRES OF RKWC
003564  011037  001164                  MOV     @R0,$REG1       ;NO, GET RKWC
003570  104010                          ERROR   10              ;RKWC COULD NOT BE CLEARED

003572  005001                  1$:     CLR     R1              ;INITIALIZE COUNT PATTERN
003574  012705  177773                  MOV     #-5,R5
003600  012737  003606  001110          MOV     #2$,$LPERR
003606  010110                  2$:     MOV     R1,@R0          ;WRITE THE PATTERN IN THE REGISTER
003610  011002                          MOV     @R0,R2
003612  020102                          CMP     R1,R2           ;WAS IT WRIITEN CORRRECTLY?
```

```
003614  001407                          BEQ     3$              ;YES, BRANCH
003616  010137  001162                  MOV     R1,$REG0        ;GET EXPECTED WORD
003622  010237  001164                  MOV     R2,$REG1        ;GET WORD THAT WAS RECVD
003626  104011                          ERROR   11              ;DID NOT READ BACK THE PATTERN THAT
                                                                ;WAS WRITTEN INTO THE REGISTER
003630  005205                          INC     R5
003632  001402                          BEQ     TST14           ;;EXIT
003634  005201                  3$:     INC     R1              ;INCREMENT COUNT PATTERN
003636  001363                          BNE     2$              ;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.
                                ;;*****************************************************************
003640  000004                  TST14:  SCOPE
003642  012700  000001                  MOV     #1,R0           ;INITIALIZE R0 FOR BIT TO BE
                                                                ;WRITTEN IN RKBA
003646  013701  001254                  MOV     RKBA,R1
003652  012737  003660  001110          MOV     #1$,$LPERR      ;SET UP RETURN ADRES FOR
                                                                ;LUPING ON EROR (SW 12)
003660  010011                  1$:     MOV     R0,@R1          ;WRITE THAT BIT IN RKBA
003662  011102                          MOV     @R1,R2
003664  020002                          CMP     R0,R2           ;WAS IT WRITTEN CORRECTLY?
003666  001405                          BEQ     2$              ;YES, BRANCH
003670  010037  001162                  MOV     R0,$REG0        ;GET EXPCTD RKBA (BIT WRITTEN)
003674  010237  001164                  MOV     R2,$REG1        ;GET RKBA (BIT READ BACK)
003700  104013                          ERROR   13              ;DID NOT READ BACK TME BIT THAT
                                                                ;WAS WRITTEN IN RKBA
003702  006300                  2$:     ASL     R0              ;SHIFT R0, TO WRITE NEXT BIT
003704  001365                          BNE     1$              ;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.
                                ;;*****************************************************************
003706  000004                  TST15:  SCOPE
003710  012737  000010  001206          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
003716  013700  001254                  MOV     RKBA,R0
003722  012710  177777                  MOV     #177777,@R0     ;SET ALL BITS IN RKBA
003726  005010                          CLR     @R0             ;CLEAR RKBA
003730  005710                          TST     @R0             ;WAS IT CLEARED?
003732  001405                          BEQ     1$              ;YES, BRANCH
003734  010037  001162                  MOV     R0,$REG0        ;GET ADRES OF RKBA
003740  011037  001164                  MOV     @R0,$REG1       ;NO, GET RKBA
003744  104010                          ERROR   10              ;RKBA COULD NOT BE CLEARED

003746  005001                  1$:     CLR     R1              ;INITIALIZE COUNT PATTERN
003750  012705  177773                  MOV     #-5,R5
003754  012737  003762  001110          MOV     #2$,$LPERR
003762  010110                  2$:     MOV     R1,@R0          ;WRITE THE PATTERN IN THE
                                                                ;REGISTER
```

```
003764  011002                          MOV     @R0,R2
003766  020102                          CMP     R1,R2           ;WAS IT WRIITEN CORRRECTLY?
003770  001407                          BEQ     3$              ;YES, BRANCH
003772  010137  001162                  MOV     R1,$REG0        ;GET EXPECTED WORD
003776  011037  001164                  MOV     @R0,$REG1       ;GET WORD THAT WAS RECVD
004002  104013                          ERROR   13              ;DID NOT READ BACK THE PATTERN THAT
                                                                ;WAS WRITTEN INTO THE REGISTER
004004  005205                          INC     R5
004006  001402                          BEQ     TST16           ;;EXIT
004010  005201                  3$:     INC     R1              ;INCREMENT COUNT PATTERN
004012  001363                          BNE     2$              ;LUP BAK & WRITE NXT PATTERN IF NOT
                                                                ;DONE WITH ALL
                                ;
                                ;;*****************************************************************
                                ;*TEST 16        CHECK THAT RKDA CAN BE SET
                                        ;*THIS TEST FLOATS A '1' THROUGH RKDA AND CHECKS THAT THE WORD WHICH WAS
                                        ;*WRITTEN IS READ BACK CORRECTLY
                                ;;*****************************************************************
004014  000004                  TST16:  SCOPE
004016  005077  175226                  CLR     @RKCS           ;CLEAR RKCS
004022  012700  000001                  MOV     #1,R0           ;INITIALIZE R0 FOR BIT TO BE WRITTEN IN RKDA
004026  013701  001256                  MOV     RKDA,R1
004032  012737  004040  001110          MOV     #1$,$LPERR      ;SET UP RETURN ADRES
                                                                ;FOR LUPING ON EROR
004040  010011                  1$:     MOV     R0,@R1          ;WRITE THAT BIT IN RKDA
004042  011102                          MOV     @R1,R2
004044  020002                          CMP     R0,R2           ;WAS IT WRITTEN CORRECTLY
004046  001405                          BEQ     2$              ;YES, BRANCH
004050  010037  001162                  MOV     R0,$REG0        ;NO, GET EXPCTD RKDA (BIT WRITTEN)
004054  010237  001164                  MOV     R2,$REG1        ;GET RKDA (BIT READ BACK)
004060  104015                          ERROR   15              ;DID NOT READ BACK THE BIT THAT
                                                                ;WAS WRITTEN IN RKDA
004062  006300                  2$:     ASL     R0              ;SHIFT R0, TOWRITE NXT BIT
004064  001365                          BNE     1$              ;IF ALL BITS ARE NOT CHKD, LOOP
                                                                ;BACK & WRITE NXT BIT
                                ;
                                ;;*****************************************************************
                                ;*TEST 17        CHECK RKDA WITH A COUNT PATTERN
                                        ;*THIS TEST CHECKS THAT RKDA CAN BE CLEARED, THEN A COUNT PATTERN
                                        ;*FROM 0 TO 177777 IS WRITTEN AND CHECKED. R1 CONTAINS THE COUNT
                                        ;*PATTERN TO BE WRITTEN
                                ;;*****************************************************************
004066  000004                  TST17:  SCOPE
004070  012737  000010  001206          MOV     #10,$TIMES      ;;DO 10 ITERATIONS
004076  013700  001256                  MOV     RKDA,R0
004102  012710  177777                  MOV     #177777,@R0     ;SET ALL BITS IN RKDA
004106  005010                          CLR     @R0             ;CLEAR RKDA
004110  005710                          TST     @R0             ;WAS IT CLEARED?
004112  001405                          BEQ     1$              ;YES, BRANCH
004114  010037  001162                  MOV     R0,$REG0        ;GET ADRES OF RKDA
004120  011037  001164                  MOV     @R0,$REG1       ;GO, GET RKDA
004124  104010                          ERROR   10              ;RKDA COULD NOT BE CLEARED

004126  005001                  1$:     CLR     R1              ;INITIALIZE COUNT PATTERN
004130  012705  177773                  MOV     #-5,R5
004134  012737  004142  001110          MOV     #2$,$LPERR
```

```
                                        T17     CHECK RKDA WITH A COUNT PATTERN

004142  010110                          2$:     MOV     R1,@R0          ;WRITE THE PATTERN IN THE REGISTER
004144  011002                                  MOV     @R0,R2
004146  020102                                  CMP     R1,R2           ;WAS IT WRITTEN CORRRECTLY?
004150  001407                                  BEQ     3$              ;YES, BRANCH
004152  010137  001162                          MOV     R1,$REG0        ;GET EXPECTED WORD
004156  010237  001164                          MOV     R2,$REG1        ;GET WORD THAT WAS RECVD
004162  104015                                  ERROR   15              ;DID NOT READ BACK THE PATTERN THAT
                                                                        ;WAS WRITTEN INTO THE REGISTER
004164  005205                                  INC     R5
004166  001402                                  BEQ     TST20           ;;EXIT
004170  005201                          3$:     INC     R1              ;INCREMENT COUNT PATTERN
004172  001363                                  BNE     2$              ;LUP BAK & WRITE NXT PATTERN IF NOT
                                                                        ;DONE WITH ALL

                                        ;
                                        ;;*****************************************************************
                                        ;*TEST 20        CHECK THAT RKWC,RKBA,RKDA CAN BE CLEARED BY RESET
                                                ;*THIS TEST CHECKS THAT RKWC, RKBA AND RKDA CAN BE CLEARED BY BUS INIT.
                                                ;*RESET INSTRUCTION IS USED
                                        ;;*****************************************************************
004174  000004                          TST20:  SCOPE
004176  013700  001252                          MOV     RKWC,R0         ;INITIALIZE R0 TO PRINT TO RKWC
004202  010002                                  MOV     R0,R2
004204  012701  177775                          MOV     #-3,R1          ;SET UP COUNT FOR 3 REGISTERS TO BE CHKD
004210  010103                                  MOV     R1,R3
004212  012720  177777                  1$:     MOV     #177777,(R0)+   ;SET ALL BITS IN RKWC
004216  005201                                  INC     R1              ;                  RKBA
004220  001374                                  BNE     1$              ;                  RKDA
004222  000005                                  RESET                   ;ISSUE A BUS INIT
004224  005712                          2$:     TST     (R2)            ;WAS THE REGISTER (PTD TO BY R2) CLEARED?
004226  001405                                  BEQ     3$              ;YES, BRANCH
004230  010237  001162                          MOV     R2,$REG0        ;NO, GET ADRES OF REG IS THAT WAS NOT CLEARED
004234  011237  001164                          MOV     @R2,$REG1       ;GET CONTENTS OF THAT REGISTER
004240  104017                                  ERROR   17              ;RK11 REGISTER (ADRES IN R0) COULD
                                                                        ;NOT BE CLEARED BY BUS INIT
004242  005722                          3$:     TST     (R2)+           ;INCREMENT POINTER TO NXT REGISTER
004244  005203                                  INC     R3              ;HAVE U CHKD ALL 3 REGISTERS?
004246  001366                                  BNE     2$              ;IF NOT, LOOP BACK & CHK NXT

                                        ;;*****************************************************************
                                        ;*TEST 21        CHECKTHAT RKCS, RKWC, RKBA, RKDA CAN BE CLEARED BY CONTROL RESET
                                                ;*RKCS IS SET TO 7560, RKWC, RKBA, RKDA ARE ALL SET
                                                ;*TO 177777.  CONTROL RESET IS DONE AND IT IS CHECKED
                                                ;*IF ALL THESE REGISTERS ARE CLEARED.
                                        ;;*****************************************************************
004250  000004                          TST21:  SCOPE
004252  012746  000340                          MOV     #340,-(SP)
004256  012746  004264                          MOV     #64$,-(SP)
004262  000002                                  RTI
004264                                  64$:
004264  013700  001250                          MOV     RKCS,R0
004270  012710  007560                          MOV     #7560,@R0       ;SET ALL WRITABLE BITS IN RKCS
004274  005210                                  INC     @R0             ;SET GO, CONTROL RESET
004276  005005                                  CLR     R5
004300  105710                          1$:     TSTB    @R0             ;DID CNTRL RDY SET?
004302  100405                                  BMI     2$              ;YES, BRANCH
004304  005205                                  INC     R5              ;WAITED LONG?
```

```
004306  001374                          BNE     1$                  ;IF NOT LUP BAK & WAIT
004310  004737  005434                  JSR     PC,GT3RG            ;GET RKCS,ER,DS
004314  104007                          ERROR   7                   ;CNTRL RDY DID NOT SET
                                                                    ;AFTER CNTRL RESET
004316  022710  000200          2$:     CMP     #200,@R0            ;DID CNTRL RESET CLEAR RKCS
004322  001405                          BEQ     3$                  ;YES, BRANCH
004324  010037  001162                  MOV     R0,$REG0            ;GET ADRES OF RKCS
004330  011037  001164                  MOV     @R0,$REG1           ;GET CONTENTS OF RKCS
004334  104014                          ERROR   14                  ;CONTROL RESET DID NOT CLEAR RKCS
004336  013702  001252          3$:     MOV     RKWC,R2
004342  010204                          MOV     R2,R4
004344  012701  177777                  MOV     #177777,R1
004350  010122                          MOV     R1,(R2)+            ;SET ALL BITS IN RKWC
004352  010122                          MOV     R1,(R2)+            ;                  RKBA
004354  010122                          MOV     R1,(R2)+            ;                  RKDA
004356  104412                          CNT.RESET                   ;GO, DO CONTROL RESET
                                                                    ;THIS IS A CALL FOR THE 'CNTRL-
                                                                    ;RESET' ROUTINE. A CONTROL RESET IS
                                                                    ;DONE & AFTER A CERTAIN TIME IF
                                                                    ;'CNTRL RDY' DOES NOT SET AN ERROR IS
                                                                    ;REPORTED. NOTE THAT THE PC IN ERROR
                                                                    ;IS THE PC WHERE CNT.RESET IS
                                                                    ;LOCATED. THIS IS A VERY BASIC ERROR &
                                                                    ;IF IT OCCURS GO BACK TO TEST 10.
004360  012703  177775                  MOV     #-3,R3
004364  005714                  4$:     TST     (R4)                ;WAS THE REGISTER CLEARED?
004366  001405                          BEQ     5$                  ;YES, BRANCH
004370  010437  001162                  MOV     R4,$REG0            ;GET ADRES OF REGISTER IN ERROR
004374  011437  001164                  MOV     @R4,$REG1           ;GET CONTENTS OF THAT REGISTER
004400  104014                          ERROR   14                  ;CONTROL RESET DID NOT CLEAR THE
                                                                    ;REGISTER WHOOSE ADRES IS IN R4
004402  005724                  5$:     TST     (R4)+               ;INCREMENT POINTER TO NXT REGISTER
004404  005203                          INC     R3                  ;CHKD ALL REGS?
004406  001366                          BNE     4$                  ;IF NOT, LUP BAK & CHK THE NXT REG

                                ;;*******************************************************************
                                ;*TEST 22        CHECK THAT EACH RK11 REGISTER IS UNIQUELY ADRESSED
                                        ;*THIS TEST CHECKS THAT EACH RK11 REGISTER CAN BE UNIQUELY ADDRESSED
                                        ;*1) RKCS RKWC, RKBA, RKDA ARE FIRST SET TO 177777 (17576 FOR RKCS)
                                        ;*2) REGISTER WHOOSE ADDRESS IS IN R0 IS CLEARED
                                        ;*3) EVERY OTHER REGISTER IS CHECKED FOR ERRONEOUS CLEARING BECAUSE OF
                                        ;*ADDRESSING ERROR. IF SO THE MULTIPLE ADRESSING ERROR IS REPORTED
                                        ;*4) ADDRESS IN R0 IS CHANGED TO THE NEXT REGISTER & THE PROCESS IS
                                        ;*   REPEATED.
                                ;;*******************************************************************
004410  000004                  TST22:  SCOPE
004412  012746  000340                  MOV     #340,-(SP)
004416  012746  004424                  MOV     #64$,-(SP)
004422  000002                          RTI
004424                          64$:
004424  012705  177771                  MOV     #-7,R5              ;SET UP COUNT FOR THE # OF
                                                                    ;REGISTERS TO BE UNIQELY ADDRSD
004430  013700  001244                  MOV     RKDS,R0             ;INITIALIZE POINTER TO REGIS TO BE SET
004434  012701  177774          1$:     MOV     #-4,R1
004440  013702  001250                  MOV     RKCS,R2
004444  012722  017576                  MOV     #17576,(R2)+        ;SET BITS IN RKCS
```

```
004450  012722  177777          2$:     MOV     #177777,(R2)+   ;SET BITS IN RKWC
004454  005201                          INC     R1              ;             RKBA
004456  001374                          BNE     2$              ;             RKDA
                                                                ;       RKMR IF RK11C
004460  005010                          CLR     @R0             ;CLEAR REGISTER (WHOSE ADDR IS IN R0)
004462  013702  001250                  MOV     RKCS,R2
004466  020002                          CMP     R0,R2           ;WAS THE CLEARED REGISTER RKCS?
004470  001406                          BEQ     3$              ;YES
                                                                ;NO-CHECK IF IT WAS INADVERTENTLY
004472  011203                          MOV     @R2,R3          ;CLEARED BECAUSE OF ADDRESSING
004474  042703  170200                  BIC     #170200,R3      ;ERROR
004500  022703  007576                  CMP     #7576,R3
004504  001020                          BNE     6$              ;IF SO, REPORT ERROR
004506  005722                  3$:     TST     (R2)+           ;INCREMENT POINTER TO NEXT REGISTER
004510  012701  177775                  MOV     #-3,R1          ;SET COUNT FOR 3 REGISTERS
004514  020200                  4$:     CMP     R2,R0           ;WAS THE CLEARED REGISTER SAME AS
                                                                ;THE REGISTER POINTED TO BY R2
004516  001404                          BEQ     5$              ;YES
                                                                ;NO, CHECK IF THE REGIS UNDER TEST
                                                                ;(POINTED BY R2) WAS INADVERTENTLY
                                                                ;CLEARED WHEN CLEARING THE  REGIS
                                                                ;POINTED TO BY R0, DUE TO
                                                                ;ADDRESSING ERROR
004520  011203                          MOV     @R2,R3          ;GET CONTENTS OF REGIS BEING CHECKED
004522  010304                          MOV     R3,R4           ;FOR INADVERTENT CLEARING
004524  005104                          COM     R4              ;CHECK IF ANY BIT WAS ERRORNEOUSLY CLEARED
004526  001007                          BNE     6$              ;IF SO, REPORT ERROR
004530  005722                  5$:     TST     (R2)+           ;INCRMENT PTR TO NEXT REGISTER
004532  005201                          INC     R1              ;INCRMENT COUNT
004534  001367                          BNE     4$              ;CHECK THE REST
                                                                ;
004536  005720                          TST     (R0)+           ;INCREMENT PTR TO THE NXT REGIS TO  BE CLEARED
004540  005205                          INC     R5              ;HAVE ALL THE REGIS BEEN CHECKED?
004542  001334                          BNE     1$              ;IF NOT, LOOP BACK
004544  000415                          BR      TST23           ;;EXIT, IF DONE
004546  010037  001162          6$:     MOV     R0,$REG0        ;GET ADRES OF REGISTER THAT WAS
                                                                ;TRIED TO REFERENCE
004552  010237  001164                  MOV     R2,$REG1        ;GET ADRES OF REGISTER THAT GOT
                                                                ;REFERENCED INSTEAD
004556  011037  001166                  MOV     @R0,$REG2       ;GET CONTENTS OF REG THAT WAS
                                                                ;ADDRESED & MEANT TO BE CLEARED
004562  010337  001170                  MOV     R3,$REG3        ;GET CONTENTS OF REGISTER THAT GOT
                                                                ;CHANGED INSTEAD
004566  104020                          ERROR   20              ;POSSIBLE ADRESING ERROR. TRIED
                                                                ;ADRESING RK11 REGISTER, ANOTHER ONE
                                                                ;GOT ADRESED. REGIS IN R0 WAS THE
                                                                ;ADRESED ONE, REG IN R2
                                                                ;WAS THE ONE THAT GOT ADRESED INSTEAD
004570  023702  001250                  CMP     RKCS,R2
004574  001744                          BEQ     3$              ;RETURN TO THE
004576  000754                          BR      5$              ;RIGHT POINT
                                ;;*******************************************************************
                                ;*TEST 23        CHECK THAT HI & LO BYTES OF RKCS CAN BE ADDRESSED
                                        ;*THIS TEST CHECKS THAT THE HI & LO BYTES OF RKCS CAN BE
                                        ;*ADDRESSED CORRECTLY.
                                        ;*BITS IN ALL REGISTERS THAT CAN BE WRITTEN ARE SET. THEN EACH
```

```
                                        ;*BYTE OF RKCS IS REFERENCED BY 'CLRB' & IT IS CHECKED THAT ONLY
                                        ;*THAT BYTE & NO OTHER REGISTER BYTE GETS CLEARED
                                        ;*****************************************************************
004600  000004                          TST23:  SCOPE
004602  012702  177776                          MOV     #-2,R2          ;SET COUNT FOR 2BYTES-HI & LO
004606  012700  177775                          MOV     #-3,R0          ;SET COUNT
004612  012746  000340                          MOV     #340,-(SP)
004616  012746  004624                          MOV     #64$,-(SP)
004622  000002                                  RTI
004624                                  64$:
004624  013701  001250                          MOV     RKCS,R1         ;INITIALIZE PTR. TO RKCS
004630  012721  007576                          MOV     #7576,(R1)+     ;SET ALL BITS IN RKCS
004634  012721  177777                  1$:     MOV     #177777,(R1)+   ;SET ALL BITS IN RKWC
004640  005200                                  INC     R0              ;                  RKBA
004642  001374                                  BNE     1$              ;                  RKDA
004644  013701  001250                          MOV     RKCS,R1         ;INITIALIZE PTR TO RKCS LO BYTE
004650  105011                          2$:     CLRB    @R1             ;CLER RKCS LO OR HI BYTE
004652  010104                                  MOV     R1,R4
004654  017703  174370                          MOV     @RKCS,R3        ;GET RKCS WORD
004660  022702  177776                          CMP     #-2,R2          ;R U CHKING HI OR LO BYTE?
004664  001011                                  BNE     3$              ;BRANCH IF HI BYTE
004666  042703  177600                          BIC     #177600,R3      ;MASK HI BYTE
004672  001417                                  BEQ     4$              ;OK IF LO BYTE WAS CLEARED
004674  005037  001162                          CLR     $REG0           ;GET EXPCTD RKCS.
004700  010337  001164                          MOV     R3,$REG1        ;GET RKCS RECVD, LO BYTE
004704  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.
004706  000411                                  BR      4$
004710  042703  000377                  3$:     BIC     #377,R3         ;MASK LO BYTE
004714  001406                                  BEQ     4$              ;OK IF HI BYTE WAS CLEARED
004716  005037  001162                          CLR     $REG0           ;GET EXPCTD RKCS, HI BYTE
                                                                        ;GET WHAT WAS ACTUALLY RECVD
004722  000303                                  SWAB    R3
004724  010337  001164                          MOV     R3,$REG1
004730  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.
004732  012700  177774                  4$:     MOV     #-4,R0          ;INITIALIZE COUNT FOR REST OF REGISTERS
004736  013705  001250                          MOV     RKCS,R5         ;INITIALIZE POINTER TO RKCS
004742  005725                          5$:     TST     (R5)+           ;INCREMENT PTR TO NXT REGIS
004744  005200                                  INC     R0              ;ALL REGS DONE?
004746  001417                                  BEQ     6$              ;IF YES, GO & ADRES TO CLEAR RKCS HI BYTE
004750  011503                                  MOV     @R5,R3          ;IF NOT, GET CONTENTS OF THE REGIS
                                                                        ;BEING CHKD
004752  005103                                  COM     R3              ;COMPLEMENT THE CONTENTS, SHOULD BE
                                                                        ;0 FOR IT WAS
004754  001772                                  BEQ     5$              ;PREVIOUSLY SET TO ALL 1'S IF IT'S
                                                                        ;0 BRANCH, IF NOT REPORT ERROR
004756  010137  001162                          MOV     R1,$REG0        ;GET RKCS-BYTE-ADRES WHICH WAS TRIED
                                                                        ;TO ADRES
004762  010537  001164                          MOV     R5,$REG1        ;GET ADRES OF REGIS WHICH GOT ADRESED
                                                                        ;INSTEAD
004766  012737  177777  001166                  MOV     #177777,$REG2   ;GET EXPCTD CONTENTS OF REGISTER
                                                                        ;THAT GO ADRESED
004774  005103                                  COM     R3              ;GET CONTENTS RECVD FROM THAT REGIS
```

```
004776  010337  001170                    MOV     R3,$REG3
005002  104024                            ERROR   24              ;ALL RK11 REGISTERS WERE LOADED
                                                                  ;WITH 1'S.  RKCS
                                                                  ;BYTE (ADRES UNDER 'BYTE' IN ER
                                                                  ;MSGE) WAS ADRESED
                                                                  ;USING 'CLRB', BUT REGISTER (ADRES
                                                                  ;UNDER 'REGIS' IN
                                                                  ;ER MSG) GOT CHANGED AS A RESULT.
005004  000756                            BR      5$
005006  005201                    6$:     INC     R1              ;POSITION PTR TO RKCS HI BYTE
005010  005202                            INC     R2              ;CHK IF BOTH HI & LO BYTES (RKCS)
                                                                  ;WERE CLEARED
005012  001316                            BNE     2$              ;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.
                                    ;;*****************************************************************
005014  000004                    TST24:  SCOPE
005016  012746  000340                    MOV     #340,-(SP)
005022  012746  005030                    MOV     #64$,-(SP)
005026  000002                            RTI
005030                            64$:
005030  012704  177772                    MOV     #-6,R4          ;SET UP COUNT FOR 6 REG-BYTES TO BE ADRESED
005034  013702  001252                    MOV     RKWC,R2         ;INITIALIZE POINTER TO RKWC
005040  012700  177775            1$:     MOV     #-3,R0
005044  013701  001250                    MOV     RKCS,R1
005050  012721  007576                    MOV     #7576,(R1)+     ;SET RKCS BITS
005054  012721  177777            2$:     MOV     #177777,(R1)+   ;     RKWC
005060  005200                            INC     R0              ;        RKBA
005062  001374                            BNE     2$              ;        RKDA

005064  105012                            CLRB    @R2             ;ADDRESS & CLEAR REGIS BYTE UNDER TEST

005066  010200                            MOV     R2,R0
005070  042700  000001                    BIC     #1,R0           ;CONVERT THE BYTE ADRES INTO WORD
                                                                  ;ADRES THAT IT BELONGS TO
005074  011001                            MOV     @R0,R1          ;GET THE ENTIRE REGIS WRD
005076  032702  000001                    BIT     #1,R2           ;WAS THE CLRD BYTE HI OR LO?
005102  001003                            BNE     3$              ;WAS HI, BRANCH
005104  042701  177400                    BIC     #177400,R1      ;WAS LO-MASK HI BYTE
005110  000402                            BR      4$
005112  042701  000377            3$:     BIC     #377,R1         ;MASK LO BYTE
005116  001411                    4$:     BEQ     5$              ;WAS THE ADDRESSED BYTE CLEARED? BR IF YES
005120  010237  001162                    MOV     R2,$REG0        ;GET ADRES OF REG-BYTE THAT WAS
                                                                  ;TRIED TO B ADRESED & CLEARED
005124  032702  000001                    BIT     #1,R2
005130  001401                            BEQ     11$
005132  000301                            SWAB    R1
005134  010137  001164            11$:    MOV     R1,$REG1        ;GET CONTENTS OF REG-BYTE
005140  104025                            ERROR   25              ;TRIED TO ADRES & CLR A REGISTER BYTE
```

```
                                                                ;(ADRES UNDER 'REG-BYTE' IN ER MSGE), COULD
                                                                ;NOT CLEAR IT.
005142  017701  174102          5$:     MOV     @RKCS,R1        ;
005146  022701  007776                  CMP     #7776,R1        ;WAS RKCS ERRONEOUSLY CLRD?
005152  001410                          BEQ     6$              ;NO, BRANCH
005154  010237  001162                  MOV     R2,$REG0        ;GET ADRES OF REG-BYTE THAT WAS
                                                                ;TRIED TO B ADRESED & CLEARED.
005160  012737  007776  001164          MOV     #7776,$REG1     ;GET EXPCTD RKCS
005166  010137  001166                  MOV     R1,$REG2        ;GET RKCS RECVD
005172  104016                          ERROR   16              ;ALL RK11 REGISTRS WERE LOADED WITH 1'S
                                                                ;TRIED TO ADRES & CLR 'REGIS-BYTE' (IN ER MSGE)
                                                                ;RKCS GOT CHANGED AS A RESULT.  '(RKCS)EXP'
                                                                ;CONTAINS EXPCTD RKCS.  'RKCS (RECVD)' CONTAINS
                                                                ;RKCS RECVD.
005174  012700  177772          6$:     MOV     #-6,R0          ;SET COUNT FOR BYTES
005200  013701  001252                  MOV     RKWC,R1         ;INITIALIZE PTR TO RKWC
005204  020102                  7$:     CMP     R1,R2           ;WAS THE CLEARED BYTE (PTD TO BY R2)
                                                                ;SAME AS BYTE TO BE CHKD (PTD TO BY R1)?
005206  001433                          BEQ     10$             ;IF YES, DO NOT CHK THIS BYTE
005210  010105                          MOV     R1,R5
005212  042705  000001                  BIC     #1,R5           ;STRIP WORD ADDRESS FROM BYTE ADDRS
005216  011503                          MOV     @R5,R3
005220  032701  000001                  BIT     #1,R1           ;IS THE BYTE TO B CHKD LO OR HI BYTE?
005224  001003                          BNE     8$              ;HI BYTE-BRANCH
005226  042703  177400                  BIC     #177400,R3      ;MASK HI BYTE
005232  000402                          BR      9$
005234  042703  000377          8$:     BIC     #377,R3         ;MASK LO BYTE
005240  001016                  9$:     BNE     10$
005242  010237  001162                  MOV     R2,$REG0        ;GET ADRES OF REG-BYTE THAT WAS
                                                                ;TRIED TO B ADRESED & CLEARED
005246  010137  001164                  MOV     R1,$REG1        ;GET ADRS OF REG-BYTE THAT GOT
                                                                ;ADRESED INSTEAD
005252  012737  000377  001166          MOV     #377,$REG2      ;GET EXPCTD CONTENTS OF REG-BYTE
                                                                ;THAT GOT ADRESED
005260  032701  000001                  BIT     #1,R1
005264  001401                          BEQ     12$
005266  000303                          SWAB    R3
005270  010337  001170          12$:    MOV     R3,$REG3        ;GET CONTENTS RECVD FOR THAT REG-BYTE
005274  104021                          ERROR   21              ;ALL RK11 REGISTRS WERE LOADED WITH 1'S.
                                                                ;TRIED TO ADRES & CLR 'REG-BYT1' (IN ER MSGE)
                                                                ;'REG-BYT2' GOT CHANGD AS A RESULT.
                                                                ;'BYT2-EXPCT' (IN ER MSGE) IS THE EXPCTD CONTENTS
                                                                ;OF REG-BYT2.  'BYT2-RECVD' IS THE CONTENTS RECVD.
005276  005201                  10$:    INC     R1              ;INCREMENT PTR TO NXT REG BYTE
005300  005200                          INC     R0              ;ALL BYTES CHKD?
005302  001340                          BNE     7$              ;NO, LOOP BACK
005304  005202                          INC     R2              ;INCREMENT PTR TO NXT REG BYTE TO B CLRD
005306  005204                          INC     R4              ;ALL BYTES CLRD?
005310  001253                          BNE     1$              ;LOOP BACK


                                .SBTTL  END OF PASS ROUTINE

                                ;;*******************************************************************
```

```
                                        ;*INCREMENT THE PASS NUMBER ($PASS)
                                        ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
                                        ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
                                        ;*IF THERES A MONITOR GO TO IT
                                        ;*IF THERE ISN'T JUMP TO START1

005312                                  $EOP:
005312  000004                                  SCOPE
005314  005037  001102                          CLR     $TSTNM          ;;ZERO THE TEST NUMBER
005320  005037  001206                          CLR     $TIMES          ;;ZERO THE NUMBER OF ITERATIONS
005324  005237  001100                          INC     $PASS           ;;INCREMENT THE PASS NUMBER
005330  042737  100000  001100                  BIC     #100000,$PASS   ;;DON'T ALLOW A NEG. NUMBER
005336  005327                                  DEC     (PC)+           ;;LOOP?
005340  000001                          $EOPCT: .WORD   1
005342  003022                                  BGT     $DOAGN          ;;YES
005344  012737                                  MOV     (PC)+,@(PC)+    ;;RESTORE COUNTER
005346  000001                          $ENDCT: .WORD   1
005350  005340                                  $EOPCT
005352  104401  005417                          TYPE    ,$ENDMG         ;;TYPE "END PASS #"
005356  013746  001100                          MOV     $PASS,-(SP)     ;;SAVE $PASS FOR TYPEOUT
005362  104405                                  TYPDS                   ;;GO TYPE--DECIMAL ASCII WITH SIGN
005364  104401  005414                          TYPE    ,$ENULL         ;;TYPE A NULL CHARACTER
005370  013700  000042                  $GET42: MOV     @#42,R0         ;;GET MONITOR ADDRESS
005374  001405                                  BEQ     $DOAGN          ;;BRANCH IF NO MONITOR
005376  000005                                  RESET                   ;;CLEAR THE WORLD
005400  004710                          $ENDAD: JSR     PC,(R0)         ;;GO TO MONITOR
005402  000240                                  NOP                     ;;SAVE ROOM
005404  000240                                  NOP                     ;;FOR
005406  000240                                  NOP                     ;;ACT11
005410                                  $DOAGN:
005410  000137                                  JMP     @(PC)+          ;;RETURN
005412  002110                          $RTNAD: .WORD   START1
005414     377     377     000          $ENULL: .BYTE   -1,-1,0         ;;NULL CHARACTER STRING
005417     015  042412  042116          $ENDMG: .ASCIZ  <15><12>/END PASS #/
005424  050040  051501  020123
005432  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

005434  017737  173610  001162  GT3RG:  MOV     @RKCS,$REG0     :GET RKCS
005442  017737  173600  001164          MOV     @RKER,$REG1     :GET RKER
005450  017737  173570  001166          MOV     @RKDS,$REG2     :GET RKDS
005456  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, $REG3 RESPECTIVELY BEFORE
                                ;TYPING OUT AN ERROR MESSAGE.
                                ;CALL:  JSR     PC,GT4RG
005460  004737  005434          GT4RG:  JSR     PC,GT3RG
005464  017737  173566  001170          MOV     @RKDA,$REG3
005472  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.

005474  017637  000000  001264  DELA.Y: MOV     @(SP),TIMER     ;GET 'AMOUNT' (N) FOR WHICH
005502  062716  000002                  ADD     #2,(SP)         ;DELAY IS TO BE PROVIDED
                                                                ;ADJUST STACK POINTER TO SKIP OVER 'N'
005506  005337  001264          1$:     DEC     TIMER           ;COUNT DOWN TO 0
005512  001375                          BNE     1$
005514  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
                                        ;SETS EXITS OUT. IF WITHIN A CERTAIN TIME CNTRL RDY DOES
                                        ;NOT SET AN ERROR IS REPORTED. WAITING TIME IS 883 MS FOR 11/20
                                        ;175 MS FOR 11/45 WITH BIPOLAR MEMORY.
                                        ;CALL:  CNT.RDY
005516  012777  000001  173524  CN.RST: MOV     #1,@RKCS        ;ISSUE A CONTROL RESET
005524  012737  177500  001170          MOV     #-300,$REG3     ;SET UP COUNT
005532  000402                          BR      CN.RDY+4                ;SKIP OVER CN.RDY
005534  005037  001170          CN.RDY: CLR     $REG3
005540  105777  173504          1$:     TSTB    @RKCS           ;DID CNTRL-RDY SET?
005544  100435                          BMI     3$              ;YES, EXIT
005546  005237  001170                  INC     $REG3           ;WAITED LONG?
005552  001372                          BNE     1$              ;IF NOT, GO BAK & WAIT
005554  032777  020000  173356  2$:     BIT     #SW13,@SWR      ;INHIBIT TYPEOUT?
005562  001026                          BNE     3$              ;IF YES, SKIP TYPEOUT
005564  104401                          TYPE
005566  001216                          MSG3
005570  104401  005576                  TYPE    ,65$            ;;TYPE ASCIZ STRING
005574  000403                          BR      64$             ;;GET OVER THE ASCIZ
                                        ;;65$:  .ASCIZ  <15><12>/PC=/
005604                                  64$:
005604  011646                          MOV     (SP),-(SP)
005606  162716  000002                  SUB     #2,(SP)
005612  104402                          TYPOC                   ;GO TYPE PC IN THE MAIN PROGRAM.
                                                                ; WHERE ERROR OCCURRED
005614  104401  005622                  TYPE    ,67$            ;;TYPE ASCIZ STRING
005620  000404                          BR      66$             ;;GET OVER THE ASCIZ
                                        ;;67$:  .ASCIZ  / RKCS=/
005632                                  66$:
005632  017746  173412                  MOV     @RKCS,-(SP)     ;GET RKCS
005636  104402                          TYPOC                   ;GO TYPE IT

005640  000002                  3$:     RTI                     ;RETURN FROM THIS
                                                                ;ROUTINE TO THE MAIN
                                                                ;PROGRAM


                                        ;THIS PART OF THE PROGRAM CONTAINS THE COMMON ROUTINES CALLED
                                        ;FROM THE SYSMAC.SML PACKAGE
                                        ;

                                        .SBTTL  SCOPE HANDLER ROUTINE

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

```
005642                                  $SCOPE:
005642  104407                                  CKSWR                           ;;TEST  FOR CHANGE IN SOFT-SWR
005644  032777  040000  173266          1$:     BIT     #BIT14,@SWR             ;;LOOP ON PRESENT TEST?
005652  001111                                  BNE     $OVER                   ;;YES IF SW14=1
                                        ;*****START OF CODE FOR THE XOR TESTER*****
005654  000416                          $XTSTR: BR      6$                      ;;IF RUNNING ON THE "XOR" TESTER CHANGE
                                                                                ;;THIS INSTRUCTION TO A "NOP" (NOP=240)
005656  013746  000004                          MOV     @#ERRVEC,-(SP)          ;;SAVE THE CONTENTS OF THE ERROR VECTOR
005662  012737  005702  000004                  MOV     #5$,@#ERRVEC            ;;SET FOR TIMEOUT
005670  005737  177060                          TST     @#177060                ;;TIME OUT ON XOR?
005674  012637  000004                          MOV     (SP)+,@#ERRVEC          ;;RESTORE THE ERROR VECTOR
005700  000463                                  BR      $SVLAD                  ;;GO TO THE NEXT TEST
005702  022626                          5$:     CMP     (SP)+,(SP)+             ;;CLEAR THE STACK AFTER A TIME OUT
005704  012637  000004                          MOV     (SP)+,@#ERRVEC          ;;RESTORE THE ERROR VECTOR
005710  000423                                  BR      7$                      ;;LOOP ON THE PRESENT TEST
005712                                  6$:;*****END OF CODE FOR THE XOR TESTER*****
005712  032777  000400  173220                  BIT     #BIT08,@SWR             ;;LOOP ON SPEC. TEST?
005720  001404                                  BEQ     2$                      ;;BR IF NO
005722  127737  173212  001102                  CMPB    @SWR,$TSTNM             ;;ON THE RIGHT TEST?    SWR<7:0>
005730  001462                                  BEQ     $OVER                   ;;BR IF YES
005732  105737  001103                  2$:     TSTB    $ERFLG                  ;;HAS AN ERROR OCCURRED?
005736  001421                                  BEQ     3$                      ;;BR IF NO
005740  123737  001115  001103                  CMPB    $ERMAX,$ERFLG           ;;MAX. ERRORS FOR THIS TEST OCCURRED?
005746  101015                                  BHI     3$                      ;;BR IF NO
005750  032777  001000  173162                  BIT     #BIT09,@SWR             ;;LOOP ON ERROR?
005756  001404                                  BEQ     4$                      ;;BR IF NO
005760  013737  001110  001106          7$:     MOV     $LPERR,$LPADR           ;;SET LOOP ADDRESS TO LAST SCOPE
005766  000443                                  BR      $OVER
005770  105037  001103                  4$:     CLRB    $ERFLG                  ;;ZERO THE ERROR FLAG
005774  005037  001206                          CLR     $TIMES                  ;;CLEAR THE NUMBER OF ITERATIONS TO MAKE
006000  000415                                  BR      1$                      ;;ESCAPE TO THE NEXT TEST
006002  032777  004000  173130          3$:     BIT     #BIT11,@SWR             ;;INHIBIT ITERATIONS?
006010  001011                                  BNE     1$                      ;;BR IF YES
006012  005737  001100                          TST     $PASS                   ;;IF FIRST PASS OF PROGRAM
006016  001406                                  BEQ     1$                      ;;      INHIBIT ITERATIONS
006020  005237  001104                          INC     $ICNT                   ;;INCREMENT ITERATION COUNT
006024  023737  001206  001104                  CMP     $TIMES,$ICNT            ;;CHECK THE NUMBER OF ITERATIONS MADE
006032  002021                                  BGE     $OVER                   ;;BR IF MORE ITERATION REQUIRED
006034  012737  000001  001104          1$:     MOV     #1,$ICNT                ;;REINITIALIZE THE ITERATION COUNTER
006042  013737  006112  001206                  MOV     $MXCNT,$TIMES           ;;SET NUMBER OF ITERATIONS TO DO
006050  105237  001102                  $SVLAD: INCB    $TSTNM                  ;;COUNT TEST NUMBERS
006054  011637  001106                          MOV     (SP),$LPADR             ;;SAVE SCOPE LOOP ADDRESS
006060  011637  001110                          MOV     (SP),$LPERR             ;;SAVE ERROR LOOP ADDRESS
006064  005037  001210                          CLR     $ESCAPE                 ;;CLEAR THE ESCAPE FROM ERROR ADDRESS
006070  112737  000001  001115                  MOVB    #1,$ERMAX               ;;ONLY ALLOW ONE(1) ERROR ON NEXT TEST
006076  013777  001102  173036          $OVER:  MOV     $TSTNM,@DISPLAY         ;;DISPLAY TEST NUMBER
006104  013716  001106                          MOV     $LPADR,(SP)             ;;FUDGE RETURN ADDRESS
006110  000002                                  RTI                             ;;FIXES PS
006112  000050                          $MXCNT: 50                              ;;MAX. NUMBER OF ITERATIONS


                                        ;;*******************************************************************

                                        .SBTTL  ERROR HANDLER ROUTINE

                                        ;*SW15=1        HALT ON ERROR
```

```
                                        ;*SW13=1          INHIBIT ERROR TYPEOUTS
                                        ;*SW10=1          BELL ON ERROR
                                        ;*SW09=1          LOOP ON ERROR
                                        ;*SW12=1          CYCLE ON ERROR TO PREVIOUS 'SCOPE'
                                        ;*GO TO $ERRTYP ON ERROR

006114  104407                          $ERROR: CKSWR                                   ;CHECK FOR SOFTWARE SWITCH ENTRY REQUEST
006116  105237  001103                  7$:     INCB    $ERFLG                          ;SET THE ERROR FLAG
006122  001775                                  BEQ     7$                              ;DON'T LET THE FLAG GO TO ZERO
006124  013777  001102  173010                  MOV     $TSTNM,@DISPLAY                 ;DISPLAY TEST NUMBER AND ERROR FLAG
006132  005237  001112                  1$:     INC     $ERTTL                          ;COUNT THE NUMBER OF ERRORS
006136  011637  001116                          MOV     (SP),$ERRPC                     ;GET ADDRESS OF ERROR INSTRUCTION
006142  162737  000002  001116                  SUB     #2,$ERRPC
006150  117737  172742  001114                  MOVB    @$ERRPC,$ITEMB                  ;STRIP AND SAVE THE ERROR ITEM CODE
006156  032777  020000  172754                  BIT     #SW13,@$SWR                     ;SKIP TYPEOUT IF SET
006164  001004                                  BNE     2$                              ;SKIP TYPEOUTS
006166  004737  006264                          JSR     PC,@#$ERRTYP                    ;GO TO USER ERROR ROUTINE
006172  104401  001213                          TYPE    ,$CRLF
006176  023737  000042  000046          2$:     CMP     @#42,@#46                       ;ARE WE IN ACT11 AUTOMATIC MODE?
006204  001403                                  BEQ     .+10                            ;IF YES, HALT ON ERROR
006206  005777  172726                          TST     @$SWR                           ;HALT ON ERROR
006212  100002                                  BPL     3$                              ;SKIP IF CONTINUE
006214  000000                                  HALT                                    ;HALT ON ERROR!
006216  104407                                  CKSWR                                   ;CHECK FOR SOFTWARE SWITCH ENTRY REQUEST
006220  032777  010000  172712          3$:     BIT     #SW12,@$SWR                     ;SW 12 SET?
006226  001402                                  BEQ     .+6                             ;NO, BRANCH
006230  013716  001106                          MOV     $LPADR,(SP)                     ;ADJUST RETURN ADRES FOR SW12
006234  032777  001000  172676                  BIT     #SW09,@$SWR                     ;LOOP ON ERROR SWITCH SET?
006242  001402                                  BEQ     4$                              ;BR IF NO
006244  013716  001110                          MOV     $LPERR,(SP)                     ;FUDGE RETURN FOR LOOPING
006250  005737  001210                  4$:     TST     $ESCAPE                         ;CHECK FOR AN ESCAPE ADDRESS
006254  001402                                  BEQ     5$                              ;BR IF NONE
006256  013716  001210                          MOV     $ESCAPE,(SP)                    ;FUDGE RETURN ADDRESS FOR ESCAPE
006262  000002                          5$:     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.

006264                                  $ERRTYP:
006264  104401  001213                          TYPE    ,$CRLF                  ;;"CARRIAGE RETURN" & "LINE FEED"
006270  010046                                  MOV     R0,-(SP)                ;;SAVE R0
006272  005000                                  CLR     R0                      ;;PICKUP THE ITEM INDEX
006274  153700  001114                          BISB    @#$ITEMB,R0
006300  001004                                  BNE     1$                      ;;IF ITEM NUMBER IS ZERO, JUST
                                                                                ;;TYPE THE PC OF THE ERROR
006302  013746  001116                          MOV     $ERRPC,-(SP)            ;;SAVE $ERRPC FOR TYPEOUT
                                                                                ;;ERROR ADDRESS
006306  104402                                  TYPOC                           ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
006310  000426                                  BR      6$                      ;;GET OUT
006312  005300                          1$:     DEC     R0                      ;;ADJUST THE INDEX SO THAT IT WILL
006314  006300                                  ASL     R0                      ;;      WORK FOR THE ERROR TABLE
006316  006300                                  ASL     R0
```

```
006320  006300                          ASL     R0
006322  062700  001272                  ADD     #$ERRTB,R0          ;;FORM TABLE POINTER
006326  012037  006336                  MOV     (R0)+,2$            ;;PICKUP "ERROR MESSAGE" POINTER
006332  001404                          BEQ     3$                  ;;SKIP TYPEOUT IF NO POINTER
006334  104401                          TYPE                        ;;TYPE THE "ERROR MESSAGE"
006336  000000                  2$:     .WORD   0                   ;;"ERROR MESSAGE" POINTER GOES HERE
006340  104401  001213                  TYPE    ,$CRLF              ;;"CARRIAGE RETURN" & "LINE FEED"
006344  012037  006354          3$:     MOV     (R0)+,4$            ;;PICKUP "DATA HEADER" POINTER
006350  001404                          BEQ     5$                  ;;SKIP TYPEOUT IF 0
006352  104401                          TYPE                        ;;TYPE THE "DATA HEADER"
006354  000000                  4$:     .WORD   0                   ;;"DATA HEADER" POINTER GOES HERE
006356  104401  001213                  TYPE    ,$CRLF              ;;"CARRIAGE RETURN" & "LINE FEED"
006362  011000                  5$:     MOV     (R0),R0             ;;PICKUP "DATA TABLE" POINTER
006364  001004                          BNE     7$                  ;;GO TYPE THE DATA
006366  012600                  6$:     MOV     (SP)+,R0            ;;RESTORE R0
006370  104401  001213                  TYPE    ,$CRLF              ;;"CARRIAGE RETURN" & "LINE FEED"
006374  000207                          RTS     PC                  ;;RETURN
006376                          7$:
006376  013046                          MOV     @(R0)+,-(SP)        ;;SAVE @(R0)+ FOR TYPEOUT
006400  104402                          TYPOC                       ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
006402  005710                          TST     (R0)                ;;IS THERE ANOTHER NUMBER?
006404  001770                          BEQ     6$                  ;;BR IF NO
006406  104401  006414                  TYPE    ,8$                 ;;TYPE TWO(2) SPACES
006412  000771                          BR      7$                  ;;LOOP
006414  020040     000          8$:     .ASCIZ  /  /                ;;TWO(2) SPACES
        006420                          .EVEN

                                .SBTTL  TTY INPUT ROUTINE

                                ;;*******************************************************************
                                .ENABL  LSB

                                ;;*******************************************************************
                                ;*SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
                                ;*ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
                                ;*SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
                                ;*WHEN OPERATING IN TTY FLAG MODE.
006420  022737  000176  001140  $CKSWR: CMP     #SWREG,SWR          ;;IS THE SOFT-SWR SELECTED?
006426  001074                          BNE     15$                 ;;BRANCH IF NO
006430  105777  172510                  TSTB    @$TKS               ;;CHAR THERE?
006434  100071                          BPL     15$                 ;;IF NO, DON'T WAIT AROUND
006436  117746  172504                  MOVB    @$TKB,-(SP)         ;;SAVE THE CHAR
006442  042716  177600                  BIC     #^C177,(SP)         ;;STRIP-OFF THE ASCII
006446  022726  000007                  CMP     #7,(SP)+            ;;IS IT A CONTROL G?
006452  001062                          BNE     15$                 ;;NO, RETURN TO USER
006454  123727  001134  000001          CMPB    $AUTOB,#1           ;;ARE WE RUNNING IN AUTO-MODE?
006462  001456                          BEQ     15$                 ;;BRANCH IF YES

006464  104401  007145                  TYPE    ,$CNTLG             ;;ECHO THE CONTROL-G (^G)
006470  104401  007152          $GTSWR: TYPE    ,$MSWR              ;;TYPE CURRENT CONTENTS
006474  013746  000176                  MOV     SWREG,-(SP)         ;;SAVE SWREG FOR TYPEOUT
006500  104402                          TYPOC                       ;;GO TYPE--OCTAL ASCII(ALL DIGITS)
006502  104401  007163                  TYPE    ,$MNEW              ;;PROMPT FOR NEW SWR
006506  005046                  19$:    CLR     -(SP)               ;;CLEAR COUNTER
006510  005046                          CLR     -(SP)               ;;THE NEW SWR
006512  105777  172426          7$:     TSTB    @$TKS               ;;CHAR THERE?
```

```
006516  100375                          BPL     7$              ;;IF NOT TRY AGAIN

006520  117746  172422                  MOVB    @$TKB,-(SP)     ;;PICK UP CHAR
006524  042716  177600                  BIC     #^C177,(SP)     ;;MAKE IT 7-BIT ASCII



006530  021627  000025          9$:     CMP     (SP),#25        ;;IS IT A CONTROL-U?
006534  001005                          BNE     10$             ;;BRANCH IF NOT
006536  104401  007140                  TYPE    ,$CNTLU         ;;YES, ECHO CONTROL-U (^U)
006542  062706  000006          20$:    ADD     #6,SP           ;;IGNORE PREVIOUS INPUT
006546  000757                          BR      19$             ;;LET'S TRY IT AGAIN


006550  021627  000015          10$:    CMP     (SP),#15        ;;IS IT A <CR>?
006554  001022                          BNE     16$             ;;BRANCH IF NO
006556  005766  000004                  TST     4(SP)           ;;YES, IS IT THE FIRST CHAR?
006562  001403                          BEQ     11$             ;;BRANCH IF YES
006564  016677  000002  172346          MOV     2(SP),@SWR      ;;SAVE NEW SWR
006572  062706  000006          11$:    ADD     #6,SP           ;;CLEAR UP STACK
006576  104401  001213          14$:    TYPE    ,$CRLF          ;;ECHO <CR> AND <LF>
006602  123727  001135  000001          CMPB    $INTAG,#1       ;;RE-ENABLE TTY KBD INTERRUPTS?
006610  001003                          BNE     15$             ;;BRANCH IF NOT
006612  012777  000100  172324          MOV     #100,@$TKS      ;;RE-ENABLE TTY KBD INTERRUPTS
006620  000002                  15$:    RTI                     ;;RETURN
006622  004737  007344          16$:    JSR     PC,$TYPEC       ;;ECHO CHAR
006626  021627  000060                  CMP     (SP),#60        ;;CHAR < 0?
006632  002420                          BLT     18$             ;;BRANCH IF YES
006634  021627  000067                  CMP     (SP),#67        ;;CHAR > 7?
006640  003015                          BGT     18$             ;;BRANCH IF YES
006642  042726  000060                  BIC     #60,(SP)+       ;;STRIP-OFF ASCII
006646  005766  000002                  TST     2(SP)           ;;IS THIS THE FIRST CHAR
006652  001403                          BEQ     17$             ;;BRANCH IF YES
006654  006316                          ASL     (SP)            ;;NO, SHIFT PRESENT
006656  006316                          ASL     (SP)            ;;    CHAR OVER TO MAKE
006660  006316                          ASL     (SP)            ;;    ROOM FOR NEW ONE.
006662  005266  000002          17$:    INC     2(SP)           ;;KEEP COUNT OF CHAR
006666  056616  177776                  BIS     -2(SP),(SP)     ;;SET IN NEW CHAR
006672  000707                          BR      7$              ;;GET THE NEXT ONE
006674  104401  001212          18$:    TYPE    ,$QUES          ;;TYPE ?<CR><LF>
006700  000720                          BR      20$             ;;SIMULATE CONTROL-U
                                .DSABL  LSB


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


006702  011646                  $RDCHR: MOV     (SP),-(SP)      ;;PUSH DOWN THE PC
006704  016666  000004  000002          MOV     4(SP),2(SP)     ;;SAVE THE PS
006712  105777  172226          1$:     TSTB    @$TKS           ;;WAIT FOR
006716  100375                          BPL     1$              ;;A CHARACTER
```

```
006720  117766  172222  000004          MOVB    @$TKB,4(SP)       ;;READ THE TTY
006726  042766  177600  000004          BIC     #^C<177>,4(SP)    ;;GET RID OF JUNK IF ANY
006734  026627  000004  000023          CMP     4(SP),#23         ;;IS IT A CONTROL-S?
006742  001013                          BNE     3$                ;;BRANCH IF NO
006744  105777  172174          2$:     TSTB    @$TKS             ;;WAIT FOR A CHARACTER
006750  100375                          BPL     2$                ;;LOOP UNTIL ITS THERE
006752  117746  172170                  MOVB    @$TKB,-(SP)       ;;GET CHARACTER
006756  042716  177600                  BIC     #^C177,(SP)       ;;MAKE IT 7-BIT ASCII
006762  022627  000021                  CMP     (SP)+,#21         ;;IS IT A CONTROL-Q?
006766  001366                          BNE     2$                ;;IF NOT DISCARD IT
006770  000750                          BR      1$                ;;YES, RESUME
006772  026627  000004  000140  3$:     CMP     4(SP),#140        ;;IS IT UPPER CASE?
007000  002407                          BLT     4$                ;;BRANCH IF YES
007002  026627  000004  000175          CMP     4(SP),#175        ;;IS IT A SPECIAL CHAR?
007010  003003                          BGT     4$                ;;BRANCH IF YES
007012  042766  000040  000004          BIC     #40,4(SP)         ;;MAKE IT UPPER CASE
007020  000002                  4$:     RTI                       ;;GO BACK TO USER
                                ;;*****************************************************************
                                ;*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
                                ;*CALL:
                                ;*      RDLIN                     ;;INPUT A STRING FROM THE TTY
                                ;*      RETURN HERE               ;;ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
                                ;*                                ;;TERMINATOR WILL BE A BYTE OF ALL 0'S

007022  010346                  $RDLIN: MOV     R3,-(SP)          ;;SAVE R3
007024  012703  007130          1$:     MOV     #$TTYIN,R3        ;;GET ADDRESS
007030  022703  007140          2$:     CMP     #$TTYIN+8.,R3     ;;BUFFER FULL?
007034  101405                          BLOS    4$                ;;BR IF YES
007036  104410                          RDCHR                     ;;GO READ ONE CHARACTER FROM THE TTY
007040  112613                          MOVB    (SP)+,(R3)        ;;GET CHARACTER
007042  122713  000177          10$:    CMPB    #177,(R3)         ;;IS IT A RUBOUT
007046  001003                          BNE     3$                ;;SKIP IF NOT
007050  104401  001212          4$:     TYPE    ,$QUES            ;;TYPE A '?'
007054  000763                          BR      1$                ;;CLEAR THE BUFFER AND LOOP
007056  111337  007126          3$:     MOVB    (R3),9$           ;;ECHO THE CHARACTER
007062  104401  007126                  TYPE    ,9$
007066  122723  000015                  CMPB    #15,(R3)+         ;;CHECK FOR RETURN
007072  001356                          BNE     2$                ;;LOOP IF NOT RETURN
007074  105063  177777                  CLRB    -1(R3)            ;;CLEAR RETURN (THE 15)
007100  104401  001214                  TYPE    ,$LF              ;;TYPE A LINE FEED
007104  012603                          MOV     (SP)+,R3          ;;RESTORE R3
007106  011646                          MOV     (SP),-(SP)        ;;ADJUST THE STACK AND PUT ADDRESS OF THE
007110  016666  000004  000002          MOV     4(SP),2(SP)       ;;      FIRST ASCII CHARACTER ON IT
007116  012766  007130  000004          MOV     #$TTYIN,4(SP)
007124  000002                          RTI                       ;;RETURN
007126     000                  9$:     .BYTE   0                 ;;STORAGE FOR ASCII CHAR. TO TYPE
007127     000                          .BYTE   0                 ;;TERMINATOR
007130  000010                  $TTYIN: .BLKB   8.                ;;RESERVE 8 BYTES FOR TTY INPUT
007140  052536  005015     000  $CNTLU: .ASCIZ  /^U/<15><12>       ;;CONTROL "U"
007145     136  006507  000012  $CNTLG: .ASCIZ  /^G/<15><12>       ;;CONTROL "G"
007152  005015  053523  020122  $MSWR:  .ASCIZ  <15><12>/SWR = /
007160  020075     000
007163     040  047040  053505  $MNEW:  .ASCIZ  /  NEW = /
007170  036440  000040

                                .SBTTL  TYPE ROUTINE
```

```
                                        ;;*****************************************************************
                                        ;*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
                                        ;*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.
                                        ;*NOTE1:        $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
                                        ;*NOTE2:        $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
                                        ;*NOTE3:        $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
                                        ;*
                                        ;*CALL:
                                        ;*1) USING A TRAP INSTRUCTION
                                        ;*      TYPE    ,MESADR         ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
                                        ;*OR
                                        ;*      TYPE
                                        ;*      MESADR
                                        ;*

007174  105737  001157          $TYPE:  TSTB    $TPFLG          ;;IS THERE A TERMINAL?
007200  100002                          BPL     1$              ;;BR IF YES
007202  000000                          HALT                    ;;HALT HERE IF NO TERMINAL
007204  000407                          BR      3$              ;;LEAVE
007206  010046                  1$:     MOV     R0,-(SP)        ;;SAVE R0
007210  017600  000002                  MOV     @2(SP),R0       ;;GET ADDRESS OF ASCIZ STRING
007214  112046                  2$:     MOVB    (R0)+,-(SP)     ;;PUSH CHARACTER TO BE TYPED ONTO STACK
007216  001005                          BNE     4$              ;;BR IF IT ISN'T THE TERMINATOR
007220  005726                          TST     (SP)+           ;;IF TERMINATOR POP IT OFF THE STACK
007222  012600                  60$:    MOV     (SP)+,R0        ;;RESTORE R0
007224  062716  000002          3$:     ADD     #2,(SP)         ;;ADJUST RETURN PC
007230  000002                          RTI                     ;;RETURN
007232  122716  000011          4$:     CMPB    #HT,(SP)        ;;BRANCH IF <HT>
007236  001430                          BEQ     8$
007240  122716  000200                  CMPB    #CRLF,(SP)      ;;BRANCH IF NOT <CRLF>
007244  001006                          BNE     5$
007246  005726                          TST     (SP)+           ;;POP  <CR><LF> EQUIV
007250  104401                          TYPE                    ;;TYPE A CR AND LF
007252  001213                          $CRLF
007254  105037  007410                  CLRB    $CHARCNT        ;;CLEAR CHARACTER COUNT
007260  000755                          BR      2$              ;;GET NEXT CHARACTER
007262  004737  007344          5$:     JSR     PC,$TYPEC       ;;GO TYPE THIS CHARACTER
007266  123726  001156          6$:     CMPB    $FILLC,(SP)+    ;;IS IT TIME FOR FILLER CHARS.?
007272  001350                          BNE     2$              ;;IF NO GO GET NEXT CHAR.
007274  013746  001154                  MOV     $NULL,-(SP)     ;;GET # OF FILLER CHARS. NEEDED
                                                                ;;AND THE NULL CHAR.
007300  105366  000001          7$:     DECB    1(SP)           ;;DOES A NULL NEED TO BE TYPED?
007304  002770                          BLT     6$              ;;BR IF NO--GO POP THE NULL OFF OF STACK
007306  004737  007344                  JSR     PC,$TYPEC       ;;GO TYPE A NULL
007312  105337  007410                  DECB    $CHARCNT        ;;DO NOT COUNT AS A COUNT
007316  000770                          BR      7$              ;;LOOP

                                        ;HORIZONTAL TAB PROCESSOR

007320  112716  000040          8$:     MOVB    #'  ,(SP)       ;;REPLACE TAB WITH SPACE
007324  004737  007344          9$:     JSR     PC,$TYPEC       ;;TYPE A SPACE
007330  132737  000007  007410          BITB    #7,$CHARCNT     ;;BRANCH IF NOT AT
007336  001372                          BNE     9$              ;;TAB STOP
007340  005726                          TST     (SP)+           ;;POP SPACE OFF STACK
007342  000724                          BR      2$              ;;GET NEXT CHARACTER
```

```
007344  105777  171600          $TYPEC: TSTB    @$TPS           ;;WAIT UNTIL PRINTER IS READY
007350  100375                          BPL     $TYPEC
007352  116677  000002  171572          MOVB    2(SP),@$TPB     ;;LOAD CHAR TO BE TYPED INTO DATA REG.
007360  122766  000015  000002          CMPB    #CR,2(SP)       ;;IS CHARACTER A CARRIAGE RETURN?
007366  001003                          BNE     1$              ;;BRANCH IF NO
007370  105037  007410                  CLRB    $CHARCNT        ;;YES--CLEAR CHARACTER COUNT
007374  000406                          BR      $TYPEX          ;;EXIT
007376  122766  000012  000002  1$:     CMPB    #LF,2(SP)       ;;IS CHARACTER A LINE FEED?
007404  001402                          BEQ     $TYPEX          ;;BRANCH IF YES
007406  105227                          INCB    (PC)+           ;;COUNT THE CHARACTER
007410  000000                  $CHARCNT:.WORD  0               ;;CHARACTER COUNT STORAGE
007412  000207                  $TYPEX: RTS     PC


                                .SBTTL  CONVERT BINARY TO DECIMAL AND TYPE ROUTINE

                                ;;*****************************************************************
                                ;*THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
                                ;*SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
                                ;*NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
                                ;*BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
                                ;*REPLACED WITH SPACES.
                                ;*CALL:
                                ;*      MOV     NUM,-(SP)       ;;PUT THE BINARY NUMBER ON THE STACK
                                ;*      TYPDS                   ;;GO TO THE ROUTINE

007414                          $TYPDS:
007414  010046                          MOV     R0,-(SP)        ;;PUSH R0 ON STACK
007416  010146                          MOV     R1,-(SP)        ;;PUSH R1 ON STACK
007420  010246                          MOV     R2,-(SP)        ;;PUSH R2 ON STACK
007422  010346                          MOV     R3,-(SP)        ;;PUSH R3 ON STACK
007424  010546                          MOV     R5,-(SP)        ;;PUSH R5 ON STACK
007426  012746  020200                  MOV     #20200,-(SP)    ;;SET BLANK SWITCH AND SIGN
007432  016605  000020                  MOV     20(SP),R5       ;;GET THE INPUT NUMBER
007436  100004                          BPL     1$              ;;BR IF INPUT IS POS.
007440  005405                          NEG     R5              ;;MAKE THE BINARY NUMBER POS.
007442  112766  000055  000001          MOVB    #'-,1(SP)       ;;MAKE THE ASCII NUMBER NEG.
007450  005000                  1$:     CLR     R0              ;;ZERO THE CONSTANTS INDEX
007452  012703  007630                  MOV     #$DBLK,R3       ;;SETUP THE OUTPUT POINTER
007456  112723  000040                  MOVB    #' ,(R3)+       ;;SET THE FIRST CHARACTER TO A BLANK
007462  005002                  2$:     CLR     R2              ;;CLEAR THE BCD NUMBER
007464  016001  007620                  MOV     $DTBL(R0),R1    ;;GET THE CONSTANT
007470  160105                  3$:     SUB     R1,R5           ;;FORM THIS BCD DIGIT
007472  002402                          BLT     4$              ;;BR IF DONE
007474  005202                          INC     R2              ;;INCREASE THE BCD DIGIT BY 1
007476  000774                          BR      3$
007500  060105                  4$:     ADD     R1,R5           ;;ADD BACK THE CONSTANT
007502  005702                          TST     R2              ;;CHECK IF BCD DIGIT=0
007504  001002                          BNE     5$              ;;FALL THROUGH IF 0
007506  105716                          TSTB    (SP)            ;;STILL DOING LEADING 0'S?
007510  100407                          BMI     7$              ;;BR IF YES
007512  106316                  5$:     ASLB    (SP)            ;;MSD?
007514  103003                          BCC     6$              ;;BR IF NO
007516  116663  000001  177777          MOVB    1(SP),-1(R3)    ;;YES--SET THE SIGN
007524  052702  000060          6$:     BIS     #'0,R2          ;;MAKE THE BCD DIGIT ASCII
007530  052702  000040          7$:     BIS     #' ,R2          ;;MAKE IT A SPACE IF NOT ALREADY A DIGIT
```

```
007534  110223                          MOVB    R2,(R3)+        ;;PUT THIS CHARACTER IN THE OUTPUT BUFFER
007536  005720                          TST     (R0)+           ;;JUST INCREMENTING
007540  020027  000010                  CMP     R0,#10          ;;CHECK THE TABLE INDEX
007544  002746                          BLT     2$              ;;GO DO THE NEXT DIGIT
007546  003002                          BGT     8$              ;;GO TO EXIT
007550  010502                          MOV     R5,R2           ;;GET THE LSD
007552  000764                          BR      6$              ;;GO CHANGE TO ASCII
007554  105726                  8$:     TSTB    (SP)+           ;;WAS THE LSD THE FIRST NON-ZERO?
007556  100003                          BPL     9$              ;;BR IF NO
007560  116663  177777  177776          MOVB    -1(SP),-2(R3)   ;;YES--SET THE SIGN FOR TYPING
007566  105013                  9$:     CLRB    (R3)            ;;SET THE TERMINATOR
007570  012605                          MOV     (SP)+,R5        ;;POP STACK INTO R5
007572  012603                          MOV     (SP)+,R3        ;;POP STACK INTO R3
007574  012602                          MOV     (SP)+,R2        ;;POP STACK INTO R2
007576  012601                          MOV     (SP)+,R1        ;;POP STACK INTO R1
007600  012600                          MOV     (SP)+,R0        ;;POP STACK INTO R0
007602  104401  007630                  TYPE    $DBLK           ;;NOW TYPE THE NUMBER
007606  016666  000002  000004          MOV     2(SP),4(SP)     ;;ADJUST THE STACK
007614  012616                          MOV     (SP)+,(SP)
007616  000002                          RTI                     ;;RETURN TO USER
007620  023420                  $DTBL:  10000.
007622  001750                          1000.
007624  000144                          100.
007626  000012                          10.
007630  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

007640  017646  000000          $TYPOS: MOV     @(SP),-(SP)     ;;PICKUP THE MODE
007644  116637  000001  010063          MOVB    1(SP),$OFILL    ;;LOAD ZERO FILL SWITCH
007652  112637  010065                  MOVB    (SP)+,$OMODE+1  ;;NUMBER OF DIGITS TO TYPE
007656  062716  000002                  ADD     #2,(SP)         ;;ADJUST RETURN ADDRESS
007662  000406                          BR      $TYPON
```

```
007664  112737  000001  010063  $TYPOC: MOVB    #1,$OFILL          ;;SET THE ZERO FILL SWITCH
007672  112737  000006  010065          MOVB    #6,$OMODE+1        ;;SET FOR SIX(6) DIGITS
007700  112737  000005  010062  $TYPON: MOVB    #5,$OCNT           ;;SET THE ITERATION COUNT
007706  010346                          MOV     R3,-(SP)           ;;SAVE R3
007710  010446                          MOV     R4,-(SP)           ;;SAVE R4
007712  010546                          MOV     R5,-(SP)           ;;SAVE R5
007714  113704  010065                  MOVB    $OMODE+1,R4        ;;GET THE NUMBER OF DIGITS TO TYPE
007720  005404                          NEG     R4
007722  062704  000006                  ADD     #6,R4              ;;SUBTRACT IT FOR MAX. ALLOWED
007726  110437  010064                  MOVB    R4,$OMODE          ;;SAVE IT FOR USE
007732  113704  010063                  MOVB    $OFILL,R4          ;;GET THE ZERO FILL SWITCH
007736  016605  000012                  MOV     12(SP),R5          ;;PICKUP THE INPUT NUMBER
007742  005003                          CLR     R3                 ;;CLEAR THE OUTPUT WORD
007744  006105                  1$:     ROL     R5                 ;;ROTATE MSB INTO "C"
007746  000404                          BR      3$                 ;;GO DO MSB
007750  006105                  2$:     ROL     R5                 ;;FORM THIS DIGIT
007752  006105                          ROL     R5
007754  006105                          ROL     R5
007756  010503                          MOV     R5,R3
007760  006103                  3$:     ROL     R3                 ;;GET LSB OF THIS DIGIT
007762  105337  010064                  DECB    $OMODE             ;;TYPE THIS DIGIT?
007766  100016                          BPL     7$                 ;;BR IF NO
007770  042703  177770                  BIC     #177770,R3         ;;GET RID OF JUNK
007774  001002                          BNE     4$                 ;;TEST FOR 0
007776  005704                          TST     R4                 ;;SUPPRESS THIS 0?
010000  001403                          BEQ     5$                 ;;BR IF YES
010002  005204                  4$:     INC     R4                 ;;DON'T SUPPRESS ANYMORE 0'S
010004  052703  000060                  BIS     #'0,R3             ;;MAKE THIS DIGIT ASCII
010010  052703  000040          5$:     BIS     #' ,R3             ;;MAKE ASCII IF NOT ALREADY
010014  110337  010060                  MOVB    R3,8$              ;;SAVE FOR TYPING
010020  104401  010060                  TYPE    ,8$                ;;GO TYPE THIS DIGIT
010024  105337  010062          7$:     DECB    $OCNT              ;;COUNT BY 1
010030  003347                          BGT     2$                 ;;BR IF MORE TO DO
010032  002402                          BLT     6$                 ;;BR IF DONE
010034  005204                          INC     R4                 ;;INSURE LAST DIGIT ISN'T A BLANK
010036  000744                          BR      2$                 ;;GO DO THE LAST DIGIT
010040  012605                  6$:     MOV     (SP)+,R5           ;;RESTORE R5
010042  012604                          MOV     (SP)+,R4           ;;RESTORE R4
010044  012603                          MOV     (SP)+,R3           ;;RESTORE R3
010046  016666  000002  000004          MOV     2(SP),4(SP)        ;;SET THE STACK FOR RETURNING
010054  012616                          MOV     (SP)+,(SP)
010056  000002                          RTI                        ;;RETURN
010060     000                  8$:     .BYTE   0                  ;;STORAGE FOR ASCII DIGIT
010061     000                          .BYTE   0                  ;;TERMINATOR FOR TYPE ROUTINE
010062     000                  $OCNT:  .BYTE   0                  ;;OCTAL DIGIT COUNTER
010063     000                  $OFILL: .BYTE   0                  ;;ZERO FILL SWITCH
010064  000000                  $OMODE: .WORD   0                  ;;NUMBER OF DIGITS TO TYPE

                                .SBTTL  TRAP DECODER

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

```
010066  010046                  $TRAP:  MOV     R0,-(SP)          ;;SAVE R0
010070  016600  000002                  MOV     2(SP),R0          ;;GET TRAP ADDRESS
010074  005740                          TST     -(R0)             ;;BACKUP BY 2
010076  111000                          MOVB    (R0),R0           ;;GET RIGHT BYTE OF TRAP
010100  006300                          ASL     R0                ;;POSITION FOR INDEXING
010102  016000  010122                  MOV     $TRPAD(R0),R0     ;;INDEX TO TABLE
010106  000200                          RTS     R0                ;;GO TO ROUTINE


                                ;;THIS IS USE TO HANDLE THE "GETPRI" MACRO

010110  011646                  $TRAP2: MOV     (SP),-(SP)        ;;MOVE THE PC DOWN
010112  016666  000004  000002          MOV     4(SP),2(SP)       ;;MOVE THE PSW DOWN
010120  000002                          RTI                       ;;RESTORE THE PSW

                                .SBTTL  TRAP TABLE


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

                                ;       ROUTINE
                                ;       -------
010122  010110                  $TRPAD: .WORD   $TRAP2
010124  007174                          $TYPE   ;;CALL=TYPE     TRAP+1(104401)  TTY TYPEOUT ROUTINE
010126  007664                          $TYPOC  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
010130  007640                          $TYPOS  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
010132  007700                          $TYPON  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
010134  007414                          $TYPDS  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)

010136  006470                          $GTSWR  ;;CALL=GTSWR    TRAP+6(104406)  GET SOFT-SWR SETTING

010140  006420                          $CKSWR  ;;CALL=CKSWR    TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
010142  006702                          $RDCHR  ;;CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
010144  007022                          $RDLIN  ;;CALL=RDLIN    TRAP+11(104411) TTY TYPEIN STRING ROUTINE

010146  005516                          CN.RST  ;;CALL=CNT.RESET        TRAP+12(104412) CONTROL RESET ROUTINE

010150  005534                          CN.RDY  ;;CALL=CNT.RDY  TRAP+13(104413) WAIT FOR CNTRL RDY TO SET

010152  005474                          DELA.Y  ;;CALL=DELAY    TRAP+14(104414) TIME DELAY ROUTINE


                                .SBTTL  POWER DOWN AND UP ROUTINES

                                ;;*************************************************************
                                ;POWER DOWN ROUTINE
010154  012737  010320  000024  $PWRDN: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST UP
010162  012737  000340  000026          MOV     #340,@#PWRVEC+2 ;;PRIO:7
010170  010046                          MOV     R0,-(SP)          ;;PUSH R0 ON STACK
010172  010146                          MOV     R1,-(SP)          ;;PUSH R1 ON STACK
010174  010246                          MOV     R2,-(SP)          ;;PUSH R2 ON STACK
010176  010346                          MOV     R3,-(SP)          ;;PUSH R3 ON STACK
010200  010446                          MOV     R4,-(SP)          ;;PUSH R4 ON STACK
010202  010546                          MOV     R5,-(SP)          ;;PUSH R5 ON STACK
010204  017746  170730                  MOV     @SWR,-(SP)        ;;PUSH @SWR ON STACK
010210  010637  010324                  MOV     SP,$SAVR6         ;;SAVE SP
```

```
010214  012737  010226  000024          MOV     #$PWRUP,@#PWRVEC ;;SET UP VECTOR
010222  000000                          HALT
010224  000776                          BR      .-2              ;;HANG UP

                                ;;*****************************************************************
                                ;POWER UP ROUTINE
010226  012737  010320  000024  $PWRUP: MOV     #$ILLUP,@#PWRVEC ;;SET FOR FAST DOWN
010234  013706  010324                  MOV     $SAVR6,SP        ;;GET SP
010240  005037  010324                  CLR     $SAVR6           ;;WAIT LOOP FOR THE TTY
010244  005237  010324          1$:     INC     $SAVR6           ;;WAIT FOR THE INC
010250  001375                          BNE     1$               ;;OF  WORD
010252  012677  170662                  MOV     (SP)+,@SWR       ;;POP STACK INTO @SWR
010256  012605                          MOV     (SP)+,R5         ;;POP STACK INTO R5
010260  012604                          MOV     (SP)+,R4         ;;POP STACK INTO R4
010262  012603                          MOV     (SP)+,R3         ;;POP STACK INTO R3
010264  012602                          MOV     (SP)+,R2         ;;POP STACK INTO R2
010266  012601                          MOV     (SP)+,R1         ;;POP STACK INTO R1
010270  012600                          MOV     (SP)+,R0         ;;POP STACK INTO R0
010272  012737  010154  000024          MOV     #$PWRDN,@#PWRVEC ;;SET UP THE POWER DOWN VECTOR
010300  012737  000340  000026          MOV     #340,@#PWRVEC+2  ;;PRIO:7
010306  104401                          TYPE                     ;REPORT THE POWER FAILURE
010310  010326                  $PWRMG: .WORD   $POWER           ;;POWER FAIL MESSAGE POINTER
010312  012716                          MOV     (PC)+,(SP)       ;;RESTART AT PFSTRT
010314  002316                  $PWRAD: .WORD   PFSTRT           ;;RESTART ADDRESS
010316  000002                          RTI
010320  000000                  $ILLUP: HALT                     ;;THE POWER UP SEQUENCE WAS STARTED
010322  000776                          BR      .-2              ;; BEFORE THE POWER DOWN WAS COMPLETE
010324  000000                  $SAVR6: 0                        ;;PUT THE SP HERE
010326  005015  047520  042527  $POWER: .ASCIZ  <15><12>"POWER"
010334  000122
                                        .EVEN

                                        ;ERROR MESSAGES

                                        .SBTTL  ERROR MESSAGES

010336  044524  042515  047440  EM1:    .ASCIZ  /TIME OUT ON RK11 REGISTER/
010344  052125  047440  020116
010352  045522  030461  051040
010360  043505  051511  042524
010366  000122

010370  042522  044507  052123  EM2:    .ASCIZ  /REGISTER NOT CLEARED/
010376  051105  047040  052117
010404  041440  042514  051101
010412  042105     000
010415     122  041513  020123  EM3:    .ASCIZ  /RKCS ERROR/
010422  051105  047522  000122

010430  045522  051503  042440  EM4:    .ASCIZ  /RKCS ERROR-ON WRITING READ ONLY BITS/
010436  051122  051117  047455
010444  020116  051127  052111
010452  047111  020107  042522
010460  042101  047440  046116
010466  020131  044502  051524
```

```
010474     000

010475     102  051525  044440  EM5:    .ASCIZ  /BUS INIT DID NOT CLEAR RKCS/
010502  044516  020124  044504
010510  020104  047516  020124
010516  046103  040505  020122
010524  045522  051503     000

010531     103  052116  046122  EM6:    .ASCIZ  /CNTRL RESET DIDN'T CLEAR RKCS, ON SETING 'GO'/
010536  051040  051505  052105
010544  042040  042111  023516
010552  020124  046103  040505
010560  020122  045522  051503
010566  020054  047117  051440
010574  052105  047111  020107
010602  043447  023517     000

010607     103  052116  046122  EM7:    .ASCIZ  /CNTRL RDY DIDN'T SET AFTER CNTRL RESET/
010614  051040  054504  042040
010622  042111  023516  020124
010630  042523  020124  043101
010636  042524  020122  047103
010644  051124  020114  042522
010652  042523  000124

010656  042522  044507  052123  EM10:   .ASCIZ  /REGISTER NOT CLEARED/
010664  051105  047040  052117
010672  041440  042514  051101
010700  042105     000

010703     122  052513  020103  EM11:   .ASCIZ  /RKWC ERROR/
010710  051105  047522  000122

010716  045522  040502  042440  EM13:   .ASCIZ  /RKBA ERROR/
010724  051122  051117     000

010731     103  052116  046122  EM14:   .ASCIZ  /CNTRL RESET DIDN'T CLEAR REGISTER/
010736  051040  051505  052105
010744  042040  042111  023516
010752  020124  046103  040505
010760  020122  042522  044507
010766  052123  051105     000

010773     122  042113  020101  EM15:   .ASCIZ  /RKDA ERROR/
011000  051105  047522  000122

011006  052502  020123  047111  EM17:   .ASCIZ  /BUS INIT DIDN'T CLR REGISTR/
011014  052111  042040  042111
011022  023516  020124  046103
011030  020122  042522  044507
011036  052123  000122

011042  042101  051104  051505  EM20:   .ASCIZ  /ADDRESSING ERROR-TRIED TO ADDRESS REG1, GOT REG2/
011050  044523  043516  042440
011056  051122  051117  052055
```

```
011064  044522  042105  052040
011072  020117  042101  051104
011100  051505  020123  042522
011106  030507  020054  047507
011114  020124  042522  031107
011122     000

011123     104  042111  023516  EM22:   .ASCIZ  /DIDN'T CLEAR RKCS LOW BYTE/
011130  020124  046103  040505
011136  020122  045522  051503
011144  046040  053517  041040
011152  052131  000105

011156  044504  047104  052047  EM23:   .ASCIZ  /DIDN'T CLEAR RKCS HIGH BYTE/
011164  041440  042514  051101
011172  051040  041513  020123
011200  044510  044107  041040
011206  052131  000105

011212  051124  042511  020104  EM24:   .ASCIZ  /TRIED TO CLEAR RKCS 'BYTE', CHANGED 'REGIS'/
011220  047524  041440  042514
011226  051101  051040  041513
011234  020123  041047  052131
011242  023505  020054  044103
011250  047101  042507  020104
011256  051047  043505  051511
011264  000047

011266  040506  046111  042105  EM25:   .ASCIZ  /FAILED TO CLEAR 'REG-BYTE'/
011274  052040  020117  046103
011302  040505  020122  051047
011310  043505  041055  052131
011316  023505     000

011321     122  041513  020123  EM26:   .ASCIZ  /RKCS ALTERED ON CLEARING 'REG-BYTE'/
011326  046101  042524  042522
011334  020104  047117  041440
011342  042514  051101  047111
011350  020107  051047  043505
011356  041055  052131  023505
011364     000

011365     124  044522  042105  EM27:   .ASCIZ  /TRIED TO CLEAR 'REG-BYT1', CHANGED 'REG-BYT2'/
011372  052040  020117  046103
011400  040505  020122  051047
011406  043505  041055  052131
011414  023461  020054  044103
011422  047101  042507  020104
011430  051047  043505  041055
011436  052131  023462     000

011443     125  042516  050130  EM43:   .ASCIZ  /UNEXPECTED RK11 INTERRUPT/
011450  041505  042524  020104
011456  045522  030461  044440
011464  052116  051105  052522
011472  052120     000
```

```
011476                                  .EVEN

                                        .SBTTL  ERROR DATA POINTERS

011476  001116  001162  000000  DT1:    .WORD   $ERRPC,$REG0,0

011504  001116  001162  001164  DT2:    .WORD   $ERRPC,$REG0,$REG1,0
011512  000000

011514  001116  001162  001164  DT20:   .WORD   $ERRPC,$REG0,$REG1,$REG2,$REG3,0
011522  001166  001170  000000

011530  001116  000000          DT21:   .WORD   $ERRPC,0

011534  001116  001162  001164  DT25:   .WORD   $ERRPC,$REG0,$REG1,$REG2,0
011542  001166  000000



                                        .SBTTL  ERROR HEADERS

011546  020040  041520  020040  DH1:    .ASCIZ  /  PC   REG-ADDR/
011554  051040  043505  040455
011562  042104  000122

011566  020040  041520  020040  DH2:    .ASCIZ  /  PC   REGADD    RECVD/
011574  051040  043505  042101
011602  020104  020040  051040
011610  041505  042126     000

011615     040  050040  020103  DH4:    .ASCIZ  /  PC    EXPCT   RECVD/
011622  020040  042440  050130
011630  052103  020040  051040
011636  041505  042126     000

011643     040  050040  020103  DH3:    .ASCIZ  /  PC    WROTE   READ/
011650  020040  053440  047522
011656  042524  020040  051040
011664  040505  000104

011670  020040  041520  020040  DH5:    .ASCIZ  /  PC    RECVD/
011676  020040  042522  053103
011704  000104

011706  020040  041520  020040  DH11:   .ASCIZ  /  PC    WROTE   READ/
011714  020040  051127  052117
011722  020105  020040  042522
011730  042101     000

011733     040  050040  000103  DH21:   .ASCIZ  /  PC/

011740  020040  041520  020040  DH20:   .ASCIZ  /  PC    REG1     REG2   (REG1)  (REG2)/
011746  020040  042522  030507
011754  020040  020040  051040
011762  043505  020062  020040
```

```
011770  051050  043505  024461
011776  020040  051050  043505
012004  024462     000

012007     040  050040  020103  DH24:   .ASCIZ  /  PC     BYTE   REGIS (REG)EXP (REG)RECVD/
012014  020040  020040  054502
012022  042524  020040  051040
012030  043505  051511  024040
012036  042522  024507  054105
012044  020120  051050  043505
012052  051051  041505  042126
012060     000

012061     040  050040  020103  DH25:   .ASCIZ  /  PC   REG-BYTE RECVD/
012066  020040  042522  026507
012074  054502  042524  051040
012102  041505  042126     000

012107     040  050040  020103  DH26:   .ASCIZ  /  PC   REG-BYT  (CS)EXP (CS)RECVD/
012114  020040  042522  026507
012122  054502  020124  024040
012130  051503  042451  050130
012136  024040  051503  051051
012144  041505  042126     000

012151     040  050040  020103  DH27:   .ASCIZ  /  PC    R-BYT1  R-BYT2  2-EXPCT 2-RECVD/
012156  020040  051040  041055
012164  052131  020061  051040
012172  041055  052131  020062
012200  031040  042455  050130
012206  052103  031040  051055
012214  041505  042126     000

012221     040  050040  020103  DH30:   .ASCIZ  /  PC     RKCS    RKER    RKDS/
012226  020040  020040  045522
012234  051503  020040  020040
012242  045522  051105  020040
012250  020040  045522  051504
012256     000

        000001                          .END
```

```
BADINT  002212
BADTMO  002126
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  005534
CN.RST  005516
CR    = 000015
CRLF  = 000200
DDISP = 177570
DELAY = 104414
DELA.Y  005474
DH1     011546
DH11    011706
DH2     011566
DH20    011740
DH21    011733
DH24    012007
DH25    012061
DH26    012107
DH27    012151
DH3     011643
DH30    012221
DH4     011615
DH5     011670
DISPLA  001142
DISPRE  000174
DSWR  = 177570
DT1     011476
DT2     011504
DT20    011514
DT21    011530
DT26    011534
EMTVEC= 000030
EM1     010336
EM10    010656
EM11    010703
EM13    010716
EM14    010731
EM15    010773
EM17    011006
EM2     010370
EM20    011042
EM22    011123
EM23    011156
EM24    011212
EM25    011266
EM26    011321
EM27    011365
EM3     010415
EM4     010430
EM43    011443
EM5     010475
EM6     010531
EM7     010607
ERRVEC= 000004
FTITLE  001262
GTSWR = 104406
GT3RG   005434
GT4RG   005460
HT    = 000011
IOTVEC= 000020
LF    = 000012
MSG3    001216
PFSTRT  002316
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  002110
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  002466
TKVEC = 000060
TPVEC = 000064
TRAPVE= 000034
TRTVEC= 000014
TST1    002362
TST10   003214
TST11   003324
TST12   003464
TST13   003532
TST14   003640
TST15   003706
TST16   004014
TST17   004066
TST2    002504
TST20   004174
TST21   004250
TST22   004410
TST23   004600
TST24   005014
TST3    002570
TST4    002650
TST5    002730
TST6    003102
TST7    003152
TYPDS = 104405
TYPE  = 104401
TYPOC = 104402
TYPON = 104404
TYPOS = 104403
T1      002412
$AUTOB  001134
$BDADR  001122
$BDDAT  001126
$CHARC  007410
$CKSWR  006420
$CMTAG  001100
$CM1  = 000012
$CM2  = 000024
$CM3  = 000012
$CNTLG  007145
$CNTLU  007140
$CRLF   001213
$DBLK   007630
$DOAGN  005410
$DTBL   007620
$ENDAD  005400
$ENDCT  005346
$ENDMG  005417
$ENULL  005414
$EOP    005312
$EOPCT  005340
$ERFLG  001103
$ERMAX  001115
$ERROR  006114
$ERRPC  001116
$ERRTB  001272
$ERRTY  006264
$ERTTL  001112
$ESCAP  001210
$FILLC  001156
$FILLS  001155
$GDADR  001120
$GDDAT  001124
$GET42  005370
$GTSWR  006470
$HD   = 000000
$ICNT   001104
$ILLUP  010320
$INTAG  001135
$ITEMB  001114
$LF     001214
$LPADR  001106
$LPERR  001110
$MNEW   007163
$MSWR   007152
$MXCNT  006112
$NULL   001154
$NWTST= 000001
$OCNT   010062
$OMODE  010064
$OVER   006076
$PASS   001100
$POWER  010326
$PWRAD  010314
$PWRDN  010154
$PWRMG  010310
$PWRUP  010226
$QUES   001212
$RDCHR  006702
$RDLIN  007022
$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  005412
$SAVR6  010324
$SCOPE  005642
$SETUP= 000117
$STUP = 177777
$SVLAD  006050
$SVPC = 000204
$SWR  = 165400
$SWRMK= 000000
```

```
$TIMES  001206          $TPFLG  001157          $TRPAD  010122          $TYPEC  007344          $XTSTR  005654
$TKB    001146          $TPS    001150          $TSTNM  001102          $TYPEX  007412          $$GET4= 000000
$TKS    001144          $TRAP   010066          $TTYIN  007130          $TYPOC  007664          $OFILL  010063
$TN   = 000025          $TRAP2  010110          $TYPDS  007414          $TYPON  007700          .     = 012257
$TPB    001152          $TRP  = 000015          $TYPE   007174          $TYPOS  007640

. ABS.  012257    000


 ERRORS DETECTED:  0

 .DSKZ:DZRKJE/SOL=DSKZ:SYSMAC.SML,DSKM:DZRKJE.P11
 RUN-TIME: 11 14 .3 SECONDS
 RUN-TIME RATIO: 430/27=15.8
 CORE USED:  32K  (63 PAGES)


EOF1DZRKJESEQ           00010000      770712          PDP10 411
```