

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

B01

00010000

00010000

770712

PDP10 411

HORIDZRYJESE0

00010000

770712

REV 1

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 TIMEOUT ROUTINE
	POWER DOWN/POWER UP ROUTINE
11.0	UNEXPECTED TIMEOUTS & RK11 INTERRUPTS
12.0	QUICK VERIFYING 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-DZKJ-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-DZKJ-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. 'RJBOU' 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 PROGRAM TO CYCLE FROM THE POINT OF ERROR TO THE PREVIOUS SCOPE STATEMENT. NOTE THAT IN DOING SO ANY INITIALIZATION BEING DONE AT THE BEGINNING 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 STARTING 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 EARLIER THE ERROR 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 HAPPENS 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$,SLPERR      : '$SLPERR' CONTAINS
        :                                     :THE ADDRESS TO LOOP
        :                                     :BACK ON ERROR- SW 9
1$:     :
        :-----
        :I
        :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.

9.0 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. Q - 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 DIFFERENT 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
'SERRTYP'.

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"

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

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

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

SPWRDN, SPWRUP:
ROUTINE FOR HANDLING POWER FAILURE/POWER UP.
LOCATED AT SPWRDN, SPWRUP
SPWRFL, CALLED WHEN THERE IS A POWER FAILURE.
SPWRUP, CALLED WHEN THERE IS A POWER UP.

11.0 UNEXPECTED TIMEOUTS AND RK11 INTERRUPTS

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

IF AN UNEXPECTED RK11 INTERRUPT OCCURS 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.

%

MD-11-DZKJ-E
DZKJE.P11
19-APR-77
09:14

BU2

```
.TITLE MD-11-DZKJ-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 ***
```

```
STACK= 1100
.EQUIV EMT,ERROR      ;;BASIC DEFINITION OF ERROR CALL
.EQUIV IOT,SCOPE      ;;BASIC DEFINITION OF SCOPE CALL
```

```
.*MISCELLANEOUS DEFINITIONS
```

```
HT= 11      ;;CODE FOR HORIZONTAL TAB
LF= 12      ;;CODE FOR LINE FEED
CR= 15      ;;CODE FOR CARRIAGE RETURN
CRLF= 200   ;;CODE FOR CARRIAGE RETURN-LINE FEED
PS= 177776  ;;PROCESSOR STATUS WORD
.EQUIV PS,PSW
```

001100

000011
000012
000015
000200
177776

177774
177772
177570
177570STKLMT= 177774 :: STACK LIMIT REGISTER
PIRQ= 177772 :: PROGRAM INTERRUPT REQUEST REGISTER
DSWR= 177570 :: HARDWARE SWITCH REGISTER
DDISP= 177570 :: HARDWARE DISPLAY REGISTER

.*GENERAL PURPOSE REGISTER DEFINITIONS

000000
000001
000002
000003
000004
000005
000006
000007
000006
000007R0= %0 :: GENERAL REGISTER
R1= %1 :: GENERAL REGISTER
R2= %2 :: GENERAL REGISTER
R3= %3 :: GENERAL REGISTER
R4= %4 :: GENERAL REGISTER
R5= %5 :: GENERAL REGISTER
R6= %6 :: GENERAL REGISTER
R7= %7 :: GENERAL REGISTER
SP= %6 :: STACK POINTER
PC= %7 :: PROGRAM COUNTER

.*PRIORITY LEVEL DEFINITIONS

000000
000040
000100
000140
000200
000240
000300
000340PR0= 0 :: PRIORITY LEVEL 0
PR1= 40 :: PRIORITY LEVEL 1
PR2= 100 :: PRIORITY LEVEL 2
PR3= 140 :: PRIORITY LEVEL 3
PR4= 200 :: PRIORITY LEVEL 4
PR5= 240 :: PRIORITY LEVEL 5
PR6= 300 :: PRIORITY LEVEL 6
PR7= 340 :: PRIORITY LEVEL 7

.*"SWITCH REGISTER" SWITCH DEFINITIONS

100000
040000
020000
010000
004000
002000
001000
000400
000200
000100
000040
000020
000010
000004
000002
000001SW15= 100000
SW14= 40000
SW13= 20000
SW12= 10000
SW11= 4000
SW10= 2000
SW09= 1000
SW08= 400
SW07= 200
SW06= 100
SW05= 40
SW04= 20
SW03= 10
SW02= 4
SW01= 2
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
 040000
 020000
 010000
 004000
 002000
 001000
 000400
 000200
 000100
 000040
 000020
 000010
 000004
 000002
 000001

 BIT15= 100000
 BIT14= 40000
 BIT13= 20000
 BIT12= 10000
 BIT11= 4000
 BIT10= 2000
 BIT09= 1000
 BIT08= 400
 BIT07= 200
 BIT06= 100
 BIT05= 40
 BIT04= 20
 BIT03= 10
 BIT02= 4
 BIT01= 2
 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

 000004
 000010
 000014
 000014
 000014
 000020
 000024
 000030
 000034
 000060
 000064
 000240

 .;BASIC "CPU" TRAP VECTOR ADDRESSES
 ERRVEC= 4 .;TIME OUT AND OTHER ERRORS
 RESVEC= 10 .;RESERVED AND ILLEGAL INSTRUCTIONS
 TBITVEC= 14 .;"T" BIT
 TRTVEC= 14 .;TRACE TRAP
 BPTVEC= 14 .;BREAKPOINT TRAP (BPT)
 IOTVEC= 20 .;INPUT/OUTPUT TRAP (IOT) **SCOPE**
 PWRVEC= 24 .;POWER FAIL
 EMTVEC= 30 .;EMULATOR TRAP (EMT) **ERROR**
 TRAPVEC= 34 .;"TRAP" TRAP
 TKVEC= 60 .;TTY KEYBOARD VECTOR
 TPVEC= 64 .;TTY PRINTER VECTOR
 PIQVEC= 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 000000
 000176 000000
 000200 000137 001542

 .;=174
 DISPREG: .WORD 0 .;SOFTWARE DISPLAY REGISTER
 SWREG: .WORD 0 .;SOFTWARE SWITCH REGISTER
 .SBTTL STARTING ADDRESS(ES)
 JMP @#START .;JUMP TO STARTING ADDRESS OF PROGRAM
 .SBTTL ACT11 HOOKS

000204

 .;*****
 .;HOOKS REQUIRED BY ACT11
 .SS.PC=. .;SAVE PC

838		000046	. =46	
0039	000046	005400	SENDAD	::1)SET LOC.46 TO ADDRESS OF SENDAD :4 .SECP
0040		000052	. =52	
0041	000052	000000	.WORD 0	::2)SET LOC.52 TO ZEP0
0042		000204	. =SSVPC	:: RESTORE PC

900 001234 042122 020131 044504
 901 001232 047104 052047 051440
 902 001240 052105 000
 903 001244
 904
 905
 906
 907
 908
 909
 910
 911 001244 177400
 912 001246 177402
 913 001250 177404
 914 001252 177406
 915 001254 177410
 916 001256 177412
 917 001260 177416
 918
 919
 920
 921
 922
 923
 924 001262 000000
 925 001264 000000
 926 001266 000200
 927
 928
 929
 930
 931 001270 000220
 932
 933

.EVEN

;RK11 REGISTERS
 ;IF FOR ANY REASON THE REGISTER ADDRESSES ARE DIFFERENT FROM THESE
 ; (GIVEN BELOW), THE CONTENTS OF THE APPROPRIATE POINTERS SHOULD BE
 ; MODIFIED SO THAT THE CORRECT ADDRESS IS USED.

;
 ;.EVEN
 RKDS: 177400
 RKER: 177402
 RKCS: 177404
 RKWC: 177406
 RKBA: 177410
 RKDA: 177412
 RKDB: 177416

;TAGS AND GENERAL DATA AREA

;
 ;

FTITLE: 0
 TIMER: 0
 RKPRI: 200

;FLAG FOR PRINTING PROGRAM TITLE
 ;TIMER REGISTER
 ;CONTAINS THE CPU LEVEL AT WHICH
 ;RK11 NORMALLY INTERRUPTS. THIS WORD
 ;SHOULD BE CHANGED IF RK11 IS DESINGATED
 ;A BR LEVEL OTHER THAN 5. E.G. IF IT IS CHANGED
 ;TO 6, THIS WORD SHOULD BE CHANGED TO 240.
 ;CONTAINS THE NORMAL VECTOR ADDRESS TO WHICH
 ;RK11 INTERRUPTS. IF THIS IS NOT SO, CHANGE
 ;THIS WORD TO CONTAIN MODIFIED VECTOR ADDRESS.

RKVEC: 220

.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 RKCS
 :177400, BECAUSE OF SOME FAULT, THE FOLLOWING 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 'SEERROR'. 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

990			:ITEM	1	
991	001272	010336		EM1	:TIME OUT ON RK11 REG
992	001274	011546		DH1	:PC REG
993	001276	011476		DT1	:SERRPC \$REGO
994	001300	000000		0	
995					
996			:ITEM	2	
997				EM2	:REGISTER NOT CLEARED
998	001302	010370		DH2	:PC REGADD RECVD
999	001304	011566		DT2	:SERRPC \$REGO \$REG1
1000	001306	011504		0	
1001	001310	000000			
1002					
1003			:ITEM	3	
1004				EM3	:RKCS ERROR
1005	001312	010415		DH3	:PC WROTE READ
1006	001314	011643		DT2	:SERRPC \$REGO \$REG1
1007	001316	011504		0	
1008	001320	000000			
1009					
1010			:ITEM	4	
1011				EM4	:RKCS ERROR-ON WRITING READ ONLY BITS
1012	001322	010430		DH4	:PC EXPT RECVD
1013	001324	011615		DT2	:SERRPC \$REGO \$REG1
1014	001326	011504		0	
1015	001330	000000			
1016					
1017			:ITEM	5	
1018				EM5	:BUS INIT DID NOT CLEAR RKCS
1019	001332	010475		DH5	:PC RECVD
1020	001334	011670		DT1	:SERRPC \$REGO
1021	001336	011476		0	
1022	001340	000000			
1023					
1024			:ITEM	6	
1025				EM6	: 'CNTRL RESET' DIDN'T CLEAR RKCS, ON SETTING GO
1026	001342	010531		DH5	:PC RECVD
1027	001344	011670		DT1	:SERRPC \$REGO
1028	001346	011476		0	
1029	001350	000000			
1030					
1031			:ITEM	7	
1032				EM7	: 'CNTRL RDY' DIDN'T SET AFTER CONTROL RESET
1033	001352	010607		DH30	:PC RKCS RKER RKDS
1034	001354	012221		DT26	:SERRPC \$REGO \$REG1 \$REG2
1035	001356	011534		0	
1036	001360	000000			
1037					
1038			:ITEM	10	
1039				EM10	:REGISTER NOT CLEARED
1040	001362	010656		DH2	:PC REGADD RECVD
1041	001364	011566		DT2	:SERRPC \$REGO \$REG1
1042	001366	011504		0	
1043	001370	000000			
1044					
1045					
1046					
1047					
1048					
1049					
1050					


```

1046      :ITEM 11
1047      :
1048      001372 010703      EM11      :RKWC ERROR
1049      001374 011706      DH11      :PC      WROTE      READ
1050      001376 011504      DT2       :$ERRPC $REG0      $REG1
1051      001400 000000      0
1052
1053      :ITEM 12
1054      :
1055      001402 011443      EM43      :UNEXPECTED PK11 INTERRUPT
1056      001404 011733      DH21      :PC
1057      001406 011530      DT21      :$ERRPC
1058      001410 000000      0
1059
1060      :ITEM 13
1061      :
1062      001412 010716      EM13      :RKBA ERROR
1063      001414 011706      DH11      :PC      WROT      READ
1064      001416 011504      DT2       :$ERRPC $REG0      $REG1
1065      001420 000000      0
1066
1067      :ITEM 14
1068      :
1069      001422 010731      EM14      :CNTRL RESET DID NOT CLEAR REGISTER
1070      001424 011566      DH2       :PC      REGADD      RECVD
1071      001426 011504      DT2       :$ERRPC $REG0      $REG1
1072      001430 000000      0
1073
1074      :ITEM 15
1075      :
1076      001432 010773      EM15      :RKDA ERROR
1077      001434 011706      DH11      :PC      WROTE      READ
1078      001436 011504      DT2       :$ERRPC $REG0      $REG1
1079      001440 000000      0
1080
1081      :ITEM 16
1082      :
1083      001442 011321      EM26      :RKCS ALTERED ON CLEARING 'REG-BYTE'
1084      001444 012107      DH26      :PC      REG-BYTE (RKCS)EXP (RKCS)RECVD
1085      001446 011534      DT26      :$ERRPC $REG0      $REG1      $REG2
1086      001450 000000      0
1087
1088      :ITEM 17
1089      :
1090      001452 011006      EM17      :BUS INIT DIDN'T CLEAR REGISTER
1091      001454 011566      DH2       :PC      REGADD      RECVD
1092      001456 011504      DT2       :$ERRPC $REG0      $REG1
1093      001460 000000      0
1094
1095      :ITEM 20
1096      :
1097      001462 011042      EM20      :ADDRESSING ERROR-TRIED TO ADDRESS REG1, GOT REG2
1098      001464 011740      DH20      :PC      REG1      REG2      (REG1) (REG2)
1099      001466 011514      DT20      :$ERRPC $REG0      $REG1      $REG2      $REG3
1100      001470 000000      0
  
```

1102				: ITEM 21	
1103					
1104	001472	011365		EM27	: TRIED TO CLEAR 'REG-BYTE' CHANGED 'REG-BYT2'
1105	001474	012151		DM27	: PC REG-BYT1 REG-BYT2 BYT2-EXPCT BYT2-RECVD
1106	001476	011514		DT20	: SERRPC \$REG0 \$REG1 \$REG2 \$REG3
1107	001500	000000		0	
1108					
1109				: ITEM 22	
1110					
1111	001502	011123		EM22	: DID NOT CLEAR RKCS LO BYTE
1112	001504	011615		DM4	: PC EXPCY RECVD
1113	001506	011504		DT2	: SERRPC \$REG0 \$REG1
1114	001510	000000		0	
1115					
1116				: IEM 23	
1117					
1118	001512	011156		EM23	: DID NOT CLEAR RKCS HI BYTE
1119	001514	011615		DM4	: PC EXPCY RECVD
1120	001516	011504		DT2	: SERRPC \$REG0 \$REG1
1121	001520	000000		0	
1122					
1123				: ITEM 24	
1124					
1125	001522	011212		EM24	: TRIED TO CLEAR RKCS 'BYTE' CHANGED 'REGIS'
1126	001524	012007		DM24	: PC BYTE REGIS (REG)EXP (REG)RECVD
1127	001526	011514		DT20	: SERRPC \$REG0 \$REG1 \$REG2 \$REG3
1128	001530	000000		0	
1129					
1130				: ITEM 25	
1131					
1132	001532	011266		EM25	: FAILED TO CLEAR 'REG-BYTE'
1133	001534	012061		DM25	: PC REG-BYTE RECVD
1134	001536	011504		DT2	: SERRPC \$REG0 \$REG1
1135	001540	000000		0	
1136					
1137					
1138					
1139					
1140	001542	000005		START: RESET	: CLEAR THE BUS
1141				: SBTTL INITIALIZE THE COMMON TAGS	
1142				: : CLEAR THE COMMON TAGS (\$CMTAG) AREA	
1143	001544	012706	001100	MOV #CMTAG,R6	: FIRST LOCATION TO BE CLEARED
1144	001550	005026		CLR (R6)+	: CLEAR MEMORY LOCATION
1145	001552	022706	001140	CMP #SWR,R6	: ;:DONE'
1146	001556	001374		BNE -6	: :LOOP BACK IF NO
1147	001560	012706	001100	MOV #STACK,SP	: :SETUP THE STACK POINTER
1148				: : INITIALIZE A FEW VECTORS	
1149	001564	012737	005642	MOV #SCOPE,%IOTVEC	: : IOT VECTOR FOR SCOPE ROUTINE
1150	001572	012737	000340	MOV #340,%IOTVEC+2	: :LEVEL 7
1151	001600	012737	006114	MOV #ERROR,%EMTVEC	: : EMT VECTOR FOR ERROR ROUTINE
1152	001606	012737	000340	MOV #340,%EMTVEC+2	: :LEVEL 7
1153	001614	012737	010066	MOV #TRAP,%TRAPVEC	: : TRAP VECTOR FOR TRAP CALLS
1154	001622	012737	000340	MOV #340,%TRAPVEC+2	: :LEVEL 7
1155	001630	012737	010154	MOV #PWRON,%PWRVEC	: : POWER FAILURE VECTOR
1156	001636	012737	000340	MOV #340,%PWRVEC+2	: :LEVEL 7
1157	001644	005037	001206	CLR \$TIMES	: : INITIALIZE NUMBER OF ITERATIONS

```

1158 001650 005037 001210          CLR      $ESCAPE          ;;CLEAR THE ESCAPE ON ERROR ADDRESS
1159 001654 112737 000001 001115    MOV      #1,$ERMAX          ;;ALLOW ONE ERROR PER TEST
1160 001662 012737 001662 001106    MOV      #,$SLPADR          ;;INITIALIZE THE LOOP ADDRESS FOR SCOPE
1161 001670 012737 001670 001110    MOV      #,$SLPERR          ;;SETUP THE ERROR LOOP ADDRESS
1162                                     ;;SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
1163                                     ;;EQUAL TO A "-1", SETUP FOR A SOFTWARE SWITCH REGISTER.
1164 001676 013746 000004          MOV      @ERRVEC, -(SP)          ;;SAVE ERROR VECTOR
1165 001702 012737 001736 000004    MOV      #64$, @ERRVEC          ;;SET UP ERROR VECTOR
1166 001710 012737 177570 001140    MOV      @DSWR, SWR          ;;SETUP FOR A HARDWARE SWICH REGISTER
1167 001716 012737 177570 001142    MOV      @DISP, DISPLAY          ;;AND A HARDWARE DISPLAY REGISTER
1168 001724 022777 177777 177206    CMP      #-1, @SWR          ;;TRY TO REFERENCE HARDWARE SWR
1169 001732 001012          BNE      66$          ;;BRANCH IF NO TIMEOUT TRAP OCCURRED
1170                                     ;;AND THE HARDWARE SWR IS NOT = -1
1171 001734 000403          BR       65$          ;;BRANCH IF NO TIMEOUT
1172 001736 012716 001744          64$: MOV      #65$, (SP)          ;;SET UP FOR TRAP RETURN
1173 001742 000002          RTI
1174 001744 012737 000176 001140    65$: MOV      @SWREG, SWR          ;;POINT TO SOFTWARE SWR
1175 001752 012737 000174 001142    MOV      @DISPREG, DISPLAY
1176 001760 012537 000004          66$: MOV      (SP)+, @ERRVEC          ;;RESTORE ERROR VECTOR
1177
1178 001764 023737 000042 000046    CMP      @42, @46          ;;ARE WE IN ACT11 AUTOMATIC MODE?
1179 001772 001416          BEQ      69$          ;;IF YES, SKIP TITLE
1180                                     .SBTTL  TYPE PROGRAM NAME
1181                                     ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
1182 001774 005227 177777          INC      #-1          ;;FIRST TIME?
1183 002000 001043          BNE      67$          ;;BRANCH IF NO
1184 002002 104401 002040          TYPE      68$          ;;TYPE ASCIZ STRING
1185                                     .SBTTL  GET VALUE FOR SOFTWARE SWITCH REGISTER
1186 002006 005737 000042          TST      @42          ;;ARE WE RUNNING UNDER XXDP ACT?
1187 002012 001006          BNE      69$          ;;BRANCH IF YES
1188 002014 023727 001140 000176    CMP      SWR, @SWREG          ;;SOFTWARE SWITCH REG SELECTED?
1189 002022 001005          BNE      70$          ;;BRANCH IF NO
1190 002024 104406          GTSWR          ;;GET SOFT-SWR SETTINGS
1191 002026 000403          BR       70$
1192 002030 112737 000001 001134    69$: MOV      #1, $AUTOB          ;;SET AUTO-MODE INDICATOR
1193 002036          70$:
1194 002036 000424          BR       67$          ;;GET OVER THE ASCIZ
1195                                     ;;68$: .ASCIZ <CRLF>/RK11 LOGIC TEST I/<15><12>/MAINDEC-11-DZKJ-E CRLF
1196 002110          67$:
1197
1198 002110 012737 002126 000004    START1: MOV      @BADTMO, @4          ;;SET TIME OUT VECTOR FOR UNEXPECTED
1199                                     ;;TIME OUTS
1200 002116 012777 002212 177144    MOV      @BADINT, @RKVEC          ;;SET UP RK11 INTERRUPT VECTOR FOR
1201                                     ;;UNEXPECTED INTERRUPTS FROM RK11
1202 002124 000516          BR       TST1          ;;GO TO TEST 1
1203
1204
1205
1206
1207                                     :THIS ROUTINE HANDLES UNEXPECTED TIME OUTS
1208
1209 002116 011600          BADTMO: MOV      (SP), RO          ;;SAVE PC WHERE TIME OUT OCCURED
1210 002131 005740          TST      -(RO)
1211 002132 022626          CMP      (SP)+, (SP)+          ;;RESTORE STACK POINTER
1212 002134 104401 002142          TYPE      65$          ;;TYPE ASCIZ STRING
1213 002140 000417          BR       64$          ;;GET OVER THE ASCIZ

```

```

1214      ;:65$: .ASCIZ <15><12>/UNEXPECTED TIME OUT AT PC=/
1215      64$:
1216      002200      010046      MOV      RO,-(SP)      ;SET UP FOR TYPING OUT PC
1217      002202      104402      TYPOC      ;GO TYPE OUT OCTAL PC
1218      002204      000000      HALT
1219      002206      000137      001542      JMP      @#START
1220
1221      ;THIS ROUTINE HANDLES UNEXPECTED INTERRUPTS FROM RK11
1222      ;SW 9 AND 10 FOR LOOPING ON ERROR
1223      ;AND LOOPING ON TEST IN WHICH TIMEOUT
1224      ;OCCURRED, ARE PROVIDED.
1225
1226      002212      011600      BADINT: MOV      (SP),RO      ;SAVE PC WHERE INTERRUPT OCCURED
1227      002214      005740      TST      -(RO)
1228      002216      032777      020000      176714      BIT      #20000,ASWR      ;INHIBIT ERROR TYPEOUT?
1229      002224      001015      BNE      1$      ;YES, DON'T TYPE OUT
1230      002226      104401      TYPE
1231      002230      001213      $CRLF
1232      002232      104401      TYPE
1233      002234      011443      EM43      ;TYPE 'UNEXPEXED RK11 INTERRUPT'
1234      ;TYPE ' AT PC='
1235      002236      104401      002244      TYPE      65$      ;TYPE ASCIZ STRING
1236      002242      000404      BR      64$      ;GET OVER THE ASCIZ
1237      ;:65$: .ASCIZ / AT PC=/
1238      64$:
1239      002254      010046      MOV      RO,-(SP)      ;SET UP FOR TYPING OUT PC
1240      002256      104402      TYPOC      ;GO TYPE OCTAL PC WHERE BAD
1241      ;INTERUPT OCCURED
1242      002260      032777      001000      176652      1$: BIT      #1000,ASWR      ;LOOP ON ERROR?
1243      002266      001403      BEQ      2$      ;NO, BRANCH
1244      002270      022626      CMP      (SP)+,(SP)+      ;YES, REPOSITION STACK
1245      002272      000177      176610      JMP      @SLPADR      ;GO TO THE STARTING ADDRESS OF
1246      ;THE TEST THAT GAVE UNEXPECTED INTERRUPT
1247      002276      032777      040000      176634      2$: BIT      #40000,ASWR      ;LOOP ON TEST?
1248      002304      001401      BEQ      3$      ;NO, BRANCH
1249      002306      000002      RTI
1250      002310      000000      3$: HALT      ;YES, LOOP. GO BACK WHER U INTERRUPTED FROM.
1251      ;UNEXPECTED INTERRUPT OCCURED AS
1252      ;INDICATED IN THE TYPE OUT.U CAN LOOP
1253      ;ON ERROR, TEST,OR INHIBIT TYPEOUT BY
1254      002312      000137      001542      JMP      @#START      ;SETTING APPROPRIATE SWITCHES.
1255      ;GO BACK TO THE START OF THE
1256      ;PROGRAM. THUS PRESSING CONTINUE
1257      ;AFTER THE ABOVE HALT WILL
1258      ;RESTART THE PROGRAM
1259
1260
1261      ;RESTART AFTER POWER FAIL
1262      ;THE PROGRAM WOULD RESTART HERE IF POWER CAME BACK AFTER A FALIURE.
1263
1264      002316      104401      002324      PFSTRT: TYPE      65$      ;:TYPE ASCIZ STRING
1265      002316      000411      BR      64$      ;:GET OVER THE ASCIZ
1266      002322
1267      ;:65$: .ASCIZ <15><12>/PWR UP,RESTART/
1268      64$:
1269      002346      005000      CLR      RO

```

```

1270 002350 005001
1271 002352 005201
1272 002354 001376
1273 002356 105200
1274 002360 001374
1275
1276
1277
1278
1279
1280
1281
1282
1283
1284 002362 000004
1285 002364 005002
1286 002366 012737 002412 001110
1287
1288 002374 012737 002466 000004
1289 002402 012700 177771
1290 002406 012701 001244
1291 002412 005731
1292
1293
1294 002414 005200
1295 002416 001375
1296 002420 012737 002126 000004
1297 002426 005702
1298 002430 001415
1299 002432 104401 002440
1300 002436 000410
1301
1302 002460
1303 002460 000137 005370
1304 002464
1305 002464 000407
1306 002466 022626
1307 002470 014137 001162
1308 002474 104001
1309
1310 002476 005721
1311 002500 005202
1312 002502 000744
1313
1314
1315
1316
1317
1318
1319
1320
1321 002504 000004
1322 002506 000005
1323 002510 012700 177772
1324 002514 012701 001246
1325 002520 023711 001250

1S: CLR R1
      INC R1
      BNE 1S
      INCB R0
      BNE 1S

*****
;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. IF THERE WAS
;AN ERROR DURING THIS TEST, THE ENTIRE PROGRAM IS ABORTED
*****
TST1: SCOPE
      CLR R2
      MOV #T1,$LPERR ;SET RETURN ADRES FOR LUP
                        ;ON EROR (SW9)
      MOV #TIMOUT,2#4 ;SET UP ADDRESS FOR TIMEOUT VECTOR
      MOV #-7,R0 ;INITIALIZE R0 TO KEEP TRACK OF REGIS REFERENCED
      MOV #RKDS,R1 ;INITIALIZE R1 WITH RKDS ADDRESS
      TST 2(R1)+ ;REFERENCE THE REGISTER
                        ;IF IT CAN'T BE, TIMEOUT TRAP WILL OCCUR
                        ;SHIFT THE POINTER
                        ;ALL REGISTERS REFERENCED?
                        ;IF NOT, LOOP BACK & REFERENCE NEXT REGISTER
      INC R0
      BNE T1
      MOV #BADTMO,2#4
      TST R2 ;WAS THERE AN ERROR?
      BEQ 1S ;NO,BRANCH
      TYPE 65$ ;TYPE ASCIZ STRING
      BR 64$ ;GET OVER THE ASCIZ
65$: .ASCIZ <15><12>/PROG ABORTED/
64$: JMP $GET42 ;IF YES, ABORT THIS ENTIRE TEST
1S: BR TST2 ;EXIT
TIMOUT: CMP (SP)+,(SP)+
      MOV -(R1),$REGO ;GET ADDRESS OF REGISTER THAT TIMED OUT
      ERROR 1 ;TIMED OUT WHEN REFERENCING RK11
                        ;REGISTERS
                        ;REPOSITION POINTER TO THE NEXT REGISTER ADDRESS
      TST (R1)+
      INC R2 ;SET FLAG INDICATING ERROR
      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.
*****
TST2: SCOPE
      RESET ;ISSUE A BUS INIT
      MOV #-6,R0 ;SET COUNT FOR 6 REGISTERS
      MOV #RKER,R1 ;INITIALIZE ADRES
1S: CMP RKCS,R1 ;IS IT RKCS?

```

```

1326 002524 001005      BNE      2$      ;NO
1327 002526 022771 00C200 000000      CMP      #200,2(R1) ;ONLY BIT 7 OF RKCS SHOULD BE SET!
1328 002534 001004      BNE      3$      ;BRANCH IF ERROR
1329 002536 000411      BR       4$
1330 002540 005771 000000      2$: TST      2(R1) ;CHECK THAT REST OF REGISTERS
1331                                ;ARE CLEAR
1332 002544 001406      BEQ      4$      ;BRANCH IF REGISTER IS CLR
1333 002546 011137 001162      3$: MOV      2R1,$REG0 ;GET ADRES OF REGISTER
1334 002552 017137 000000 001164      MOV      2(R1),$REG1 ;GET CONTENTS OF REGISTER
1335 002560 104002      ERROR      2 ;RK11 REGISTER WAS FOUND TO B NOT CLEAR
1336 002562 005721      4$: TST      (R1)+ ;INCREMENT POINTER TO NXT REG
1337 002564 005200      INC      R0 ;CHKD ALL REGS?
1338 002566 001354      BNE      1$      ;IF NOT LUP BAK
1339
1340 ;*****
1341 ;*TEST 3 TEST RKCS FUNCTION BITS - 1,2,3
1342 ;*THIS TEST CHECKS IF THE FUNCTION BIT-1,2,3 IN RKCS CAN BE WRITTEN &
1343 ;*READ BACK. R0 CONTAINS THE BIT THAT WAS WRITTEN. R1 CONTAINS THE
1344 ;*BITS THAT WAS/WERE READ BACK.
1345 ;*****
1346 002570 000004      1$*3: SCOPE
1347 002572 012737 002604 001110      MOV      #1$,$LPERR ;SET RETURN ADRES FOR LOPING ON
1348                                ;ERROR (SW 9)
1349 002600 012700 000002      MOV      #2,R0 ;INITIALIZE BIT TO BE WRITTEN IN RKCS
1350 002604 010077 176440      1$: MOV      R0,2RKCS ;WRITE THAT BIT IN RKCS
1351 002610 017701 176434      MOV      2RKCS,R1 ;GET RKCS
1352 002614 042701 000200      BIC      #200,R1 ;MASK OUT CNTRL RDY BIT
1353 002620 020001      CMP      R0,R1 ;WAS RECVD BIT SAME AS WRITTEN BIT
1354 002622 001406      BEQ      2$      ;YES, BRANCH OTHERWISE REPORT ERROR
1355 002624 010037 001162      MOV      R0,$REG0 ;GET EXPCD RKCS, BIT THAT WAS WRITTEN
1356 002630 017737 176414 001164      MOV      2RKCS,$REG1 ;GET RECVD RKCS, BIT READ
1357 002636 104003      ERROR      3 ;BIT THAT WAS WRITTEN WAS NOT READ BACK
1358 002640 006300      2$: ASL      R0 ;SHIFT TO WRITE NEXT BIT
1359 002642 020027 000020      CMP      R0,#20 ;HAVE U CHKD ALL 3 BITS 1, 2, 3
1360 002646 001356      BNE      1$      ;IF NOT, LOOP BACK & CHECK THE NXT BIT
1361
1362 ;*****
1363 ;*TEST 4 TEST RKCS EXTENDED MEMORY BITS - 4,5
1364 ;*THIS TEST CHECKS IF THE 'EXTENDED MEMORY' BITS CAN BE WRITTEN
1365 ;*AND READ BACK CORRECTLY.
1366 ;*****
1367 002650 000004      1$*4: SCOPE
1368 002652 012737 002664 001110      MOV      #1$,$LPERR ;SET RETURN ADRES FOR LUPING
1369                                ;ON EROR (SW 9)
1370 002660 012700 000020      MOV      #20,R0 ;INITIALIZE BIT TO BE WRITTEN IN RKCS
1371 002664 010077 176360      1$: MOV      R0,2RKCS ;WRITE THAT BIT IN RKCS
1372 002670 017701 176354      MOV      2RKCS,R1 ;GET RKCS
1373 002674 042701 000200      BIC      #200,R1 ;MASK OUT CNTRL RDY BIT
1374 002700 020001      CMP      R0,R1 ;WAS RECVD BIT SAME AS WRITTEN BIT
1375 002702 001406      BEQ      2$      ;YES, BRANCH
1376 002704 010037 001162      MOV      R0,$REG0 ;GET EXPCD RKCS, BIT THAT WAS WRITTEN
1377 002710 017737 176334 001164      MOV      2RKCS,$REG1 ;GET RKCD RECVD, BIT THAT WAS READ
1378 002716 104003      ERROR      3 ;BIT THAT WAS WRITTEN WAS NOT READ BACK
1379 002720 006300      2$: ASL      R0 ;SHIFT TO WRITE NEXT BIT
1380 002722 022700 000100      CMP      #100,R0 ;HAVE U CHKD BOTH BITS, 4, 5
1381 002726 001356      BNE      1$      ;IF NOT, LOOP BACK & CHECK THE NXT BIT

```



```

1382
1383
1384
1385
1386
1387
1388
1389
1390 002730 000004
1391 002732 012746 000340
1392 002736 012746 002744
1393 002742 000002
1394 002744
1395 002744 013700 001250
1396 002750 012710 000100
1397 002754 022710 000300
1398 002760 001406
1399 002762 012737 000300 001162
1400 002770 011937 001164
1401 002774 104003
1402
1403 002776 104412
1404
1405
1406
1407
1408
1409
1410
1411 003000 022710 000200
1412 003004 001406
1413 003006 010037 001162
1414 003012 011937 001164
1415 003016 104010
1416 003020 000430
1417 003022 104414 000010
1418
1419 003026 012777 003076 176234
1420
1421
1422 003034 013746 001266
1423 003040 012746 003046
1424 003044 000002
1425 003046 000240
1426 003050 000240
1427 003052 000240
1428 003054 012746 000340
1429 003060 012746 003066
1430 003064 000002
1431 003066
1432
1433 003066 012777 002212 176174
1434 003074 000402
1435 003076 022626
1436 003100 104012

```

```

*****
*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.
*****
ST5:  SCOPE
      MOV     #340, -(SP)
      MOV     #645, -(SP)
      RTI

645:  MOV     RKCS, R0
      MOV     #100, R0
      CMP     #300, R0
      BEQ     IS
      MOV     #300, $REG0
      MOV     R0, $REG1
      ERROR   3

IS:   CNT.RESET

      CMP     #200, R0
      BEQ     25
      MOV     R0, $REG0
      MOV     R0, $REG1
      ERROR   10
      BR      TS6
      DELAY   .10

25:   MOV     #35, R0KVEC

      MOV     RKPRI, -(SP)
      MOV     #45, -(SP)
      RTI

45:   NOP
      NOP
      NOP
      MOV     #340, -(SP)
      MOV     #655, -(SP)
      RTI

655:

      MOV     #BADINT, R0KVEC
      BR      TS6
      CMP     (SP), (SP)
      ERROR   12

```

```

:SET THE IDE BIT
:WAS IT WRITTEN CORRECTLY
:YES, BRANCH OTHERWISE REPORT ERROR
:GET EXPTD RKCS
:GET RKCS RECVD
:IDE BIT WAS WRITTEN, BUT WAS NOT
:READ BACK
: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 R0Y' DOES NOT SET
:AN ERROR IS REPORTED.  NOTE THAT
:THE PC IN ERROR MESSAGE IS THE
:PC WHERE 'CNT.RESET' IS LOCATED.
:DID IDE BIT GET CLRD?
:YES, BRANCH
:GET ADRES OF RKCS
:GET RKCS
:IDE BIT COULD NOT B CLRD
:EXIT
:WAIT FOR AT LEAST 60 US
:ON 11/20 12 US FOR 11/45
:SET RK11 INTERRUPT VECTOR TO
:WHICH RK11 CAN INTERRUPT IF THERE
:IS FAULTY LOGIC
:LOWER CPU PRIORITY SO THAT
:RK11 CAN POSSIBLY INTERRUPT IF
:THER IS MALFUNCTIONING LOGIC
:THE INTERRUPT WOULD OCCUR
:PRIORITY
:SET UP UNEXPTD INTRUPT VECTOR
:EXIT
:RESTORE STACK
:AN UNEXPECTED RK11 INTERRUPT
:OCCURED PROBABLY DUE TO FAULTY LOGIC

```

```

438
439
440
441
442
443
444
445 003102 000004
446 003104 012701 000400
447 003110 013702 001250
448 003114 010112
449 003116 011200
450 003120 042700 000200
451 003124 020100
452
453 003126 001405
454 003130 010137 001162
455 003134 011237 001164
456 003140 104003
457
458 003142 006301
459 003144 022701 010000
460 003150 001361
461
462
463
464
465
466
467
468 003152 000004
469 003154 013700 001250
470 003160 012710 170200
471
472 003164 011001
473 003166 042701 010000
474 003172 022701 000200
475
476 003176 001406
477 003200 012737 000200 001162
478 003206 011037 001164
479 003212 104004
480
481
482
483
484
485
486
487
488
489
490 003214 000004
491 003216 012746 000340
492 003222 012746 003230
493 003226 000002
  
```

```

*****
*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
*****
+ST6:  SCOPE
      MOV     #400, R1      ; INITIALIZE BIT TO BE WRITTEN IN RKCS
      MOV     RKCS, R2
15:    MOV     R1, R2       ; WRITE THAT BIT IN RKCS
      MOV     R2, R0       ; GET RKCS
      BIC     #200, R0     ; MASK CNTRL RDY BIT
      CMP     R1, R0       ; WAS THE READ BIT SAME AS THE
                          ; WRITTEN BIT
      BEQ     25           ; YES BRANCH, OTHERWISE REPORT ERROR
      MOV     R1, $REG0    ; GET EXPTD RKCS
      MOV     R2, $REG1    ; GET RECVD RKCS
      ERROR   3           ; BIT THAT WAS WRITTEN AS IN $REG0
                          ; WAS NOT READ BACK
25:    ASL     R1          ; SHIFT TO WRITE NEXT BIT
      CMP     #10000, R1   ; HAVE U CHECKED ALL BITS 8, 9, 10, 11
      BNE     15          ; IF NOT, LOOP BACK & CHECK THE NEXT BIT
*****
*TEST 7      CHECK READ ONLY BITS OF RKCS
*THIS TEST CHECKS THAT TRYING TO SET THE UNUSED BIT OF THE READ ONLY
*BITS DOES NOT SET THEM OR AFFECT ANY OTHER BITS IN RKCS
*****
+ST7:  SCOPE
      MOV     RKCS, R0
      MOV     #170200, R0  ; TRY SETTING THE UNUSED BIT & RD
                          ; ONLY BITS
      MOV     R0, R1       ; GET RKCS
      BIC     #10000, R1   ; MASK BIT 12
      CMP     #200, R1     ; IS 'RDY' BIT SET? NO OTHER
                          ; BIT SHOULD BE SET.
      BEQ     TST10        ; OK, EXIT
      MOV     #200, $REG0  ; GET EXPTD RKCS
      MOV     R0, $REG1    ; GET RKCS RECVD
      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.
*****
+ST10: SCOPE
      MOV     #340, -(SP)
      MOV     #645, -(SP)
      RTI
  
```



```

1494 003230 013701 001250
1495 003230 012711 007576
1496 003234 012711 007576
1497 003240 005000
1498 003242 000005
1499 003244 022711 000200
1500 003250 001403
1501 003252 011137 001162
1502 003256 104005
1503 003260 012711 005000
1504 003264 005211
1505 003266 005200
1506 003270 105700
1507
1508 003272 100411
1509 003274 105711
1510 003276 100373
1511 003300 022711 000200
1512
1513 003304 001407
1514 003306 011137 001162
1515 003312 104006
1516
1517 003314 000403
1518 003316 004737 005434
1519 003322 104007
1520
1521
1522
1523
1524
1525
1526
1527
1528
1529 003324 000004
1530 003326 012737 000010 001206
1531 003334 012746 000340
1532 003340 012746 003346
1533 003344 000002
1534 003346
1535 003346 013700 001250
1536 003352 012710 007576
1537 003356 005010
1538 003360 022710 000200
1539 003364 001405
1540 003366 010037 001162
1541 003372 011037 001164
1542 003376 104010
1543 003400 012701 000002
1544 003404 012737 003416 001110
1545 003412 012705 177773
1546 003416 010110
1547 003420 010102
1548 003422 052702 000200
1549 003426 011003

645:
MOV RKCS,R1
MOV #7576,R1
CLR R0
RESET
CMP #200,R1
BEQ 1$
MOV R1,$REG0
ERROR 5
1$: MOV #5000,R1
INC R1
2$: INC R0
TSTB R0
BMI 3$
TSTB R1
BPL 2$
CMP #200,R1
BEQ TST11
MOV R1,$REG0
ERROR 6
BR TST11
3$: JSR FCT3RG
ERROR 7

;SET ALL BITS EXCEPT GO
;ISSUE BUS INIT
;CHECK IF RKCS WAS CLEARED?
;YES, BRANCH OTHERWISE REPORT ERROR
;GET RKCS
;BUS INIT DID NOT CLEAR RKCS
;SET IBA & EXB IN RKCS
;SET GO, CONTROL RESET
;KEEP TIME
;HAVE U WAITED LONG FOR CNTRL RDY
;TO SET?
;IF YES, BRANCH & REPORT ERROR
;WAS CNTRL RDY SET?
;IF NOT LOOP BACK & WAIT FOR IT
;IF CNTRL RDY WAS SET, CHK IF 'CNTRL
;RESET' CLEARED IBA & EXB BITS
;IF YES, EXIT. OTHERWISE ERROR
;GET RKCS
;GO BIT COULD NOT BE SET OR FAULT IN
;THE 'INIT L' GENERATING LOGIC
;EXIT
;GO, GET RKCS, ER, DS
;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 UNDESIRE OPERATION.
;R1 CONTAINS THE COUNT PATTERN THAT WAS WRITTEN
;*****

TST11: SCOPE
MOV #10,$TIMES ;DO 10 ITERATIONS
MOV #340,-(SP)
MOV #645,-(SP)
RTI

645:
MOV RKCS,R0
MOV #7576,R0
CLR R0
CMP #200,R0
BEQ 1$
MOV R0,$REG0
MOV R0,$REG1
ERROR 10
1$: MOV #2,R1
MOV #2,$SLPERR
MOV #5,R5
2$: MOV R1,R0
MOV R1,R2
BIS #200,R2
MOV R0,R3

;SET ALL BITS IN RKCS EXCEPT GO
;CLEAR RKCS
;WAS IT CLEARED
;YES, BRANCH
;GET ADRES OF RKCS
;NO, GET RKCS
;RKCS COULD NOT BE CLEARED
;WRITE THIS BIT IN RKCS

;WRITE IT
;GET BIT THAT WAS WRITTEN
;SET CNTRL RDY BIT

```

```

1550 003430 020203      CMP      R2,R3      ;WAS THAT BIT WRITTEN CORRECTLY?
1551 003432 001407      BEQ      3$      ;YES, BRANCH
1552 003434 010237 001162  MOV      R2,$REGO    ;GET EXPTD WORD
1553 003440 010337 001164  MOV      R3,$REG1    ;GET RKCS RECVD
1554 003444 104003      ERROR    3      ;DID NOT READ BAK THE BIT THAT
1555                                     ;WAS WRITTEN
1556 003446 005205      INC      R5
1557 003450 001405      BEQ      TST12      ;:EXIT
1558 003452 062701 000002 3$:      ADD      #2,R1      ;GENERATE NXT PATTERN TO BE WRITTEN
1559 003456 022701 010000      CMP      #10000,R1    ;ALL PATTERNS WRITTEN?
1560 003462 001355      BNE      2$      ;IF NOT, LUP BAK & CHK NXT PATTERN
1561                                     ;
1562                                     ;*****
1563                                     ;*TEST 12      CHECK THAT RKWC BIT 0-15 CAN BE SET
1564                                     ;*THIS TEST FLOATS A '1' THROUGH RKWC BIT 0-15 AND CHECKS THAT IT
1565                                     ;*CAN BE READ BACK CORRECTLY. R0 CONTAINS THE WORD THAT IS WRITTEN.
1566                                     ;*****
1567 003464 000004      *TST12: SCOPE
1568 003466 012700 000001      MOV      #1,R0      ;INITIALIZE R0 FOR THE BIT TO BE WRITTEN IN RKWC
1569 003472 013701 001252      MOV      RKWC,R1
1570 003476 012737 003504 001110  MOV      #15,$LPERR    ;SET UP RETURN ADRES FOR
1571                                     ;LUPING ON ERROR (SW 9)
1572 003504 010011      1$:      MOV      R0,R1      ;WRITE THAT BIT IN RKWC
1573 003506 011102      MOV      R1,R2
1574 003510 020002      CMP      R0,R2
1575 003512 001405      BEQ      2$      ;WAS IT WRITTEN CORRECTLY
1576 003514 010037 001162      MOV      R0,$REGO    ;YES, BRANCH. OTHERWISE, ERROR
1577 003520 010237 001164      MOV      R2,$REG1    ;GET EXPTD RKWC BIT THAT WAS WRITTEN
1578 003524 104011      ERROR    11      ;GET RKWC (THAT WAS READ BACK)
1579                                     ;DID NOT READ BACK THE BIT THAT
1580                                     ;WAS WRITTEN IN RKWC
1581 003526 006300      2$:      ASL      R0      ;SHIFT TO WRITE THE NXT BIT
1582 003530 001365      BNE      1$      ;IF ALL THE BITS HAVE NOT BEEN
1583                                     ;DONE, LOOP BACK
1584                                     ;
1585                                     ;*****
1586                                     ;*TEST 13      CHECK RKWC WITH A COUNT PATTERN
1587                                     ;*THIS TEST CHECKS THAT RKWC CAN BE CLEARED, THEN A COUNT PATTERN
1588                                     ;*FROM 0 TO 177777 IS WRITTEN & CHECKED IF IT WAS WRITTEN CORRECTLY.
1589                                     ;*****
1590 003532 000004      *TST13: SCOPE
1591 003534 012737 000010 001206      MOV      #10,$TIMES    ;:DO 10 ITERATIONS
1592 003542 013700 001252      MOV      RKWC,R0
1593 003546 012710 177777      MOV      #177777,$R0    ;SET ALL BITS IN RKWC
1594 003552 005010      CLR      $R0      ;CLEAR RKWC
1595 003554 005710      TST      $R0      ;WAS IT CLEARED?
1596 003556 001405      BEQ      1$      ;YES, BRANCH
1597 003560 010037 001162      MOV      R0,$REGO    ;GET ADRES OF RKWC
1598 003564 011037 001164      MOV      $R0,$REG1    ;NO, GET RKWC
1599 003570 104010      ERROR    10      ;RKWC COULD NOT BE CLEARED
1600 003572 005001      1$:      CLR      R1      ;INITIALIZE COUNT PATTERN
1601 003574 012705 177773      MOV      #-5,R5
1602 003600 012737 003606 001110  MOV      #25,$LPERR
1603 003606 010110      2$:      MOV      R1,$R0      ;WRITE THE PATTERN IN THE REGISTER
1604 003610 011002      MOV      $R0,R2
1605 003612 020102      CMP      R1,R2      ;WAS IT WRITTEN CORRECTLY?

```

```

1606 003614 001407      BEQ      3$      :YES, BRANCH
1607 003616 010137 001152  MOV      R1,$REG0  :GET EXPECTED WORD
1608 003622 010237 001154  MOV      R2,$REG1  :GET WORD THAT WAS REC'D
1609 003626 104011      ERROR    11  :DID NOT READ BACK THE PATTERN THAT
1610                                     :WAS WRITTEN INTO THE REGISTER
1611 003630 005205      INC      R5
1612 003632 001402      BEQ      *ST14  :EXIT
1613 003634 005201 3$: INC      R1      :INCREMENT COUNT PATTERN
1614 003636 001363      BNE      2$      :LUP BAK & WRITE NXT PATTERN IF NOT
1615                                     :DONE WITH ALL
1616
1617 :*****
1618 :*TEST 14 CHECK THAT RKBA CAN BE SET
1619 :*THIS TEST FLOATS A '1' THROUGH RKBA BITS 0-15 AND CHECKS THAT
1620 :*IT CAN BE READ BACK CORRECTLY. RO CONTAINS THE WORD THAT WAS
1621 :*WRITTEN.
1622 :*****
1623 003640 000004 1$ST14: SCOPE
1624 003642 012700 000001  MOV      #1,RO      :INITIALIZE RO FOR BIT TO BE
1625                                     :WRITTEN IN RKBA
1626 003646 013701 001254  MOV      RKBA,R1
1627 003652 012737 003660 001110  MOV      #15,$LPERF  :SET UP RETURN ADRES FOR
1628                                     :LUPING ON EROR (SW 12)
1629 003660 010011 1$: MOV      RO,R1      :WRITE THAT BIT IN RKBA
1630 003662 011102      MOV      R1,R2
1631 003664 020002      CMP      RO,R2      :WAS IT WRITTEN CORRECTLY?
1632 003666 001405      BEQ      2$      :YES, BRANCH
1633 003670 010037 001162  MOV      RO,$REG0  :GET EXPCD RKBA (BIT WRITTEN)
1634 003674 010237 001154  MOV      R2,$REG1  :GET RKBA (BIT READ BACK)
1635 003700 104013      ERROR    13  :DID NOT READ BACK THE BIT THAT
1636                                     :WAS WRITTEN IN RKBA
1637 003702 006300 2$: ASL      RO      :SHIFT RO TO WRITE NEXT BIT
1638 003704 001365      BNE      1$      :IF ALL BITS ARE NOT CHKD, LOOP
1639                                     :BACK & WRITE NXT BIT
1640
1641 :*****
1642 :*TEST 15 CHECK RKBA WITH A COUNT PATTERN
1643 :*THIS TEST CHECKS THAT RKBA CAN BE CLEARED, THEN IT RUNS A COUNT
1644 :*PATTERN FROM 0 TO 177777. R1 CONTAINS THE COUNT PATTERN TO BE WRITTEN.
1645 :*****
1646 003706 000004 1$ST15: SCOPE
1647 003710 012737 000010 001206  MOV      #10,$TIMES  :DO 10 ITERATIONS
1648 003716 013700 001254  MOV      RKBA,RO
1649 003722 012710 177777  MOV      #177777,$RO  :SET ALL BITS IN RKBA
1650 003726 005010      CLR      $RO      :CLEAR RKBA
1651 003730 005710      TST      $RO      :WAS IT CLEARED?
1652 003732 001405      BEQ      1$      :YES, BRANCH
1653 003734 010037 001162  MOV      RO,$REG0  :GET ADRES OF RKBA
1654 003740 011037 001154  MOV      $RO,$REG1  :NO, GET RKBA
1655 003744 104010      ERROR    10  :RKBA COULD NOT BE CLEARED
1656
1657 003746 005001 1$: CLP      R1      :INITIALIZE COUNT PATTERN
1658 003750 012705 177773  MOV      #-5,R5
1659 003754 012737 003762 001110  MOV      #25,$LPERF
1660 003762 010110 2$: MOV      R1,$RO      :WRITE THE PATTERN IN THE
1661                                     :REGISTER

```

```

003764 011002      MOV      @RC,R2
003766 020102      CMP      R1,R2      ;WAS IT WRITTEN CORRECTLY
003768 001407      BEQ      3$      ;YES, BRANCH
003770 010137      MOV      R1,$REG0    ;GET EXPECTED WORD
003772 011037      MOV      @R0,$REG1    ;GET WORD THAT WAS RECVD
003774 104013      ERROR      13      ;DID NOT READ BACK THE PATTERN
004002 104013      ;WAS WRITTEN INTO THE REGISTER

004004 005205      INC      R5
004006 001402      BEQ      TS*16      ;EXIT
004008 005205      INC      R1      ;INCREMENT COUNT PATTERN
004010 001363      BNE      2$      ;LUP BAK & WRITE NXT PATTERN IF NOT
004012 001363      ;DONE WITH ALL

;*****
;*TEST 16      CHECK THAT RKDA CAN BE SET
;*THIS TEST FLOATS A '1' THROUGH RKDA AND CHECKS THAT THE WORD WAS
;*WRITTEN IS READ BACK CORRECTLY
;*****
+ST16:  SCOPE
004014 000004      CLR      @RKCS      ;CLEAR RKCS
004016 005077      MOV      #1,R0      ;INITIALIZE R0 FOR BIT TO BE WRITTEN IN RKDA
004022 012700      MOV      RKDA,R1
004026 013701      MOV      #15,$LPERF    ;SET UP RETURN ADRES
004032 012737      MOV      ;FOR LUPING ON EROR
004034 010011      1$:  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      MOV      R0,$REG0    ;NO, GET EXPCD RKDA (BIT WRITTEN)
004054 010237      MOV      R2,$REG1    ;GET RKDA (BIT READ BACK)
004060 104015      ERROR      15      ;DID NOT READ BACK THE BIT THAT
004062 006300      2$:  ASL      R0      ;WAS WRITTEN IN RKDA
004064 001365      BNE      1$      ;SHIFT R0, TOWRITE NXT BIT
004066 006300      ;IF ALL BITS ARE NOT CHKD, LOOP
004068 001365      ;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
;*****
+ST17:  SCOPE
004066 000004      MOV      #10,$TIMES      ;DO 10 ITERATIONS
004070 012737      MOV      RKDA,R0
004076 013700      MOV      #177777,@R0    ;SET ALL BITS IN RKDA
004102 012710      CLR      @R0      ;CLEAR RKDA
004106 005010      TST      @R0      ;WAS IT CLEARED?
004110 005710      BEQ      1$      ;YES, BRANCH
004112 001405      MOV      R0,$REG0    ;GET ADRES OF RKDA
004114 010037      MOV      @R0,$REG1    ;GO, GET RKDA
004120 011037      ERROR      10      ;RKDA COULD NOT BE CLEARED
004124 104010      1$:  CLR      R1      ;INITIALIZE COUNT PATTERN
004126 005001      MOV      #5,R5
004130 012705      MOV      #25,$LPERF
004134 012737      MOV

```

```

2$: MOV R1, QRO ;WRITE THE PATTERN IN THE REGISTER
   MOV QRO, R2
   CMP R1, R2 ;WAS IT WRITTEN CORRECTLY?
   BEQ 3$ ;YES, BRANCH
   MOV R1, $REGO ;GET EXPECTED WORD
   MOV R2, $REG1 ;GET WORD THAT WAS RECVD
   ERROR 15 ;DID NOT READ BACK THE PATTERN THAT
               ;WAS WRITTEN INTO THE REGISTER

   INC R5
   BEQ TST20 ;EXIT
3$: INC R1 ;INCREMENT COUNT PATTERN
   BNE 2$ ;LOOP BACK & 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
;*****

```

```

;ST20: SCOPE
   MOV RKWC, R0 ;INITIALIZE R0 TO POINT TO RKWC
   MOV R0, R2
   MOV #3, R1 ;SET UP COUNT FOR 3 REGISTERS TO BE CHKD
   MOV R1, R3
1$: MOV #177777, (R0,+) ;SET ALL BITS IN RKWC
   INC R1 ; RKBA
   BNE 1$ ; RKDA
   RESET ;ISSUE A BUS INIT
2$: TST (R2) ;WAS THE REGISTER (PTD TO BY R2) CLEARED?
   BEQ 3$ ;YES, BRANCH
   MOV R2, $REGO ;NO, GET ADRES OF REG IS THAT WAS NOT CLEARED
   MOV QRO, $REG1 ;GET CONTENTS OF THAT REGISTER
   ERROR 17 ;RK11 REGISTER (ADRES IN R0) COULD
               ;NOT BE CLEARED BY BUS INIT
3$: TST (R2)+ ;INCREMENT POINTER TO NXT REGISTER
   INC R3 ;HAVE U CHKD ALL 3 REGISTERS?
   BNE 2$ ;IF NOT, LOOP BACK & CHK NXT

```

```

;*****
;*TEST 21 CHECK THAT 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.
;*****

```

```

;ST21: SCOPE
   MOV #340, -(SP)
   MOV #64$, -(SP)
   RTI
64$: MOV RKCS, R0 ;SET ALL WRITABLE BITS IN R0
   MOV #7560, QRO ;SET GO, CONTROL RESET
   INC QRO
   CLR R5
1$: TSTB QRO ;DID CNTRL RDY SET?
   BMI 2$ ;YES, BRANCH
   INC R5 ;WAITED LONG?

```

004174 000004
 004176 013700 001252
 004202 010002
 004204 012701 177775
 004210 010103
 004212 012720 177777
 004216 005201
 004220 001374
 004222 000005
 004224 005712
 004226 001405
 004230 010237 001162
 004234 011237 001164
 004240 104017
 004242 005722
 004244 005203
 004246 001366
 004250 000004
 004252 012746 000340
 004256 012746 004264
 004262 000002
 004264
 004264 013700 001250
 004270 012710 007560
 004274 005210
 004276 005005
 004300 105710
 004302 100405
 004304 005205

```

1774 004306 001374      BNE 1$      :IF NOT LUP BAK & WAIT
1775 004310 004737      JSR PC,GT3RG :GET RKCS,ER,DS
1776 004314 104007      ERROR 7      :CNTRL RDY DID NOT SET
1777      :AFTER CNTRL RESET
1778 004316 022710 2$: CMP #200,R0 :DID CNTRL RESET CLEAR RKCS
1779 004322 001405      BEQ 3$      :YES, BRANCH
1780 004324 010037      MOV R0,$REG0 :GET ADRES OF RKCS
1781 004330 011037      MOV R0,$REG1 :GET CONTENTS OF RKCS
1782 004334 104014      ERROR 14     :CONTROL RESET DID NOT CLEAR RKCS
1783 004336 013702 3$: MOV RKWC,R2
1784 004342 010204      MOV R2,R4
1785 004344 012701 177777 MOV #177777,R1
1786 004350 010122      MOV R1,(R2)+ :SET ALL BITS IN RKWC
1787 004352 010122      MOV R1,(R2)+ :RKBA
1788 004354 010122      MOV R1,(R2)+ :RKDA
1789 004356 104412      CNT.RESET :GO, DO CONTROL RESET
1790      :THIS IS A CALL FOR THE 'CNTRL-
1791      :RESET' ROUTINE. A CONTROL RESET IS
1792      :DONE & AFTER A CERTAIN TIME IF
1793      :'CNTRL RDY' DOES NOT SET AN ERROR IS
1794      :REPORTED. NOTE THAT THE PC IN ERROR
1795      :IS THE PC WHERE CNT.RESET IS
1796      :LOCATED. THIS IS A VERY BASIC ERROR &
1797      :IF IT OCCURS GO BACK TO TEST 10.
1798 004360 012703 177775 MOV #-3,R3
1799 004364 005714 4$: TST (R4) :WAS THE REGISTER CLEARED?
1800 004366 001405      BEQ 5$      :YES, BRANCH
1801 004370 010437      MOV R4,$REG0 :GET ADRES OF REGISTER IN ERROR
1802 004374 011437      MOV R4,$REG1 :GET CONTENTS OF THAT REGISTER
1803 004400 104014      ERROR 14     :CONTROL RESET DID NOT CLEAR THE
1804      :REGISTER WHOOSE ADRES IS IN R4
1805 004402 005724 5$: TST (R4)+ :INCREMENT POINTER TO NXT REGISTER
1806 004404 005203      INC R3      :CHKD ALL REGS?
1807 004406 001366      BNE 4$      :IF NOT, LUP BAK & CHK THE NXT REG
1808
1809 *****
1810 :*TEST 22 CHECK THAT EACH RK11 REGISTER IS UNIQUELY ADRESSED
1811 :*THIS TEST CHECKS THAT EACH RK11 REGISTER CAN BE UNIQUELY ADRESSED
1812 :*1) RKCS RKWC, RKBA, RKDA ARE FIRST SET TO 177777 (17576 FOR RKCS)
1813 :*2) REGISTER WHOOSE ADDRESS IS IN R0 IS CLEARED
1814 :*3) EVERY OTHER REGISTER IS CHECKED FOR ERRONEOUS CLEARING BECAUSE OF
1815 :*ADDRESSING ERROR. IF SO THE MULTIPLE ADDRESSING ERROR IS REPORTED
1816 :*4) ADDRESS IN R0 IS CHANGED TO THE NEXT REGISTER & THE PROCESS IS
1817 :* REPEATED.
1818 *****
1819 :*ST22: SCOPE
1820 MOV #340,-(SP)
1821 MOV #64$,-(SP)
1822 RTI
1823
1824 004424 012705 64$: MOV #-7,R5 :SET UP COUNT FOR THE # OF
1825 :REGISTERS TO BE UNIQUELY ADDRSD
1826 004430 013700 1$: MOV RKDS,R0 :INITIALIZE POINTER TO REGIS TO BE SET
1827 004434 012701      MOV #-4,R1
1828 004440 013702      MOV RKCS,R2
1829 004444 012722      MOV #17576,(R2)+ :SET BITS IN RKCS

```



```

1830 004450 012722 177777 2$: MOV #177777,(R2)+ ;SET BITS IN RKWC
1831 004454 005201 INC R1 ;RKBA
1832 004456 001374 BNE 2$ ;RKDA
1833 ;RKMR IF RK11C
1834 004460 005010 CLR R0 ;CLEAR REGISTER (WHOSE ADDR IS IN R0)
1835 004462 013732 001250 MOV RKCS,R2
1836 004466 020002 CMP R0,R2 ;WAS THE CLEARED REGISTER RKCS?
1837 004470 001406 BEQ 3$ ;YES
1838 ;NO-CHECK IF IT WAS INADVERTENTLY
1839 004472 011203 MOV R2,R3 ;CLEARED BECAUSE OF ADDRESSING
1840 004474 042703 170200 BIC #170200,R3 ;ERROR
1841 004500 022703 007576 CMP #7576,R3
1842 004504 001020 BNE 6$ ;IF SO, REPORT ERROR
1843 004506 005722 3$: TST (R2)+ ;INCREMENT POINTER TO NEXT REGISTER
1844 004510 012701 177775 MOV #-3,R1 ;SET COUNT FOR 3 REGISTERS
1845 004514 020200 4$: CMP R2,R0 ;WAS THE CLEARED REGISTER SAME AS
1846 ;THE REGISTER POINTED TO BY R2
1847 004516 001404 BEQ 5$ ;YES
1848 ;NO, CHECK IF THE REGIS UNDER TEST
1849 ; (POINTED BY R2) WAS INADVERTENTLY
1850 ; CLEARED WHEN CLEARING THE REGIS
1851 ; POINTED TO BY R0, DUE TO
1852 ; ADDRESSING ERROR
1853 004520 011203 MOV R2,R3 ;GET CONTENTS OF REGIS BEING CHECKED
1854 004522 010304 MOV R3,R4 ;FOR INADVERTENT CLEARING
1855 004524 005104 COM R4 ;CHECK IF ANY BIT WAS ERRORNEOUSLY CLEARED
1856 004526 001007 BNE 6$ ;IF SO, REPORT ERROR
1857 004530 005722 5$: TST (R2)+ ;INCRMENT PTR TO NEXT REGISTER
1858 004532 005201 INC R1 ;INCRMENT COUNT
1859 004534 001367 BNE 4$ ;CHECK THE REST
1860 ;
1861 004536 005720 TST (R0)+ ;INCREMENT PTR TO THE NXT REGIS TO BE CLEARED
1862 004540 005205 INC R0 ;HAVE ALL THE REGIS BEEN CHECKED?
1863 004542 001334 BNE 1$ ;IF NOT, LOOP BACK
1864 004544 000415 BR TST23 ;EXIT, IF DONE
1865 004546 010037 001162 6$: MOV R0,$REG0 ;GET ADRES OF REGISTER THAT WAS
1866 ;TRIED TO REFERENCE
1867 004552 010237 001164 MOV R2,$REG1 ;GET ADRES OF REGISTER THAT GOT
1868 ;REFERENCED INSTEAD
1869 004556 011037 001166 MOV R0,$REG2 ;GET CONTENTS OF REG THAT WAS
1870 ;ADDRESSED & MEANT TO BE CLEARED
1871 004562 010337 001170 MOV R3,$REG3 ;GET CONTENTS OF REGISTER THAT GOT
1872 ;CHANGED INSTEAD
1873 004566 104020 ERROR 20 ;POSSIBLE ADRESING ERROR. TRIED
1874 ;ADRESING RK11 REGISTER, ANOTHER ONE
1875 ;GOT ADRESED. REGIS IN R0 WAS THE
1876 ;ADRESED ONE, REG IN R2
1877 ;WAS THE ONE THAT GOT ADRESED INSTEAD
1878 004570 023702 001250 CMP RKCS,R2
1879 004574 001744 BEQ 3$ ;RETURN TO THE
1880 004576 000754 BR 5$ ;RIGHT POINT
1881 ;*****
1882 ;*TEST 23 CHECK THAT HI & LO BYTES OF RKCS CAN BE ADRESSED
1883 ;*THIS TEST CHECKS THAT THE HI & LO BYTES OF RKCS CAN BE
1884 ;*ADRESSED CORRECTLY.
1885 ;*BITS IN ALL REGISTERS THAT CAN BE WRITTEN ARE SET. THEN EACH

```

```

1886 ;*BYTE OF RKCS IS REFERENCED BY 'CLRB' & IT IS CHECKED THAT ONLY
1887 ;*THAT BYTE & NO OTHER REGISTER BYTE GETS CLEARED
1888 .....
1889 004600 000004 TST23: SCOPE
1890 004602 012702 177776 MOV #-2,R2 ;SET COUNT FOR 2BYTES-HI & LO
1891 004606 012700 177775 MOV #-3,R0 ;SET COUNT
1892 004612 012746 000340 MOV #340,-(SP)
1893 004616 012746 004624 MOV #645,-(SP)
1894 004622 000002 RTI
1895 004624 645:
1896 004624 013701 001250 MOV RKCS,R1 ;INITIALIZE PTR. TO RKCS
1897 004630 012721 007576 MOV #7576,(R1)+ ;SET ALL BITS IN RKCS
1898 004634 012721 177777 15: MOV #177777,(R1)+ ;SET ALL BITS IN RKWC
1899 004640 005200 INC R0 ; RKBA
1900 004642 001374 BNE 15 ; RKDA
1901 004644 013701 001250 MOV RKCS,R1 ;INITIALIZE PTR TO RKCS LO BYTE
1902 004650 105011 25: CLRB R1 ;CLR RKCS LO OR HI BYTE
1903 004652 010104 MOV R1,R4
1904 004654 017703 174370 MOV #RKCS,R3 ;GET RKCS WORD
1905 004660 022702 177776 CMP #-2,R2 ;R U CHKING HI OR LO BYTE?
1906 004664 001011 BNE 35 ;BRANCH IF HI BYTE
1907 004666 042703 177600 BIC #177600,R3 ;MASK HI BYTE
1908 004672 001417 BEQ 45 ;OK IF LO BYTE WAS CLEARED
1909 004674 005037 001162 CLR $REG0 ;GET EXPCTD RKCS.
1910 004700 010337 001164 MOV R3,$REG1 ;GET RKCS RECVD, LO BYTE
1911 004704 104022 ERROR 22 ;ALL RK11 REGISTER'S WERE LOADED WITH
1912 ;1'S THEN TRIED TO ADRES & CLR RKCS
1913 ;LO BYTE, IT COULD NOT BE CLEARED.
1914 004706 000411 BR 45
1915 004710 042703 000377 35: BIC #377,R3 ;MASK LO BYTE
1916 004714 001406 BEQ 45 ;OK IF HI BYTE WAS CLEARED
1917 004716 005037 001162 CLR $REG0 ;GET EXPCTD RKCS, HI BYTE
1918 ;GET WHAT WAS ACTUALLY RECVD
1919 004722 000303 SWAB R3
1920 004724 010337 001164 MOV R3,$REG1
1921 004730 104023 ERROR 23 ;ALL RK11 REGISTER'S WERE LOADED WITH
1922 ;1'S THEN TRIED TO ADRES & CLR RKCS
1923 ;HI BYTE IT COULD NOT BE CLEARED.
1924 004732 012700 177774 45: MOV #-4,R0 ;INITIALIZE COUNT FOR REST OF REGISTERS
1925 004736 013705 001250 MOV RKCS,R5 ;INITIALIZE POINTER TO RKCS
1926 004742 005725 55: TST (R5)+ ;INCREMENT PTR TO NXT REGIS
1927 004744 005200 INC R0 ;ALL REGS DONE?
1928 004746 001417 BEQ 65 ;IF YES, GO & ADRES TO CLEAR RKCS HI BYTE
1929 004750 011503 MOV #R5,R3 ;IF NOT, GET CONTENTS OF THE REGIS
1930 ;BEING CHKD
1931 004752 005103 COM R3 ;COMPLEMENT THE CONTENTS. SHOULD BE
1932 ;0 FOR IT WAS
1933 004754 001772 BEQ 55 ;PREVIOUSLY SET TO ALL 1'S IF IT'S
1934 ;0 BRANCH, IF NOT REPORT ERROR
1935 004756 010137 001162 MOV R1,$REG0 ;GET RKCS-BYTE-ADRES WHICH WAS *PIED
1936 ;TO ADRES
1937 004762 010537 001164 MOV R5,$REG1 ;GET ADRES OF REGIS WHICH GOT ADRESED
1938 ;INSTEAD
1939 004766 012737 177777 001166 MOV #177777,$REG2 ;GET EXPCTD CONTENTS OF REGISTER
1940 ;THAT GO ADRESED
1941 004774 005103 COM R3 ;GET CONTENTS RECVD FROM THAT REGIS

```



```

1942 004776 010337 001170      MOV    R3,$REG3
1943 005002 104024      ERROR    24      ;ALL RK11 REGISTERS WERE LOADED
1944                                     ;WITH 1'S. RKCS
1945                                     ;BYTE (ADRES UNDER 'BYTE' IN ER
1946                                     ;MSG) WAS ADRESED
1947                                     ;USING 'CLRB', BUT REGISTER (ADRES
1948                                     ;UNDER 'REGIS' IN
1949                                     ;ER MSG) GOT CHANGED AS A RESULT.
1950 005004 000756      BR        5$
1951 005006 005201      6$: INC     R1
1952 005010 005202      INC     R2      ;POSITION PTR TO RKCS HI BYTE
1953                                     ;CHK IF BOTH HI & LO BYTES (RKCS)
1954 005012 001316      BNE     2$      ;WERE CLEARED
1955                                     ;IF NOT BRANCH BACK
1956                                     ;*****
1957 :*TEST 24 CHECK THAT HI & LO BYTES OF RKWC,BA,DA CAN BE ADDRESSED
1958 :*THIS TEST CHECKS THAT BYTE OPERATIONS ON RKWC, RKBA & RKDA CAN BE DONE
1959 :*CORRECTLY. FIRST RKWC, RKCS, RKBA, RKDA ARE SET TO 177777.
1960 :*1) REGISTER BYTE POINTED TO BY R2 IS CLEARED USING 'CLRB'
1961 :*2) IT IS CHECKED THAT ONLY THAT BYTE AND NO OTHER BYTES GET CLEARED
1962 :*3) POINTER R2 IS INCREMENTED TO THE NEXT REGISTER-BYTE & THE PROCESS
1963 :*IS REPEATED. LO BYTE IS DONE FIRST, THEN HI-BYTE IS DONE.
1964 :*****
1965 005014 000004      ST24: SCOPE
1966 005016 012746 000340      MOV     #340,-(SP)
1967 005022 012746 005030      MOV     #64$,-(SP)
1968 005026 000002      RTI
1969 005030      64$:
1970 005030 012704 177772      MOV     #-6,R4      ;SET UP COUNT FOR 6 REG-BYTES TO BE ADRESED
1971 005034 013702 001252      MOV     RKWC,R2      ;INITIALIZE POINTER TO RKWC
1972 005040 012700 177775      1$: MOV     #-3,R0
1973 005044 013701 001250      MOV     RKCS,R1
1974 005050 012721 007576      MOV     #7576,(R1)+      ;SET RKCS BITS
1975 005054 012721 177777      2$: MOV     #177777,(R1)+      ;RKWC
1976 005060 005200      INC     R0      ;RKBA
1977 005062 001374      BNE     2$      ;RKDA
1978
1979 005064 105012      CLRB    R2      ;ADDRESS & CLEAR REGIS BYTE UNDER 'EST'
1980
1981 005066 010200      MOV     R2,R0
1982 005070 042700 000001      BIC     #1,R0      ;CONVERT THE BYTE ADRES INTO WORD
1983                                     ;ADRES THAT IT BELONGS TO
1984 005074 011001      MOV     R0,R1      ;GET THE ENTIRE REGIS WRD
1985 005076 032702 000001      BIT     #1,R2      ;WAS THE CLRD BYTE HI OR LO?
1986 005102 001003      BNE     3$      ;WAS HI, BRANCH
1987 005104 042701 177400      BIC     #177400,R1      ;WAS LO-MASK HI BYTE
1988 005110 000402      BR      4$
1989 005112 042701 000377      3$: BIC     #377,R1      ;MASK LO BYTE
1990 005116 001411      4$: BEQ     5$      ;WAS THE ADDRESSED BYTE CLEARED? BR IF YES
1991 005120 010237 001162      MOV     R2,$REG0      ;GET ADRES OF REG-BYTE THAT WAS
1992                                     ;TRIED TO B ADRESED & CLEARED
1993 005124 032702 000001      BIT     #1,R2
1994 005130 001401      BEQ     11$
1995 005132 000301      SWAB    R1
1996 005134 010137 001164      11$: MOV     R1,$REG1      ;GET CONTENTS OF REG-BYTE
1997 005140 104025      ERROR    2$      ;TRIED TO ADRES & CLR A REGISTER BYTE

```

```

1998
1999
2000 005142 017701 174102      5$:  MOV  2RKCS,R1      ;(ADRES UNDER 'REG-BYTE' IN ER MSGE), COULD
2001 005146 022701 007776      CMP  #7776,R1      ;NOT CLEAR IT.
2002 005152 001410              BEQ  6$,          ;WAS RKCS ERRONEOUSLY CLRD?
2003 005154 010237 001162      MOV  R2,$REG0      ;NO BRANCH
2004                      ;GET ADRES OF REG-BYTE THAT WAS
2005 005160 012737 007776 001164  MOV  #7776,$REG1  ;TRIED TO B ADRESED & CLEARED.
2006 005166 010137 001166      MOV  R1,$REG2      ;GET EXPCTD RKCS
2007 005172 104016              ERROR 16          ;GET RKCS RECVD
2008                      ;ALL RK11 REGISTRARS WERE LOADED WITH 1'S
2009                      ;TRIED TO ADRES & CLR 'REGIS-BYTE' (IN ER MSGE)
2010                      ;RKCS GOT CHANGED AS A RESULT. '(RKCS)EXP'
2011                      ;CONTAINS EXPCTD RKCS. 'RKCS (RECVD)' CONTAINS
2012 005174 012700 177772      6$:  MOV  #-6,R0      ;RKCS RECVD.
2013 005200 013701 001252      MOV  RKWC,R1      ;SET COUNT FOR BYTES
2014 005204 020102      7$:  CMP  R1,R2      ;INITIALIZE PTR TO RKWC
2015                      ;WAS THE CLEARED BYTE (PTD TO BY R2)
2016 005206 001433              BEQ  10$,          ;SAME AS BYTE TO BE CHKD (PTD TO BY R1)?
2017 005210 010105              MOV  R1,R5      ;IF YES, DO NOT CHK THIS BYTE
2018 005212 042705 000001      BIC  #1,R5      ;STRIP WORD ADDRESS FROM BYTE ADDR
2019 005216 011503              MOV  2R5,R3      ;IS THE BYTE TO B CHKD LO OR HI BYTE?
2020 005220 032701 000001      BIT  #1,R1      ;HI BYTE-BRANCH
2021 005224 001003              BNE  8$,          ;MASK HI BYTE
2022 005226 042703 177400      BIC  #177400,R3
2023 005232 000402              BR  9$,          ;MASK LO BYTE
2024 005234 042703 000377      8$:  BIC  #377,R3
2025 005240 001016      9$:  BNE  10$,
2026 005242 010237 001162      MOV  R2,$REG0      ;GET ADRES OF REG-BYTE THAT WAS
2027                      ;TRIED TO B ADRESED & CLEARED
2028 005246 010137 001164      MOV  R1,$REG1      ;GET ADRES OF REG-BYTE THAT GOT
2029                      ;ADRESED INSTEAD
2030 005252 012737 000377 001166  MOV  #377,$REG2  ;GET EXPCTD CONTENTS OF REG-BYTE
2031                      ;THAT GOT ADRESED
2032 005260 032701 000001      BIT  #1,R1
2033 005264 001401              BEQ  12$,
2034 005266 000303              SWAB R3
2035 005270 010337 001170      12$: MOV  R3,$REG3      ;GET CONTENTS RECVD FOR THAT REG-BYTE
2036 005274 104021              ERROR 21          ;ALL RK11 REGISTRARS WERE LOADED WITH 1'S.
2037                      ;TRIED TO ADRES & CLR 'REG-BYT1' (IN ER MSGE)
2038                      ;'REG-BYT2' GOT CHANGED AS A RESULT.
2039                      ;'BYT2-EXPCT' (IN ER MSGE) IS THE EXPCTD CONTENTS
2040                      ;OF REG-BYT2. 'BYT2-RECVD' IS THE CONTENTS RECVD.
2041 005276 005201      10$:  INC  R1      ;INCREMENT PTR TO NXT REG BYTE
2042 005300 005200              INC  R0      ;ALL BYTES CHKD?
2043 005302 001340              BNE  7$,          ;NO, LOOP BACK
2044 005304 005202              INC  R2      ;INCREMENT PTR TO NXT REG BYTE TO B CLRD
2045 005306 005204              INC  R4      ;ALL BYTES CLRD?
2046 005310 001253              BNE  1$,          ;LOOP BACK
2047
2048
2049
2050
2051      .SBTTL  END OF PASS ROUTINE
2052
2053      ;:*****

```

TEST 1 MACY11 30.1046, 06-JUN-77 14:40 PAGE 40
END OF PASS ROUTINE

```

2054      ;*INCREMENT THE PASS NUMBER ($PASS)
2055      ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM
2056      ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
2057      ;*IF THERES A MONITOR GO TO IT
2058      ;*IF THERE ISN'T JUMP TO START1
2059
2060      $EOP:
2061      SCOPE
2062      CLR $STNM      ;;ZERO THE TEST NUMBER
2063      CLR $TIMES     ;;ZERO THE NUMBER OF ITERATIONS
2064      INC $PASS      ;;INCREMENT THE PASS NUMBER
2065      BIC #100000,$PASS ;;DON'T ALLOW A NEG. NUMBER
2066      DEC (PC)+      ;;LOOP?
2067      $EOPCT: .WORD 1
2068      BGT $DOAGN      ;;YES
2069      MOV (PC)+,$(PC)+ ;;RESTORE COUNTER
2070      $ENDCT: .WORD 1
2071      $EOPCT
2072      TYPE $ENDMG      ;;TYPE "END PASS #"
2073      MOV $PASS,-(SP)   ;;SAVE $PASS FOR TYPEOUT
2074      TYPDS           ;;GO TYPE--DECIMAL ASCII WITH SIGN
2075      TYPE $ENULL      ;;TYPE A NULL CHARACTER
2076      $GET42: MOV #42,R0 ;;GET MONITOR ADDRESS
2077      BEQ $DOAGN       ;;BRANCH IF NO MONITOR
2078      RESET           ;;CLEAR THE WORLD
2079      $ENDAD: JSR PC,(R0) ;;GO TO MONITOR
2080      NOP             ;;SAVE ROOM
2081      NOP             ;;FOR
2082      NOP             ;;ACT11
2083      $DOAGN:
2084      JMP $(PC)+      ;;RETURN
2085      $RTNAD: .WORD START1
2086      $ENULL: .BYTE -1,-1,0 ;;NULL CHARACTER STRING
2087      $ENDMG: .ASCII (15)(12) "END PASS #."
2088
2089      005312 000004
2090      005312 005037 001102
2091      005314 005037 001206
2092      005320 005237 001100
2093      005324 042737 100000 001100
2094      005330 005327
2095      005336 000001
2096      005340 003022
2097      005342 012737
2098      005344 000001
2099      005350 005340
2100      005352 104401 005417
2101      005356 013746 001100
2102      005362 104405
2103      005364 104401 005414
2104      005370 013700 000042
2105      005374 001403
2106      005376 000005
2107      005400 004710
2108      005402 000240
2109      005404 000240
2110      005406 000240
2111      005410
2112      005410 000137
2113      005412 002110
2114      005414 377 377 000
2115      005417 015 042412 042116
2116      005424 050040 051501 020123
2117      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
 :GT4RG: JSR PC,GT3RG
 :MOV JAKDA,\$REG3
 :RTS PC

005460 004737 005434
 005461 017737 173566 001170
 005472 000207

.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 DELAY: MOV 2,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 IS: DEC TIMER :COUNT DOWN TO 0
 005512 001375 :BNE IS
 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

```

2166 ;CN.RDY
2167 ;THIS ROUTINE WAITS FOR THE CONTROL READY BIT TO SET AND WHEN IT
2168 ;SETS EXITS OUT. IF WITHIN A CERTAIN TIME CNTRL RDY DOES
2169 ;NOT SET AN ERROR IS REPORTED. WAITING TIME IS 883 MS FOR 1: 20
2170 ;175 MS FOR 11/45 WITH BIPOLAR MEMORY.
2171 ;CALL: CNT.RDY
2172 005516 012777 000001 173524 CN.RST: MOV #1,ARKCS ;ISSUE A CONTROL RESET
2173 005524 012737 177500 001170 MOV #300,$REG3 ;SET UP COUNT
2174 005532 000402 BR CN.RDY+4 ;SKIP OVER CN.RDY
2175 005534 005037 001170 CN.RDY: CLR $REG3
2176 005540 105777 173504 IS: TSTB ARKCS ;DID CNTRL-RDY SET?
2177 005544 100435 BMI 3$ ;YES, EXIT
2178 005546 005237 001170 INC $REG3 ;WAITED LONG?
2179 005552 001372 BNE 1$ ;IF NOT, GO BAK & WAIT
2180 005554 032777 020000 173356 2$: BIT #SW13,$SWR ;INHIBIT TYPEOUT?
2181 005562 001026 BNE 3$ ;IF YES, SKIP TYPEOUT
2182 005564 104401 TYPE MSG3
2183 005566 001216 TYPE MSG3
2184 005570 104401 005576 TYPE 65$ ;:TYPE ASCIZ STRING
2185 005574 000403 BR 64$ ;:GET OVER THE ASCIZ
2186 ;:65$: .ASCIZ '<15><12> PC='
2187 64$: MOV (SP),-(SP)
2188 005604 011646 SUB #2,(SP)
2189 005606 162716 000002 TYPCO ;GO TYPE PC IN THE MAIN PROGRAM.
2190 005612 104402 ; WHERE ERROR OCCURRED
2191 ;:TYPE ASCIZ STRING
2192 005614 104401 005622 TYPE 67$
2193 005620 000404 BR 66$ ;:GET OVER THE ASCIZ
2194 ;:67$: .ASCIZ 'ARKCS='
2195 66$: MOV ARKCS, -(SP) ;GET ARKCS
2196 005632 017746 173412 TYPCO ;GO TYPE IT
2197 005636 104402
2198
2199 005640 000002 3$: RTI ;RETURN FROM THIS
2200 ;ROUTINE TO THE MAIN
2201 ;PROGRAM
2202
2203 ;THIS PART OF THE PROGRAM CONTAINS THE COMMON ROUTINES CALLED
2204 ;FROM THE SYSMAC.SML PACKAGE
2205 ;
2206 ;
2207 ;
2208 ;
2209 ;
2210 ;
2211 ;
2212 ;
2213 ;
2214 ;
2215 ;
2216 ;
2217 ;
2218 ;
2219 ;
2220 ;
2221 ;
2222 ;
2223 ;
2224 ;
2225 ;
2226 ;
2227 ;
2228 ;
2229 ;
2230 ;
2231 ;
2232 ;
2233 ;
2234 ;
2235 ;
2236 ;
2237 ;
2238 ;
2239 ;
2240 ;
2241 ;
2242 ;
2243 ;
2244 ;
2245 ;
2246 ;
2247 ;
2248 ;
2249 ;
2250 ;
2251 ;
2252 ;
2253 ;
2254 ;
2255 ;
2256 ;
2257 ;
2258 ;
2259 ;
2260 ;
2261 ;
2262 ;
2263 ;
2264 ;
2265 ;
2266 ;
2267 ;
2268 ;
2269 ;
2270 ;
2271 ;
2272 ;
2273 ;
2274 ;
2275 ;
2276 ;
2277 ;
2278 ;
2279 ;
2280 ;
2281 ;
2282 ;
2283 ;
2284 ;
2285 ;
2286 ;
2287 ;
2288 ;
2289 ;
2290 ;
2291 ;
2292 ;
2293 ;
2294 ;
2295 ;
2296 ;
2297 ;
2298 ;
2299 ;
2300 ;
2301 ;
2302 ;
2303 ;
2304 ;
2305 ;
2306 ;
2307 ;
2308 ;
2309 ;
2310 ;
2311 ;
2312 ;
2313 ;
2314 ;
2315 ;
2316 ;
2317 ;
2318 ;
2319 ;
2320 ;
2321 ;
2322 ;
2323 ;
2324 ;
2325 ;
2326 ;
2327 ;
2328 ;
2329 ;
2330 ;
2331 ;
2332 ;
2333 ;
2334 ;
2335 ;
2336 ;
2337 ;
2338 ;
2339 ;
2340 ;
2341 ;
2342 ;
2343 ;
2344 ;
2345 ;
2346 ;
2347 ;
2348 ;
2349 ;
2350 ;
2351 ;
2352 ;
2353 ;
2354 ;
2355 ;
2356 ;
2357 ;
2358 ;
2359 ;
2360 ;
2361 ;
2362 ;
2363 ;
2364 ;
2365 ;
2366 ;
2367 ;
2368 ;
2369 ;
2370 ;
2371 ;
2372 ;
2373 ;
2374 ;
2375 ;
2376 ;
2377 ;
2378 ;
2379 ;
2380 ;
2381 ;
2382 ;
2383 ;
2384 ;
2385 ;
2386 ;
2387 ;
2388 ;
2389 ;
2390 ;
2391 ;
2392 ;
2393 ;
2394 ;
2395 ;
2396 ;
2397 ;
2398 ;
2399 ;
2400 ;
2401 ;
2402 ;
2403 ;
2404 ;
2405 ;
2406 ;
2407 ;
2408 ;
2409 ;
2410 ;
2411 ;
2412 ;
2413 ;
2414 ;
2415 ;
2416 ;
2417 ;
2418 ;
2419 ;
2420 ;
2421 ;
2422 ;
2423 ;
2424 ;
2425 ;
2426 ;
2427 ;
2428 ;
2429 ;
2430 ;
2431 ;
2432 ;
2433 ;
2434 ;
2435 ;
2436 ;
2437 ;
2438 ;
2439 ;
2440 ;
2441 ;
2442 ;
2443 ;
2444 ;
2445 ;
2446 ;
2447 ;
2448 ;
2449 ;
2450 ;
2451 ;
2452 ;
2453 ;
2454 ;
2455 ;
2456 ;
2457 ;
2458 ;
2459 ;
2460 ;
2461 ;
2462 ;
2463 ;
2464 ;
2465 ;
2466 ;
2467 ;
2468 ;
2469 ;
2470 ;
2471 ;
2472 ;
2473 ;
2474 ;
2475 ;
2476 ;
2477 ;
2478 ;
2479 ;
2480 ;
2481 ;
2482 ;
2483 ;
2484 ;
2485 ;
2486 ;
2487 ;
2488 ;
2489 ;
2490 ;
2491 ;
2492 ;
2493 ;
2494 ;
2495 ;
2496 ;
2497 ;
2498 ;
2499 ;
2500 ;

```

E04

```

2222 005642          $SCOPE:
2223 005642 104407          CKSWR          ;; TEST FOR CHANGE IN SOW*-SWR
2224 005644 032777 040000 173266 1$: BIT    #BIT14,2SWR          ;; LOOP ON PRESENT TEST?
2225 005652 001111          BNE    $OVER          ;; YES IF SW14=1
2226          ;*****START OF CODE FOR THE XOR TESTER*****
2227 005654 000416          $XTSTR: BR    6$          ;; IF RUNNING ON THE "XOR" TESTER CHANGE
2228          ;THIS INSTRUCTION TO A "NOP" (NOP=240)
2229 005656 013746 000004          MOV    2$ERRVEC, -(SP)          ;; SAVE THE CONTENTS OF THE ERROR VEC*CP
2230 005662 012737 005702 000004          MOV    #5$, 2$ERRVEC          ;; SET FOR TIMEOUT
2231 005670 005737 177060          TST    2$177060          ;; TIME OUT ON XOR?
2232 005674 012637 000004          MOV    (SP)+, 2$ERRVEC          ;; RESTORE THE ERROR VECTOR
2233 005700 000463          BR    $SVLAD          ;; GO TO THE NEXT TEST
2234 005702 022626          5$: CMP    (SP)+, (SP)+          ;; CLEAR THE STACK AFTER A TIME OUT
2235 005704 012637 000004          MOV    (SP)+, 2$ERRVEC          ;; RESTORE THE ERROR VECTOR
2236 005710 000423          BR    7$          ;; LOOP ON THE PRESENT TEST
2237 005712          6$: ;*****END OF CODE FOR THE XOR TESTER*****
2238 005712 032777 000400 173220          BIT    #BIT08,2SWR          ;; LOOP ON SPEC. TEST?
2239 005720 001404          BEQ    2$          ;; BR IF NO
2240 005722 127737 173212 001102          CMPB   2SWR, $STNM          ;; ON THE RIGHT TEST? SWR/7:0/
2241 005730 001462          BEQ    $OVER          ;; BR IF YES
2242 005732 105737 001103          2$: TSTB   $ERFLG          ;; HAS AN ERROR OCCURRED?
2243 005736 001421          BEQ    3$          ;; BR IF NO
2244 005740 123737 001115 001103          CMPB   $ERMAX, $ERFLG          ;; MAX. ERRORS FOR THIS TEST OCCURRED?
2245 005746 101015          BHI    3$          ;; BR IF NO
2246 005750 032777 001000 173162          BIT    #BIT09,2SWR          ;; LOOP ON ERROR?
2247 005756 001404          BEQ    4$          ;; BR IF NO
2248 005760 013737 001110 001106          7$: MOV    $LPERR, $LPADR          ;; SET LOOP ADDRESS TO LAST SCOPE
2249 005766 000443          BR    $OVER
2250 005770 105037 001103          4$: CLRB   $ERFLG          ;; ZERO THE ERROR FLAG
2251 005774 005037 001206          CLR    $TIMES          ;; CLEAR THE NUMBER OF ITERATIONS TO MAKE
2252 006000 000415          BR    1$          ;; ESCAPE TO THE NEXT TEST
2253 006002 032777 004000 173130          3$: BIT    #BIT11,2SWR          ;; INHIBIT ITERATIONS?
2254 006010 001011          BNE    1$          ;; BR IF YES
2255 006012 005737 001100          TST    $PASS          ;; IF FIRST PASS OF PROGRAM
2256 006016 001406          BEQ    1$          ;; INHIBIT ITERATIONS
2257 006020 005237 001104          INC    $ICNT          ;; INCREMENT ITERATION COUNT
2258 006024 023737 001206 001104          CMP    $TIMES, $ICNT          ;; CHECK THE NUMBER OF ITERATIONS MADE
2259 006032 002021          BGE    $OVER          ;; BR IF MORE ITERATION REQUIRED
2260 006034 012737 000001 001104          1$: MOV    #1, $ICNT          ;; REINITIALIZE THE ITERATION COUNTER
2261 006042 013737 006112 001206          MOV    $MXCNT, $TIMES          ;; SET NUMBER OF ITERATIONS TO DO
2262 006050 105237 001102          $SVLAD: INCB   $STNM          ;; COUNT TEST NUMBERS
2263 006054 011637 001106          MOV    (SP), $LPADR          ;; SAVE SCOPE LOOP ADDRESS
2264 006060 011637 001110          MOV    (SP), $LPERR          ;; SAVE ERROR LOOP ADDRESS
2265 006064 005037 001210          CLR    $ESCAPE          ;; CLEAR THE ESCAPE FROM ERROR ADDRESS
2266 006070 112737 000001 001115          MOVB   #1, $ERMAX          ;; ONLY ALLOW ONE(1) ERROR ON NEXT TEST
2267 006076 013777 001102 173036          $OVER: MOV    $STNM, 2$DISPLAY          ;; DISPLAY TEST NUMBER
2268 006104 013716 001106          MOV    $LPADR, (SP)          ;; FUDGE RETURN ADDRESS
2269 006110 000002          RTI          ;; FIXES PS
2270 006112 000050          $MXCNT: SC          ;; MAX. NUMBER OF ITERATIONS

```

::*****

.SBTTL ERROR HANDLER ROUTINE

:*SW15=1 HALT ON ERROR

F04

```

2279      *SW13=1      INHIBIT ERROR TYPEOUTS
2280      *SW10=1      BELL ON ERROR
2281      *SW09=1      LOOP ON ERROR
2282      *SW12=1      CYCLE ON ERROR TO PREVIOUS 'SCOPE'
2283      *GO TO SERRTYP ON ERROR

2284 006114 104407      SERROR: CKSWR      :CHECK FOR SOFTWARE SWITCH ENTRY REQUEST
2285 006116 105237 001103 5: INCB SERFLG      :SET THE ERROR FLAG
2286 006122 001775      BEQ 7$      :DON'T LET THE FLAG GO TO ZERO
2287 006124 013777 001102 173010 MOV $TSTNM, @DISPLAY :DISPLAY TEST NUMBER AND ERROR FLAG
2288 006132 005237 001112 1$: INC SERRTL      :COUNT THE NUMBER OF ERRORS
2289 006136 011637 001116      MOV (SP), SERRPC :GET ADDRESS OF ERROR INSTRUCTION
2290 006142 162737 000002 001116 SUB #2, SERRPC
2291 006150 117737 172742 001114 MOVB @SERRPC, $ITEMB :STRIP AND SAVE THE ERROR ITEM CODE
2292 006156 032777 020000 172754 BIT #SW13, @SWR :SKIP TYPEOUT IF SET
2293 006164 001004      BNE 2$      :SKIP TYPEOUTS
2294 006166 004737 006264      JSR PC, @SERRTYP :GO TO USER ERROR ROUTINE
2295 006172 104401 001213      TYPE $CRLF
2296 006176 023737 000042 000046 2$: CMP @#42, @#46 :ARE WE IN ACTION AUTOMATIC MODE?
2297 006204 001403      BEQ .+10 :IF YES, HALT ON ERROR
2298 006206 005777 172726      TST @SWR :HALT ON ERROR
2299 006212 100002      BPL 3$      :SKIP IF CONTINUE
2300 006214 000007      HALT :HALT ON ERROR!
2301 006216 104407      CKSWR :CHECK FOR SOFTWARE SWITCH ENTRY REQUEST
2302 006220 032777 010000 172712 3$: BIT #SW12, @SWR :SW 12 SET?
2303 006226 001402      BEQ .+6 :NO, BRANCH
2304 006230 013716 001106      MOV $LPADR, SP :ADJUST RETURN ADRES FOR SW12
2305 006234 032777 001000 172676 BIT #SW09, @SWR :LOOP ON ERROR SWITCH SET?
2306 006242 001402      BEQ 4$      :BR IF NO
2307 006244 013716 001110      MOV $LPERR, SP :FUDGE RETURN FOR LOOPING
2308 006250 005737 001210 4$: TST $ESCAPE :CHECK FOR AN ESCAPE ADDRESS
2309 006254 001402      BEQ 5$      :BR IF NONE
2310 006256 013716 001210      MOV $ESCAPE, SP :FUDGE RETURN ADDRESS FOR ESCAPE
2311 006262 000002      5$: 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" SERRTE,
 :AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.

```

2321 006264 104401 001213 SERRTYP: TYPE $CRLF      : "CARRIAGE RETURN" & "LINE FEED"
2322 006270 010046      MOV RD, -(SP) :SAVE RD
2323 006272 005000      CLR RC :PICKUP THE ITEM INDEX
2324 006274 153700 001114 BISB @#$ITEMB, RC
2325 006300 001004      BNE 1$      : IF ITEM NUMBER IS ZERO, JUST
2326      : TYPE THE PC OF THE ERROR
2327 006302 013746 001116      MOV SERRPC, - SP :SAVE SERRPC FOR TYPEOUT
2328      : ERROR ADDRESS
2329 006306 104402      TYPE :GO TYPE--OCTAL ASCII(ALL DIGITS)
2330 006310 000426      BR 6$      :GET OUT
2331 006312 005300 1$: DEC RC :ADJUST THE INDEX SO THAT IT WILL
2332 006314 006300      ASL RC :WORK FOR THE ERROR TABLE
2333 006316 006300      ASL RC
  
```

604

```

00334 006320 006300      ASL      RO
00335 006322 062700 001272  RDC      #ERRTB,RO      :: FORM TABLE POINTER
00336 006326 012037 006335  MOV      (RO)+,2$      :: PICKUP "ERROR MESSAGE" POINTER
00337 006332 001404      BEQ      3$      :: SKIP TIMEOUT IF NO POINTER
00338 006334 104401      TYPE      :: TYPE THE "ERROR MESSAGE"
00339 006336 000000 2$:  .WORD      0      :: "ERROR MESSAGE" POINTER GOES HERE
00340 006340 104401 001213  .E      $CRLF      :: "CARRIAGE RETURN" & "LINE FEED"
00341 006344 012037 006354 3$:  MOV      (RO)+,4$      :: PICKUP "DATA HEADER" POINTER
00342 006350 001404      BEQ      5$      :: SKIP TIMEOUT IF 0
00343 006352 104401      TYPE      :: TYPE THE "DATA HEADER"
00344 006354 000000 4$:  .WORD      0      :: "DATA HEADER" POINTER GOES HERE
00345 006356 104401 001213  .E      $CRLF      :: "CARRIAGE RETURN" & "LINE FEED"
00346 006362 011000 5$:  MOV      (RO),RO      :: PICKUP "DATA TABLE" POINTER
00347 006364 001004      BNE      7$      :: GO TYPE THE DATA
00348 006366 012600 6$:  MOV      (SP)+,RO      :: RESTORE RO
00349 006370 104401 001213  .E      $CRLF      :: "CARRIAGE RETURN" & "LINE FEED"
00350 006374 000207      RTS      PC      :: RETURN
00351 006376      7$:
00352 006376 013046      MOV      2(RO)+,- SP      :: SAVE 2(RO)+ FOR TYPEOUT
00353 006400 104402      TYPCC      :: GO TYPE--OCTAL ASCII/ALL DIGITS
00354 006402 005710      TST      (RO)      :: IS THERE ANOTHER NUMBER?
00355 006404 001770      BEQ      6$      :: BR IF NO
00356 006406 104401 006414  .E      $      :: TYPE TWO(2) SPACES
00357 006412 000771      BR      7$      :: LOOP
00358 006414 020040 000 9$:  .ASCIIZ      :: TWO(2) SPACES
00359 006420 006420
00360 006420
00361 006420
00362 006420
00363 006420
00364 006420
00365 006420
00366 006420
00367 006420
00368 006420
00369 006420
00370 006420
00371 006420 022737 000176 001140 $CKSWR: CMP      #SWREG,SWR      :: IS THE SOFT-SWR SELECTED?
00372 006426 001074      BNE      15$      :: BRANCH IF NO
00373 006430 105777 172510      TSTB      2$TKS      :: CHAR THERE?
00374 006434 100071      BPL      15$      :: IF NO, DON'T WAIT AROUND
00375 006436 117746 172504      MOV      2$TKB,-(SP)      :: SAVE THE CHAR
00376 006442 042716 177600      BIC      #10177,(SP)      :: STRIP-OFF THE ASCII
00377 006446 022726 000007      CMP      #7,(SP)+      :: IS IT A CONTROL G?
00378 006452 001062      BNE      15$      :: NO, RETURN TO USER
00379 006454 123727 001134 000001      CMPB      $AUTOB,#1      :: ARE WE RUNNING IN AUTO-MODE?
00380 006462 001456      BEQ      15$      :: BRANCH IF YES
00381 006464 104401 007145      TYPE      .,$CNTLG      :: ECHO THE CONTROL-G LOG
00382 006470 104401 007152  $G*SWR: TYPE      .,$MSWR      :: TYPE CURRENT CONTENTS
00383 006474 013746 000176      MOV      SWREG,-(SP)      :: SAVE SWREG FOR TYPEOUT
00384 006500 104402      TYPCC      :: GO TYPE--OCTAL ASCII/ALL DIGITS
00385 006502 104401 007163      TYPE      .,$MNEW      :: PROMPT FOR NEW SWR
00386 006506 005046 19$:  CLP      -,$SP      :: CLEAR COUNTER
00387 006510 005046      CLP      -,$SP      :: THE NEW SWR
00388 006512 105777 172426 7$:  TSTB      2$TKS      :: CHAR THERE?

```

.SBTTL ITTY 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 ITTY FLAG MODE.

```

00371 006420 022737 000176 001140 $CKSWR: CMP      #SWREG,SWR      :: IS THE SOFT-SWR SELECTED?
00372 006426 001074      BNE      15$      :: BRANCH IF NO
00373 006430 105777 172510      TSTB      2$TKS      :: CHAR THERE?
00374 006434 100071      BPL      15$      :: IF NO, DON'T WAIT AROUND
00375 006436 117746 172504      MOV      2$TKB,-(SP)      :: SAVE THE CHAR
00376 006442 042716 177600      BIC      #10177,(SP)      :: STRIP-OFF THE ASCII
00377 006446 022726 000007      CMP      #7,(SP)+      :: IS IT A CONTROL G?
00378 006452 001062      BNE      15$      :: NO, RETURN TO USER
00379 006454 123727 001134 000001      CMPB      $AUTOB,#1      :: ARE WE RUNNING IN AUTO-MODE?
00380 006462 001456      BEQ      15$      :: BRANCH IF YES
00381 006464 104401 007145      TYPE      .,$CNTLG      :: ECHO THE CONTROL-G LOG
00382 006470 104401 007152  $G*SWR: TYPE      .,$MSWR      :: TYPE CURRENT CONTENTS
00383 006474 013746 000176      MOV      SWREG,-(SP)      :: SAVE SWREG FOR TYPEOUT
00384 006500 104402      TYPCC      :: GO TYPE--OCTAL ASCII/ALL DIGITS
00385 006502 104401 007163      TYPE      .,$MNEW      :: PROMPT FOR NEW SWR
00386 006506 005046 19$:  CLP      -,$SP      :: CLEAR COUNTER
00387 006510 005046      CLP      -,$SP      :: THE NEW SWR
00388 006512 105777 172426 7$:  TSTB      2$TKS      :: CHAR THERE?

```


Address	Instruction	Comment	Label	Register	Value	Comment
006516	BPL	75				:: IF NOT TRY AGAIN
006520	MOVB	057BA- SP				:: PICK UP CHAR
006524	BIC	0017A- SP				:: MAKE IT 7-BIT ASCII
006530	CMP	(SP), #25	95:			:: IS IT A CONTROL-U?
006534	BNE	105				:: BRANCH IF NOT
006536	TYPE	\$CNTL				:: YES, ECHO CONTROL-U
006542	ADD	#6, SP	205:			:: IGNORE PREVIOUS INPUT
006546	BR	195				:: LET'S TRY IT AGAIN
006550	CMP	(SP), #15	105:			:: IS IT A (CR)?
006554	BNE	165				:: BRANCH IF NO
006556	TST	4(SP)				:: YES, IS IT THE FIRST CHAR?
006562	BEG	115				:: BRANCH IF YES
006564	MOV	2(SP), 2SWP				:: SAVE NEW SWR
006572	ADD	#6, SP	115:			:: CLEAR UP STACK
006576	TYPE	\$CRLF	145:			:: ECHO (CR) AND (LF)
006602	CMPB	\$INTAG, #1				:: RE-ENABLE TTY KBD INTERRUPTS
006610	BNE	155				:: BRANCH IF NOT
006612	MOV	#100, 2STKS				:: RE-ENABLE TTY KBD INTERRUPTS
006620	RTI		155:			:: RETURN
006622	JSR	PC, \$TYPE0	165:			:: ECHO CHAR
006626	CMP	(SP), #60				:: CHAR < 0?
006632	BLT	185				:: BRANCH IF YES
006634	CMP	(SP), #67				:: CHAR > 7?
006640	BGT	185				:: BRANCH IF YES
006642	BIC	#60, (SP)+				:: STRIP-OFF ASCII
006646	TST	2(SP)				:: IS THIS THE FIRST CHAR
006652	BEG	175				:: BRANCH IF YES
006654	ASL	(SP)				:: NO, SHIFT PRESENT
006656	ASL	(SP)				:: CHAR OVER TO MAKE
006660	ASL	(SP)				:: ROOM FOR NEW ONE.
006662	INC	2(SP)	175:			:: KEEP COUNT OF CHAR
006666	BIS	-2(SP), (SP)				:: SET IN NEW CHAR
006672	BR	75				:: GET THE NEXT ONE
006674	TYPE	\$QUES	185:			:: TYPE ?(CR)(LF)
006700	BR	205				:: SIMULATE CONTROL-U
	DSABL	LSB				

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

006702	MOV	(SP), -(SP)	\$ROCHR:			:: PUSH DOWN THE PC
006704	MOV	4(SP), 2(SP)				:: SAVE THE PS
006712	TSTB	2STKS	15:			:: WAIT FOR
006716	BPL	15				:: A CHARACTER

```

2446 006720 117766 172222 000004      MOV     0$TKB,4(SP)      ;; READ THE TTY
2447 006726 042766 177600 000004      BIC     #1C(177),4(SP)  ;; GET RID OF JUNK IF ANY
2448 006734 026627 000004 000023      CMP     4(SP),#23      ;; IS IT A CONTROL-S?
2449 006742 001013          BNE     3$          ;; BRANCH IF NO
2450 006744 105777 172174      TS      0$TKS          ;; WAIT FOR A CHARACTER
2451 006750 100375          BPL     2$          ;; LOOP UNTIL ITS THERE
2452 006752 117746 172170      MOV     0$TKB, -(SP)  ;; GET CHARACTER
2453 006756 042716 177600      BIC     #1C177, (SP)  ;; MAKE IT 7-BIT ASCII
2454 006762 022627 000021      CMP     (SP)+, #21     ;; IS IT A CONTROL-Q?
2455 006766 001366          BNE     2$          ;; IF NOT DISCARD IT
2456 006770 000750          BR      1$          ;; YES, RESUME
2457 006772 026627 000004 000140 3$:    CMP     4(SP), #140    ;; IS IT UPPER CASE?
2458 007000 002107          BLT     4$          ;; BRANCH IF YES
2459 007002 026627 000004 000175      CMP     4(SP), #175    ;; IS IT A SPECIAL CHAR?
2460 007010 003003          BGT     4$          ;; BRANCH IF YES
2461 007012 042766 000040 000004      BIC     #40, 4(SP)     ;; MAKE IT UPPER CASE
2462 007020 000002          RTI          ;; GO BACK TO USER
2463                                     ;; *****
2464                                     ;; THIS ROUTINE WILL INPUT A STRING FROM THE TTY
2465                                     ;; *CALL:
2466                                     ;; *      RDLIN
2467                                     ;; *      RETURN HERE
2468                                     ;; *
2469                                     ;; *      INPUT A STRING FROM THE TTY
2470                                     ;; *      ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
2471                                     ;; *      TERMINATOR WILL BE A BYTE OF ALL C'S
2472 007022 010346          $RDLIN: MOV     R3, -(SP)      ;; SAVE R3
2473 007024 012703 007130 1$:      MOV     #0$TTYIN, R3    ;; GET ADDRESS
2474 007030 022703 007140 2$:      CMP     #0$TTYIN+8, R3    ;; BUFFER FULL?
2475 007034 101405          BLOS     4$          ;; BR IF YES
2476 007036 104410          RDCHR          ;; GO READ ONE CHARACTER FROM THE TTY
2477 007040 112613          MOV     (SP)+, (R3)          ;; GET CHARACTER
2478 007042 122713 000177 10$:     CMP     #177, (R3)      ;; IS IT A RUBOUT
2479 007046 001003          BNE     3$          ;; SKIP IF NOT
2480 007050 104401 001212 4$:      TYPE     3QUES        ;; TYPE A '?'
2481 007054 000763          BR      1$          ;; CLEAR THE BUFFER AND LOOP
2482 007056 111337 007126 3$:      MOV     (R3), 9$          ;; ECHO THE CHARACTER
2483 007062 104401 007126          TYPE     9$
2484 007066 122723 000015          CMP     #15, (R3)+      ;; CHECK FOR RETURN
2485 007072 001356          BNE     2$          ;; LOOP IF NOT RETURN
2486 007074 105063 177777          CLRB     -1(R3)        ;; CLEAR RETURN (THE 15)
2487 007100 104401 001214          TYPE     $LF          ;; TYPE A LINE FEED
2488 007104 012603          MOV     (SP)+, R3          ;; RESTORE R3
2489 007106 011646          MOV     (SP), -(SP)         ;; ADJUST THE STACK AND PUT ADDRESS OF THE
2490 007110 016666 000004 000002      MOV     4(SP), 2(SP)    ;; FIRST ASCII CHARACTER ON IT
2491 007116 012766 007130 000004      MOV     #0$TTYIN, 4(SP)
2492 007124 000002          RTI          ;; RETURN
2493 007126 000          9$:      .BYTE     0          ;; STORAGE FOR ASCII CHAR. TO TYPE
2494 007127 000          .BYTE     0          ;; TERMINATOR
2495 007130 000010          $TTYIN: .BLKB     8          ;; RESERVE 8 BYTES FOR TTY INPUT
2496 007140 052536 005015 000      $CNTLU: .ASCIZ  /1U/15/12/    ;; CONTROL "U"
2497 007145 136 006507 000012      $CNTLG: .ASCIZ  /1G/15/12/    ;; CONTROL "G"
2498 007152 005015 053523 020122      $MSWR:  .ASCIZ  <15><12>/SWR = /
2499 007160 020075 000
2500 007163 040 047040 053505      $MNEW:  .ASCIZ  / NEW = /
2501 007170 036440 000040
      .SBTTL  TYPE ROUTINE
  
```

J04

```

2502
2503
2504
2505
2506
2507
2508
2509
2510
2511
2512
2513
2514
2515
2516
2517
2518 007174 105737 001157
2519 007200 100002
2520 007202 000000
2521 007204 000407
2522 007206 010046
2523 007210 017600 000002
2524 007214 112046
2525 007216 001005
2526 007220 005726
2527 007222 012600
2528 007224 062716 000002
2529 007230 000002
2530 007232 122716 000011
2531 007236 001430
2532 007240 122716 000200
2533 007244 001006
2534 007246 005726
2535 007250 104401
2536 007252 001213
2537 007254 105037 007410
2538 007260 000755
2539 007262 004737 007344
2540 007266 123726 001156
2541 007272 001350
2542 007274 013746 001154
2543
2544 007300 105366 000001
2545 007304 002770
2546 007306 004737 007344
2547 007312 105337 007410
2548 007316 000770
2549
2550
2551
2552 007320 112716 000040
2553 007324 004737 007344
2554 007330 132737 000007 007410
2555 007336 001372
2556 007340 005726
2557 007342 000724

*****
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
*
$TYPE:  TSTB      $TPFLG      ;; IS THERE A TERMINAL?
        BPL       1$          ;; BR IF YES
        HALT      HERE       ;; HALT HERE IF NO TERMINAL
        BR        3$          ;; LEAVE
1$:      MOV      RO, -(SP)    ;; SAVE RO
        MOV      22(SP), RO   ;; GET ADDRESS OF ASCIZ STRING
2$:      MOVB     (RO)+, -(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
        BNE      4$          ;; BR IF IT ISN'T THE TERMINATOR
        TST      (SP)+        ;; IF TERMINATOR POP IT OFF THE STACK
60$:     MOV      (SP)+, RO    ;; RESTORE RO
3$:      ADD      #2, (SP)     ;; ADJUST RETURN PC
        RTI              ;; RETURN
4$:      CMPB     #HT, (SP)    ;; BRANCH IF <HT>
        BEQ      8$          ;; BRANCH IF NOT <CR>
        CMPB     #CRLF, (SP)  ;; BRANCH IF NOT <CRLF>
        BNE      5$          ;; POP <CR><LF> EQUIV
        TST      (SP)+        ;; TYPE A CR AND LF
        TYPE
        $CRLF
        CLRB      $CHARCNT    ;; CLEAR CHARACTER COUNT
        BR        2$          ;; GET NEXT CHARACTER
5$:      JSR      PC, $TYPEPC  ;; GO TYPE THIS CHARACTER
6$:      CMPB     $FILLC, (SP)+ ;; IS IT TIME FOR FILLER CHARS.?
        BNE      2$          ;; IF NO GO GET NEXT CHAR.
        MOV      $NULL, -(SP) ;; GET # OF FILLER CHARS. NEEDED
        AND      THE NULL CHAR.
7$:      DECB     1(SP)        ;; DOES A NULL NEED TO BE TYPED?
        BLT      6$          ;; BR IF NO--GO POP THE NULL OFF OF STACK
        JSR      PC, $TYPEPC  ;; GO TYPE A NULL
        DECB     $CHARCNT     ;; DO NOT COUNT AS A COUNT
        BR        7$          ;; LOOP

:HORIZONTAL TAB PROCESSOR
8$:      MOVB     #' , (SP)    ;; REPLACE TAB WITH SPACE
9$:      JSR      PC, $TYPEPC  ;; TYPE A SPACE
        BITB     #7, $CHARCNT ;; BRANCH IF NOT AT
        BNE      9$          ;; TAB STOP
        TST      (SP)+        ;; POP SPACE OFF STACK
        BR        2$          ;; GET NEXT CHARACTER

```

```

2558 007344 105777 171600 STYPEC: TSTB 25TPS ::WAIT UNTIL PRINTER IS READY
2559 007350 100375 BPL STYPEC
2560 007352 116677 000002 171572 MOVB 2(SP),25TPB ::LOAD CHAR TO BE TYPED INTO DATA REG.
2561 007360 122766 000015 000002 CMPB #CR,2(SP) ::IS CHARACTER A CARRIAGE RETURN?
2562 007366 001003 BNE 15 ::BRANCH IF NO
2563 007370 105037 007410 CLRB SCHARCNT ::YES--CLEAR CHARACTER COUNT
2564 007374 000406 BR STYPEX ::EXIT
2565 007376 122766 000012 000002 15: CMPB #LF,2(SP) ::IS CHARACTER A LINE FEED?
2566 007404 001402 BEQ STYPEX ::BRANCH IF YES
2567 007406 105227 INCB (PC)+ ::COUNT THE CHARACTER
2568 007410 000000 SCHARCNT: WORD 0 ::CHARACTER COUNT STORAGE
2569 007412 000207 STYPEX: RTS PC

2570
2571
2572 .SBTTL CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
2573
2574 ::*****
2575 ::THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 5-DIGIT
2576 ::SIGNED DECIMAL (ASCII) NUMBER AND TYPE IT. DEPENDING ON WHETHER THE
2577 ::NUMBER IS POSITIVE OR NEGATIVE A SPACE OR A MINUS SIGN WILL BE TYPED
2578 ::BEFORE THE FIRST DIGIT OF THE NUMBER. LEADING ZEROS WILL ALWAYS BE
2579 ::REPLACED WITH SPACES.
2580 ::CALL:
2581 ::* MOV NUM,-(SP) ::PLT THE BINARY NUMBER ON THE STACK
2582 ::* TYPDS ::GO TO THE ROUTINE
2583
2584 STYPDS:
2585 007414 010046 MOV R0,-(SP) ::PUSH R0 ON STACK
2586 007416 010146 MOV R1,-(SP) ::PUSH R1 ON STACK
2587 007420 010246 MOV R2,-(SP) ::PUSH R2 ON STACK
2588 007422 010346 MOV R3,-(SP) ::PUSH R3 ON STACK
2589 007424 010546 MOV R5,-(SP) ::PUSH R5 ON STACK
2590 007426 012746 020200 MOV #20200,-(SP) ::SET BLANK SWITCH AND SIGN
2591 007432 016605 000020 MOV 20(SP),R5 ::GET THE INPUT NUMBER
2592 007436 100004 BPL 15 ::BR IF INPUT IS POS.
2593 007440 005405 NEG R5 ::MAKE THE BINARY NUMBER POS.
2594 007442 112766 000055 000001 15: MOVB #-1,(SP) ::MAKE THE ASCII NUMBER NEG.
2595 007450 005000 CLR R0 ::ZERO THE CONSTANTS INDEX
2596 007452 012703 007630 MOV #5DBLK,R3 ::SETUP THE OUTPUT POINTER
2597 007456 112723 000040 MOVB #1,(R3)+ ::SET THE FIRST CHARACTER TO A BLANK
2598 007462 005002 25: CLR R2 ::CLEAR THE BCD NUMBER
2599 007464 016001 007620 MOV SCTL(R0),R4 ::GET THE CONSTANT
2600 007470 160105 35: SUB R1,R5 ::FORM THIS BCD DIGIT
2601 007472 002402 BLT 45 ::BR IF DONE
2602 007474 005202 INC R2 ::INCREASE THE BCD DIGIT BY 1
2603 007476 000774 BF 35
2604 007500 060105 45: ADD R1,R5 ::ADD BACK THE CONSTANT
2605 007502 005702 TST R2 ::CHECK IF BCD DIGIT=0
2606 007504 001002 BNE 55 ::FALL THROUGH IF 0
2607 007506 105716 TSTB (SP) ::STILL DOING LEADING 0'S?
2608 007510 100407 BMI 75 ::BR IF YES
2609 007512 106316 55: ASLB (SP) ::MSD?
2610 007514 103003 BCC 65 ::BR IF NO
2611 007516 116663 000001 177777 MOVB 1(SP),-1(R3) ::YES--SET THE SIGN
2612 007524 052702 65: BIS #0,R2 ::MAKE THE BCD DIGIT ASCII
2613 007530 052702 75: BIS #1,R2 ::MAKE IT A SPACE IF NOT ALREADY A DIGIT

```

L04

2614	007534	1.0223		MOVB	R2 (R3 +	:: PUT THIS CHARACTER IN THE OUTPUT BUFFER
2615	007536	005720		TST	(R0)+	:: JUST INCREMENTING
2616	007540	020027	000010	CMP	R0, #10	:: CHECK THE TABLE INDEX
2617	007544	002746		BLT	25	:: GO DO THE NEXT DIGIT
2618	007546	003002		BGT	85	:: GO TO EXIT
2619	007550	010502		MOV	R5, R2	:: GET THE LSD
2620	007552	000764		BR	65	:: GO CHANGE TO ASCII
2621	007554	105726		TSTB	(SP)+	:: WAS THE LSD THE FIRST NON-ZERO?
2622	007556	100003		BPL	95	:: BR IF NO
2623	007560	116663	177777 177776	MOVB	-1(SP), -2(R3)	:: YES--SET THE SIGN FOR TYPING
2624	007566	105013		CLRB	(R3)	:: SET THE TERMINATOR
2625	007570	012605		MOV	(SP)+, R5	:: POP STACK INTO R5
2626	007572	012603		MOV	(SP)+, R3	:: POP STACK INTO R3
2627	007574	012602		MOV	(SP)+, R2	:: POP STACK INTO R2
2628	007576	012601		MOV	(SP)+, R1	:: POP STACK INTO R1
2629	007600	012600		MOV	(SP)+, R0	:: POP STACK INTO R0
2630	007602	104401	007630	TYPE	\$DBLK	:: NOW TYPE THE NUMBER
2631	007606	016666	000002 000004	MOV	2(SP), 4(SP)	:: ADJUST THE STACK
2632	007614	012616		MOV	(SP)+, (SP)	
2633	007616	000002		RTT		:: RETURN TO USER
2634	007620	023420		\$DBLK:	10000.	
2635	007622	001750			1000.	
2636	007624	000144			100.	
2637	007626	000012			10.	
2638	007630	000004		\$DBLK:	.BLKW 4	
2639						
2640				.SBTTL	BINARY TO OCTAL (ASCII) AND TYPE	
2641						
2642				::*****		
2643				::THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT		
2644				::OCTAL (ASCII) NUMBER AND TYPE IT.		
2645				::\$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE		
2646				::CALL:		
2647				::MOV	NUM, -(SP)	::NUMBER TO BE TYPED
2648				::TYPOS		::CALL FOR TYPEOUT
2649				::.BYTE	N	::N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE
2650				::.BYTE	M	::M=1 OR 0
2651				::		::1=TYPE LEADING ZEROS
2652				::		::0=SUPPRESS LEADING ZEROS
2653				::		
2654				::\$STYPON---ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST		
2655				::\$STYPOS OR \$STYPOC		
2656				::CALL:		
2657				::MOV	NUM, -(SP)	::NUMBER TO BE TYPED
2658				::TYPON		::CALL FOR TYPEOUT
2659				::		
2660				::\$STYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER		
2661				::CALL:		
2662				::MOV	NUM, -(SP)	::NUMBER TO BE TYPED
2663				::TYPOC		::CALL FOR TYPEOUT
2664				::		
2665	007640	017646	000000	\$TYPOS:	MOV	2(SP), -(SP)
2666	007644	116637	000001 013063		MOVB	1(SP), \$OFILL
2667	007652	112637	010065		MOVB	(SP)+, \$OMODE+1
2668	007656	062716	000002		ADD	#2, (SP)
2669	007662	000406			BR	\$STYPON

```

2670 007664 112737 000001 010063 $TYPOC: MOVB #1,$OFILL ;; SET THE ZERO FILL SWITCH
2671 007672 112737 000006 010065 MOVB #6,$OMODE+1 ;; SET FOR SIX(6) DIGITS
2672 007700 112737 000005 010062 $TYPON: MOVB #5,$OCNT ;; SET THE ITERATION COUNT
2673 007706 010346 MOV R3,-(SP) ;; SAVE R3
2674 007710 010446 MOV R4,-(SP) ;; SAVE R4
2675 007712 010546 MOV R5,-(SP) ;; SAVE R5
2676 007714 113704 010065 MOVB $OMODE+1,R4 ;; GET THE NUMBER OF DIGITS TO TYPE
2677 007720 005404 NEG R4
2678 007722 062704 000006 ADD #6,R4 ;; SUBTRACT IT FOR MAX. ALLOWED
2679 007726 110437 010064 MOVB R4,$OMODE ;; SAVE IT FOR USE
2680 007732 113704 010063 MOVB $OFILL,R4 ;; GET THE ZERO FILL SWITCH
2681 007736 016605 000012 MOV 12(SP),R5 ;; PICKUP THE INPUT NUMBER
2682 007742 005003 CLR R3 ;; CLEAR THE OUTPUT WORD
2683 007744 006105 1$: ROL R5 ;; ROTATE MSB INTO "C"
2684 007746 000404 BR 3$ ;; GO DO MSB
2685 007750 006105 2$: ROL R5 ;; FORM THIS DIGIT
2686 007752 006105 ROL R5
2687 007754 006105 ROL R5
2688 007756 010503 MOV R5,R3
2689 007760 006103 3$: ROL R3 ;; GET LSB OF THIS DIGIT
2690 007762 105337 010064 DECB $OMODE ;; TYPE THIS DIGIT?
2691 007766 100016 BPL 7$ ;; BR IF NO
2692 007770 042703 177770 BIC #177770,R3 ;; GET RID OF JUNK
2693 007774 001002 BNE 4$ ;; TEST FOR 0
2694 007776 005704 TST R4 ;; SUPPRESS THIS 0?
2695 010000 001403 BEQ 5$ ;; BR IF YES
2696 010002 005204 4$: INC R4 ;; DON'T SUPPRESS ANYMORE 0'S
2697 010004 052703 000060 BIS #'0,R3 ;; MAKE THIS DIGIT ASCII
2698 010010 052703 000040 5$: BIS #'',R3 ;; MAKE ASCII IF NOT ALREADY
2699 010014 110337 010060 MOVB R3,$$ ;; SAVE FOR TYPING
2700 010020 104401 010060 TYPE 8$ ;; GO TYPE THIS DIGIT
2701 010024 105337 010062 7$: DECB $OCNT ;; COUNT BY 1
2702 010030 003347 BGT 2$ ;; BR IF MORE TO DO
2703 010032 002402 BLT 6$ ;; BR IF DONE
2704 010034 005204 INC R4 ;; INSURE LAST DIGIT ISN'T A BLANK
2705 010036 000744 BR 2$ ;; GO DO THE LAST DIGIT
2706 010040 012605 6$: MOV (SP)+,R5 ;; RESTORE R5
2707 010042 012604 MOV (SP)+,R4 ;; RESTORE R4
2708 010044 012603 MOV (SP)+,R3 ;; RESTORE R3
2709 010046 016666 000002 000004 MOV 2(SP),4(SP) ;; SET THE STACK FOR RETURNING
2710 010054 012616 MOV (SP)+,(SP)
2711 010056 000002 RTI ;; RETURN
2712 010060 000 8$: .BYTE 0 ;; STORAGE FOR ASCII DIGIT
2713 010061 000 .BYTE 0 ;; TERMINATOR FOR TYPE ROUTINE
2714 010062 000 $OCNT: .BYTE 0 ;; OCTAL DIGIT COUNTER
2715 010063 000 $OFILL: .BYTE 0 ;; ZERO FILL SWITCH
2716 010064 000000 $OMODE: .WORD 0 ;; NUMBER OF DIGITS TO TYPE
2717
2718 .SBTTL TRAP DECODER
2719
2720 ;;*****
2721 ;;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
2722 ;;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
2723 ;;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
2724 ;;*GO TO THAT ROUTINE.
2725

```



```

2726 010066 010046          $TRAP: MOV    RO, -(SP)      ;; SAVE RO
2727 010070 016600 000002    MOV    2(SP), RO      ;; GET TRAP ADDRESS
2728 010074 005740          TST     -(RO)          ;; BACKUP BY 2
2729 010076 111000          MOVB   (RO), RO        ;; GET RIGHT BYTE OF TRAP
2730 010100 006300          ASL     RO              ;; POSITION FOR INDEXING
2731 010102 016000 010122    MOV    $TRPAD(RO), RO  ;; INDEX TO TABLE
2732 010106 000200          RTS     RO              ;; GO TO ROUTINE
2733
2734
2735      ;; THIS IS USE TO HANDLE THE "GETPRI" MACRO
2736
2737 010110 011646          $TRAP2: MOV    (SP), -(SP)  ;; MOVE THE PC DOWN
2738 010112 016666 000004 000002 MOV    4(SP), 2(SP)  ;; MOVE THE PSW DOWN
2739 010120 000002          RTI                    ;; RESTORE THE PSW
2740
2741      .SBTTL  TRAP TABLE
2742
2743      ; *THIS TABLE CONTAINS THE STARTING ADDRESSES OF THE ROUTINES CALLED
2744      ; *BY THE "TRAP" INSTRUCTION.
2745
2746      ;
2747      ; ROUTINE
2748      ; -----
2748 010122 010110  $TRPAD: .WORD  $TRAP2
2749 010124 007174  ;;CALL=TYPE      TRAP+1(104401)  TTY TYPEOUT ROUTINE
2750 010126 007664  ;;CALL=TYPOC    TRAP+2(104402)  TYPE OCTAL NUMBER (WITH LEADING ZEROS)
2751 010130 007640  ;;CALL=TYPOS    TRAP+3(104403)  TYPE OCTAL NUMBER (NO LEADING ZEROS)
2752 010132 007700  ;;CALL=TYPON    TRAP+4(104404)  TYPE OCTAL NUMBER (AS PER LAST CALL)
2753 010134 007414  ;;CALL=TYPDS    TRAP+5(104405)  TYPE DECIMAL NUMBER (WITH SIGN)
2754
2755 010136 006470  $GTSWR  ;;CALL=GTSWR  TRAP+6(104406)  GET SOFT-SWR SETTING
2756
2757 010140 006420  $CKSWR  ;;CALL=CKSWR  TRAP+7(104407)  TEST FOR CHANGE IN SOFT-SWR
2758 010142 006702  $RDCHR  ;;CALL=RDCHR  TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
2759 010144 007022  $RDLIN  ;;CALL=RDLIN  TRAP+11(104411) TTY TYPEIN STRING ROUTINE
2760
2761 010146 005516  CN.RST  ;;CALL=CN.RST  TRAP+12(104412) CONTROL RESET ROUTINE
2762
2763 010150 005534  CN.RDY  ;;CALL=CN.RDY  TRAP+13(104413) WAIT FOR CNTRL RDY TO SET
2764
2765 010152 005474  DELA.Y  ;;CALL=DELAY  TRAP+14(104414) TIME DELAY ROUTINE
2766
2767
2768      .SBTTL  POWER DOWN AND UP ROUTINES
2769
2770      ; *****
2771      ; POWER DOWN ROUTINE
2772 010154 012737 010320 000024 $PWDRN: MOV    #SILLUP, 2#PWAVEC  ;; SET FOR FAST UP
2773 010162 012737 000340 000026 MOV    #340, 2#PWAVEC+2  ;; PRI0:7
2774 010170 010046          MOV    RO, -(SP)        ;; PUSH RO ON STACK
2775 010172 010146          MOV    R1, -(SP)        ;; PUSH R1 ON STACK
2776 010174 010246          MOV    R2, -(SP)        ;; PUSH R2 ON STACK
2777 010176 010346          MOV    R3, -(SP)        ;; PUSH R3 ON STACK
2778 010200 010446          MOV    R4, -(SP)        ;; PUSH R4 ON STACK
2779 010202 010546          MOV    R5, -(SP)        ;; PUSH R5 ON STACK
2780 010204 017746 170730      MOV    2$SWR, -(SP)  ;; PUSH 2$SWR ON STACK
2781 010210 010637 010324      MOV    SP, $SAVR6    ;; SAVE SP

```

```

2782 010214 012737 010226 000324      MOV      $PWRUP,2$PWRVEC ;;SET UP VECTOR
2783 010222 000000      HALT
2784 010224 000776      BR      -2                ;;HANG UP
2785
2786 *****
2787 :POWER UP ROUTINE
2788 010226 012737 010320 000024 $PWRUP: MOV      $SILLUP,2$PWRVEC ;;SET FOR FAST DOWN
2789 010234 013706 010324      MOV      $SAVR6,SP      ;;GET SP
2790 010240 005037 010324      CLR      $SAVR6      ;;WAIT LOOP FOR THE TTY
2791 010244 005237 010324      IS:      INC      $SAVR6      ;;WAIT FOR THE INC
2792 010250 001375      BNE      IS      ;;OF WORD
2793 010252 012677 170662      MOV      (SP)+,2$SWR      ;;POP STACK INTO 2$SWR
2794 010256 012605      MOV      (SP)+,R5      ;;POP STACK INTO R5
2795 010260 012604      MOV      (SP)+,R4      ;;POP STACK INTO R4
2796 010262 012603      MOV      (SP)+,R3      ;;POP STACK INTO R3
2797 010264 012602      MOV      (SP)+,R2      ;;POP STACK INTO R2
2798 010266 012601      MOV      (SP)+,R1      ;;POP STACK INTO R1
2799 010270 012600      MOV      (SP)+,R0      ;;POP STACK INTO R0
2800 010272 012737 010154 000024      MOV      $PWRDN,2$PWRVEC ;;SET UP THE POWER DOWN VECTOR
2801 010300 012737 000340 000026      MOV      $340,2$PWRVEC+2 ;;PRIO:7
2802 010306 104401      TYPE      ;;REPORT THE POWER FAILURE
2803 010310 010326      $PWRMG: .WORD $POWER      ;;POWER FAIL MESSAGE POINTER
2804 010312 012716      MOV      (PC)+,(SP)      ;;RESTART AT PFSTRT
2805 010314 002316      $PWRAD: .WORD PFSTRT      ;;RESTART ADDRESS
2806 010316 000002      RTI
2807 010320 000000      $SILLUP: HALT      ;;THE POWER UP SEQUENCE WAS STARTED
2808 010322 000776      BR      -2                ;;BEFORE THE POWER DOWN WAS COMPLETE
2809 010324 000000      $SAVR6: 0      ;;PUT THE SP HERE
2810 010326 005015 047520 042527 $POWER: .ASCIZ  <15><12>"POWER"
2811 010334 000122      .EVEN
2812
2813      :ERROR MESSAGES
2814
2815      .SBTTL  ERROR MESSAGES
2816
2817
2818 010336 044524 042515 047440 EM1:  .ASCIZ  /TIME OUT ON RK11 REGISTER
2819 010344 052125 047440 020116
2820 010352 045522 030461 051040
2821 010360 043505 051511 042524
2822 010366 000122
2823
2824 010370 042522 044507 052123 EM2:  .ASCIZ  /REGISTER NOT CLEARED
2825 010376 051105 047040 052117
2826 010404 041440 042514 051101
2827 010412 042105      000
2828
2829 010415      122 041513 020123 EM3:  .ASCIZ  /PKCS ERROR
2830 010422 051105 047522 000122
2831
2832 010430 045522 051503 042440 EM4:  .ASCIZ  /PKCS ERROR-ON WRITING READ ONLY BITS
2833 010436 051122 051117 047455
2834 010444 020116 051127 052111
2835 010452 047111 020107 042522
2836 010460 042101 047440 046116
2837 010466 020131 044502 051524

```


2838	010474	000				
2839						
2840	010475	102	051525	044440	EM5:	.ASCIZ BUS INIT DID NOT CLEAR RKCS/
2841	010502	044516	020124	044504		
2842	010510	020104	047516	020124		
2843	010516	046103	040505	020122		
2844	010524	045522	051503	000		
2845						
2846	010531	103	052116	046122	EM6:	.ASCIZ /CNTRL RESET DIDN'T CLEAR PKCS, ON SETING 'GC'
2847	010536	051040	051505	052105		
2848	010544	042040	042111	023516		
2849	010552	020124	046103	040505		
2850	010560	020122	045522	051503		
2851	010566	020054	047117	051440		
2852	010574	052105	047111	020107		
2853	010602	043447	023517	000		
2854						
2855	010607	103	052116	046122	EM7:	.ASCIZ /CNTRL RDY DIDN'T SET AFTER CNTRL RESET
2856	010614	051040	054504	042040		
2857	010622	042111	023516	020124		
2858	010630	042523	020124	043101		
2859	010636	042524	020122	047103		
2860	010644	051124	020114	042522		
2861	010652	042523	000124			
2862						
2863	010656	042522	044507	052123	EM10:	.ASCIZ /REGISTER NOT CLEARED/
2864	010664	051105	047040	052117		
2865	010672	041440	042514	051101		
2866	010700	042105	000			
2867						
2868	010703	122	052513	020103	EM11:	.ASCIZ /RKWC ERROR/
2869	010710	051105	047522	000122		
2870						
2871						
2872	010716	045522	040502	042440	EM13:	.ASCIZ /RKBA ERROR/
2873	010724	051122	051117	000		
2874						
2875	010731	103	052116	046122	EM14:	.ASCIZ /CNTRL RESET DIDN'T CLEAR REGISTER
2876	010736	051040	051505	052105		
2877	010744	042040	042111	023516		
2878	010752	020124	046103	040505		
2879	010760	020122	042522	044507		
2880	010766	052123	051105	000		
2881						
2882	010773	122	042113	020101	EM15:	.ASCIZ /RKDA ERROR/
2883	011000	051105	047522	000122		
2884						
2885	011006	052502	020123	047111	EM17:	.ASCIZ /BUS INIT DIDN'T CLR REGISTR/
2886	011014	052111	042040	042111		
2887	011022	023516	020124	046103		
2888	011030	020122	042522	044507		
2889	011036	052123	000122			
2890						
2891	011042	042101	051104	051505	EM20:	.ASCIZ /ADDRESSING ERROR-TRIED TO ADDRESS REG1, GOT REG2
2892	011050	044523	043516	042440		
2893	011056	051122	051117	052055		

2894	011064	044522	042105	052040
2895	011072	020117	042101	051104
2896	011100	051505	020123	042522
2897	011106	030507	020054	047507
2898	011114	020124	042522	031107
2899	011122	000		
2900				
2901	011123	104	042111	023516
2902	011130	020124	046103	040505
2903	011136	020122	045522	051503
2904	011144	046040	053517	041040
2905	011152	052131	000105	
2906				
2907	011156	044504	047104	052047
2908	011164	041440	042514	051101
2909	011172	051040	041513	020123
2910	011200	044510	044107	041040
2911	011206	052131	000105	
2912				
2913	011212	051124	042511	020104
2914	011220	047524	041440	042514
2915	011226	051101	051040	041513
2916	011234	020123	041047	052131
2917	011242	023505	020054	044103
2918	011250	047101	042507	020104
2919	011256	051047	043505	051511
2920	011264	000047		
2921				
2922	011266	040506	046111	042105
2923	011274	052040	020117	046103
2924	011302	040505	020122	051047
2925	011310	043505	041055	052131
2926	011316	023505	000	
2927				
2928	011321	122	041513	020123
2929	011326	046101	042524	042522
2930	011334	020104	047117	041440
2931	011342	042514	051101	047111
2932	011350	020107	051047	043505
2933	011356	041055	052131	023505
2934	011364	000		
2935				
2936	011365	124	044522	042105
2937	011372	052040	020117	046103
2938	011400	040505	020122	051047
2939	011406	043505	041055	052131
2940	011414	023461	020054	044103
2941	011422	047101	042507	020104
2942	011430	051047	043505	041055
2943	011436	052131	023462	000
2944				
2945	011443	125	042516	050130
2946	011450	041505	042524	020104
2947	011456	045522	030461	044440
2948	011464	052116	051105	052522
2949	011472	052120	000	

EM22: .ASCIZ DIDN'T CLEAR RKCS LOW BYTE

EM23: .ASCIZ DIDN'T CLEAR RKCS HIGH BYTE

EM24: .ASCIZ /TRIED TO CLEAR RKCS 'BYTE'. CHANGED 'REGIS'

EM25: .ASCIZ FAILED TO CLEAR 'REG-BYTE'

EM26: .ASCIZ RKCS ALTERED ON CLEARING 'REG-BYTE'

EM27: .ASCIZ /TRIED TO CLEAR 'REG-BYT1'. CHANGED 'REG-BYT2'

EM43: .ASCIZ UNEXPECTED RK11 INTERRUPT

2950										
2951		011476				.EVEN				
2952										
2953						.SBTTL	ERROR DATA POINTERS			
2954										
2955	011476	001116	001162	000000	DT1:	.WORD	SERRPC,\$REG0,0			
2956										
2957	011504	001116	001162	001164	DT2:	.WORD	SERRPC,\$REG0,\$REG1,0			
2958	011512	000000								
2959										
2960	011514	001116	001162	001164	DT20:	.WORD	SERRPC,\$REG0,\$REG1,\$REG2,\$REG3,0			
2961	011522	001166	001170	000000						
2962										
2963	011530	001116	000000		DT21:	.WORD	SERRPC,0			
2964										
2965	011534	001116	001162	001164	DT25:	.WORD	SERRPC,\$REG0,\$REG1,\$REG2,0			
2966	011542	001166	000000							
2967										
2968										
2969										
2970						.SBTTL	ERROR HEADERS			
2971										
2972	011546	020040	041520	020040	DM1:	.ASCIZ	/ PC REG-ADDR			
2973	011554	051040	043505	040455						
2974	011562	042104	000122							
2975										
2976	011566	020040	041520	020040	DM2:	.ASCIZ	PC REGADD RECVD			
2977	011574	051040	043505	042101						
2978	011602	020104	020040	051040						
2979	011610	041505	042126	000						
2980										
2981	011615	040	050040	020103	DM4:	.ASCIZ	PC EXPCY RECVD			
2982	011622	020040	042440	050130						
2983	011630	052103	020040	051040						
2984	011636	041505	042126	000						
2985										
2986	011643	040	050040	020103	DM3:	.ASCIZ	PC WROTE READ			
2987	011650	020040	053440	047522						
2988	011656	042524	020040	051040						
2989	011664	040505	000104							
2990										
2991	011670	020040	041520	020040	DM5:	.ASCIZ	PC RECVD			
2992	011676	020040	042522	053103						
2993	011704	000104								
2994										
2995	011706	020040	041520	020040	DM11:	.ASCIZ	PC WROTE READ			
2996	011714	020040	051127	052117						
2997	011722	020105	020040	042522						
2998	011730	042101	000							
2999										
3000	011733	040	050040	000103	DM21:	.ASCIZ	PC			
3001										
3002	011740	020040	041520	020040	DM20:	.ASCIZ	PC REG1 REG2 (REG1) REG2			
3003	011746	020040	042522	030507						
3004	011754	020040	020040	051040						
3005	011762	043505	020062	020040						

3006	011770	051050	043505	024461					
3007	011776	020040	051050	043505					
3008	012004	024462	000						
3009									
3010	012007	040	050040	020103	DM24:	.ASCIIZ	PC	BYTE	REGIS (REG)EXP (REG)RECVD/
3011	012014	020040	020040	054502					
3012	012022	042524	020040	051040					
3013	012030	043505	051511	024040					
3014	012036	042522	024507	054105					
3015	012044	020120	051050	043505					
3016	012052	051051	041505	042126					
3017	012060	000							
3018									
3019	012061	040	050040	020103	DM25:	.ASCIIZ	PC	REG-BYTE	RECVD/
3020	012066	020040	042522	026507					
3021	012074	054502	042524	051040					
3022	012102	041505	042126	000					
3023									
3024	012107	040	050040	020103	DM26:	.ASCIIZ	PC	REG-BYT	(CS)EXP (CS)RECVD/
3025	012114	020040	042522	026507					
3026	012122	054502	020124	024040					
3027	012130	051503	042451	050130					
3028	012136	024040	051503	051051					
3029	012144	041505	042126	000					
3030									
3031	012151	040	050040	020103	DM27:	.ASCIIZ	PC	R-BYT1 R-BYT2	2-EXPCY 2-RECVD/
3032	012156	020040	051040	041055					
3033	012164	052131	020061	051040					
3034	012172	041055	052131	020062					
3035	012200	031040	042455	050130					
3036	012206	052103	031040	051055					
3037	012214	041505	042126	000					
3038									
3039	012221	040	050040	020103	DM30:	.ASCIIZ	PC	RKCS	RKEP RKDS/
3040	012226	020040	020040	045522					
3041	012234	051503	020040	020040					
3042	012242	045522	051105	020040					
3043	012250	020040	045522	051504					
3044	012256	000							
3045									
3046									
3047									
3048									
3049									
3050									
3051									
3052									
3053									
3054									
3055									
3056									
3057									
3058									
3059									
3060									
3061									
3062									
3063									
3064									
3065									
3066									
3067									
3068									
3069									
3070									
3071									
3072									
3073									
3074									
3075									
3076									
3077									
3078									
3079									
3080									
3081									
3082									
3083									
3084									
3085									
3086									
3087									
3088									
3089									
3090									
3091									
3092									
3093									
3094									
3095									
3096									
3097									
3098									
3099									
3100									
3101									
3102									
3103									
3104									
3105									
3106									
3107									
3108									
3109									
3110									
3111									
3112									
3113									
3114									
3115									
3116									
3117									
3118									
3119									
3120									
3121									
3122									
3123									
3124									
3125									
3126									
3127									
3128									
3129									
3130									
3131									
3132									
3133									
3134									
3135									
3136									
3137									
3138									
3139									
3140									
3141									
3142									
3143									
3144									
3145									
3146									
3147									
3148									
3149									
3150									
3151									
3152									
3153									
3154									
3155									
3156									
3157									
3158									
3159									
3160									
3161									
3162									
3163									
3164									
3165									
3166									
3167									
3168									
3169									
3170									
3171									
3172									
3173									
3174									
3175									
3176									
3177									
3178									
3179									
3180									
3181									
3182									
3183									
3184									
3185									
3186									
3187									
3188									
3189									
3190									
3191									
3192									
3193									
3194									
3195									
3196									
3197									
3198									
3199									
3200									
3201									
3202									
3203									
3204									
3205									
3206									
3207									
3208									
3209									
3210									
3211									
3212									
3213									
3214									
3215									
3216									
3217									
3218									
3219									
3220									
3221									
3222									
3223									
3224									
3225									
3226									
3227									
3228									

G05

002212
 002126
 003001
 000001
 000002
 000004
 000010
 000020
 000040
 000100
 000200
 000400
 001000
 000002
 002000
 004000
 010000
 020000
 040000
 100000
 000004
 000010
 000020
 000040
 000100
 000200
 000400
 001000
 000014
 104407
 104413
 104412
 005534
 005516
 000015
 000200
 177570
 104414
 005474
 011546
 011706
 011566
 011740
 011733
 012007
 012061
 012107
 012151
 011643
 012221
 011615
 011670
 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
 PRO = 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
 RKCS 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
 TIMEOUT 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
 TYP0C = 104402
 TYP0N = 104404
 TYP0S = 104403
 T1 002412
 SAUTOB 001134
 SBDADR 001122
 SBDADR 001126
 SCHARC 007410
 SCKSWR 006420
 SCMTAG 001100
 SCM1 = 000012
 SCM2 = 000024
 SCM3 = 000012
 SCNTLG 007145
 SCNTLU 007140
 SCRLF 001213
 SDBLK 007630
 SDOAGN 005410
 SDBL 007620
 SENDAO 005400
 SENDCT 005346
 SENDMG 005417
 SNULL 005414
 SEOP 005312
 SEOPCT 005340
 SERFLG 001103
 SERMAX 001115
 SERROR 006114
 SERRPC 001116
 SERRTB 001272
 SERRTY 006264
 SERTTL 001112

\$ESCAP 001210
 \$FILLC 001166
 \$FILLS 001155
 \$GCADR 001120
 \$GCADR 001124
 \$GET42 005370
 \$GTSWR 006470
 \$HC = 000000
 \$ICNT 001104
 \$ILLUP 010320
 \$INTAG 001135
 \$ITEMB 001114
 \$LF 001214
 \$LPADR 001106
 \$LPERF 001110
 \$MNEW 007163
 \$MSWR 007152
 \$MXCNT 006112
 \$NULL 001154
 \$NWTST= 000001
 \$OCNT 010062
 \$OMODE 010064
 \$OVER 006076
 \$PASS 001100
 \$POWER 010326
 \$PWRAC 010314
 \$PWRDN 010154
 \$PWRMG 010310
 \$PWRUP 010226
 \$QUES 001212
 \$RDCHR 006702
 \$RDLIN 007022
 \$RDSZ = 000010
 \$REGAO 001160
 \$REGO 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
 \$SWRMA= 000000

\$TIMES	001206	\$TPFLG	001157	\$TRPAD	010122	\$TYPEC	007344	\$XTSTR	005654
\$TKB	001146	\$TPS	001150	\$TSTNM	001102	\$TYPEX	007412	\$SGE74	000000
\$TKS	001144	\$TRAP	010066	\$TTYIN	007130	\$TYPOC	007664	\$CFILL	010000
\$TN	= 000025	\$RAP2	010110	\$TYPDS	007414	\$TYPON	007700		= 010000
\$TPB	001152	\$TRP	= 000015	\$TYPE	007174	\$TYPDS	007640		

. ABS. 012257 000

ERRORS DETECTED: 0

DSKZ:DZKJE/SOL=DSKZ:SYSMAC.SML.DSKM:DZKJE.F11
 RUN-TIME: 11 14 .3 SECONDS
 RUN-TIME RATIO: 430/27=15.8
 CORE USED: 32K (63 PAGES)

EOF:DZKJESE2 00010000 770712 PDP10 411