

# RK11

BASIC LOGIC TEST NO. 1  
MD-11-DZRKJ-D

EP-DZRKJ-D-DL-A  
COPYRIGHT © 1976  
FICHE 1 OF 1

DEC 1976  
**digital**  
MADE IN USA

The microfiche card displays a grid of 20 columns and 20 rows of frames. Each frame contains technical information, likely related to the BASIC LOGIC TEST NO. 1. The content is too small to be legible, but the layout suggests a structured presentation of data or diagrams.

11

IDENTIFICATION

SEG 0001

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

## QUICK LOOK-UP OPERATING INSTRUCTIONS

FOR A QUICK REFERENCE, LOOK UP THE FOLLOWING SECTIONS:

- 1.0 ABSTRACT
- 2.0 REQUIREMENTS
- 4.1 LOADING AND OPERATOR ACTION
- 5.0 SWITCH OPTIONS

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

## TABLE OF CONTENTS

1.0	ABSTRACT
2.0	REQUIREMENTS
2.1	EQUIPMENT
2.2	PRELIMINARY PROGRAMS
2.3	EXECUTION TIME
3.0	STARTING ADDRESS
4.0	PROGRAM CONTROL MODES & OPERATOR ACTION
4.1	PAPER TAPE
4.2	RKDP DUMP MODE
4.3	RKDP CHAIN MODE
4.4	ACT11
5.0	SWITCH OPTIONS
6.0	SCOPE LOOPS
7.0	PROGRAM STRUCTURE
8.0	ERROR REPORTING
9.0	ERROR INTERPRETATION
10.0	HANDLERS AND COMMON ROUTINES
10.1	TRAP HANDLER
10.2	SCOPE HANDLER
10.3	ERROR HANDLER
10.4	CONTROL RESET ROUTINE
10.5	CONTROL READY ROUTINE
10.6	TIME DELAY ROUTINE
10.7	OTHER ROUTINES
	TTY HANDLER (I/O), ERROR TYPEOUT ROUTINE
	POWER DOWN/POWER UP ROUTINE
11.0	UNEXPECTED TIMEOUTS & RK11 INTERRUPTS
12.0	QUICK 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. BK 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 &amp; OPERATOR ACTION

PAPER TAPE LOADING  
RKDP DUMP MODE  
RKDP CHAIN MODE  
ACT11

## 4.1 PAPER TAPE LOADING

- 4.1.1 LOAD PROGRAM INTO MEMORY USING STANDARD PROCEDURE FOR .ABS TAPES.
- 4.1.2 PUT THE DRIVES ON 'WRT PROT' AND 'LOAD' AS A PRECAUTION AGAINST MALFUNCTIONING.
- 4.1.3 LOAD ADDRESS 200

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

PRESS START.

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

RK11 LOGIC TEST I  
MAINDEC-11-DZRKJ-D

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

END PASS # X

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

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

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

4.2 RKDP DUMP MODE

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

4.2.2 START AS NORMALLY USING SA 200

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

4.3 RKDP CHAIN MODE

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

4.4 ACT11 MODE

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

5.0 SWITCH OPTIONS

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

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

'SWR = NNNNNN NEW ='

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

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

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

#### 5.1 SW<15>

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

#### 5.2 SW<14>

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

#### 5.3 SW <13>

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

#### 5.4 SW <12>

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

## 5.5 SW &lt;11&gt;

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 &lt;10&gt;

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

## 5.7 SW &lt;09&gt;

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 &lt;08&gt;

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 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

```



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

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  
1S..K..ERROR 2-->1S..K..ERROR 2-->1S...

#### 7.0 PROGRAM DESCRIPTION

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

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

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

#### 8.0 ERROR REPORTING

THE ERROR TABLE STARTING AT SERRTB 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. D - 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.

SEQ 0010

## 10.2 SCOPE HANDLER

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

## 10.3 ERROR HANDLER

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

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

## 10.4 CONTROL RESET ROUTINE

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

## 10.5 CONTROL READY ROUTINE

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

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

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

#### 10.6 TIME DELAY ROUTINE

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

#### 10.7 OTHER ROUTINES

THERE ARE OTHER COMMONLY USED ROUTINES AS LISTED BELOW.

**STYPE:**

TYPE ROUTINE FOR TYPING OUT ASCII STRINGS.  
LOCATED AT "STYPE"  
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.

MAINDEC-11-DZRKJ-D MACY11 27(1006) 04-OCT-76 13:08  
 DZRKJD.P11 30-AUG-76 14:48 TABLE OF CONTENTS

SEQ 0013

22	OPERATIONAL SWITCH SETTINGS
49	BASIC DEFINITIONS
159	TRAP CATCHER
168	STARTING ADDRESS(ES)
170	ACT11 HOOKS
181	COMMON TAGS
271	ERROR POINTER TABLE
478	INITIALIZE THE COMMON TAGS
515	TYPE PROGRAM NAME
520	GET VALUE FOR SOFTWARE SWITCH REGISTER
612	T1 TEST THAT ALL RK11 REGISTERS CAN BE REFERENCED
651	T2 CHECK RK11 INITIALIZATION
677	T3 TEST RKCS FUNCTION BITS - 1,2,3
700	T4 TEST RKCS EXTENDED MEMORY BITS - 4,5
722	T5 CHECK RKCS IDE BIT - 6
780	T6 CHECK RKCS SSE, EXB, FMT, IBA BITS - 8,9,10,11
804	T7 CHECK READ ONLY BITS OF RKCS
826	T10 CHECK THAT 'GO' BIT (0) CAN BE SET
865	T11 CHECK RKCS WITH A COUNT PATTERN
906	T12 CHECK THAT RKWC BIT 0-15 CAN BE SET
929	T13 CHECK RKWC WITH A COUNT PATTERN
963	T14 CHECK THAT RKBA CAN BE SET
988	T15 CHECK RKBA WITH A COUNT PATTERN
1023	T16 CHECK THAT RKDA CAN BE SET
1047	T17 CHECK RKDA WITH A COUNT PATTERN
1082	T20 CHECK THAT RKWC, RKBA, RKDA CAN BE CLEARED BY RESET
1107	T21 CHECK THAT RKCS, RKWC, RKBA, RKDA CAN BE CLEARED BY CONTROL RESET
1161	T22 CHECK THAT EACH RK11 REGISTER IS UNIQUELY ADDRESSED
1234	T23 CHECK THAT HI & LO BYTES OF RKCS CAN BE ADDRESSED
1310	T24 CHECK THAT HI & LO BYTES OF RKWC, BA, DA CAN BE ADDRESSED
1406	END OF PASS ROUTINE
1452	GT3RG: ROUTINE FOR GETTING RKCS, RKER, RKDS
1468	GT4RG: ROUTINE FOR GETTING RKCS, RKER, RKDS, RKDA
1480	DELAY: TIME DELAY ROUTINE
1502	CON.RESET: CONTROL REST ROUTINE
1519	CNT.RDY: WAIT FOR CONTROL READY ROUTINE
1563	SCOPE HANDLER ROUTINE
1631	ERROR HANDLER ROUTINE
1667	ERROR MESSAGE TIMEOUT ROUTINE
1715	TTY INPUT ROUTINE
1855	TYPE ROUTINE
1926	CONVERT BINARY TO DECIMAL AND TYPE ROUTINE
1994	BINARY TO OCTAL (ASCII) AND TYPE
2072	TRAP DECODER
2095	TRAP TABLE
2122	POWER DOWN AND UP ROUTINES
2170	ERROR MESSAGES
2307	ERROR DATA POINTERS
2324	ERROR HEADERS

11  
12  
13  
14  
15  
16  
17  
18  
19  
20  
21  
22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51  
52  
53  
54  
55  
56

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

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

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

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

```
.******
.SBTTL BASIC DEFINITIONS
```

001100

```
.*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
```

000011

```
.*MISCELLANEOUS DEFINITIONS
HT= 11      ;;CODE FOR HORIZONTAL TAB
```

57	000012	LF= 12	:: CODE FOR LINE FEED
58	000015	CR= 15	:: CODE FOR CARRIAGE RETURN
59	000200	CRLF= 200	:: CODE FOR CARRIAGE RETURN-LINE FEED
60	177776	PS= 177776	:: PROCESSOR STATUS WORD
61		.EQUIV PS,PSW	
62	177774	STKLMT= 177774	:: STACK LIMIT REGISTER
63	177772	PIRQ= 177772	:: PROGRAM INTERRUPT REQUEST REGISTER
64	177570	DSWR= 177570	:: HARDWARE SWITCH REGISTER
65	177570	DDISP= 177570	:: HARDWARE DISPLAY REGISTER
66			
67		:*GENERAL PURPOSE REGISTER DEFINITIONS	
68	000000	R0= %0	:: GENERAL REGISTER
69	000001	R1= %1	:: GENERAL REGISTER
70	000002	R2= %2	:: GENERAL REGISTER
71	000003	R3= %3	:: GENERAL REGISTER
72	000004	R4= %4	:: GENERAL REGISTER
73	000005	R5= %5	:: GENERAL REGISTER
74	000006	R6= %6	:: GENERAL REGISTER
75	000007	R7= %7	:: GENERAL REGISTER
76	000006	SP= %6	:: STACK POINTER
77	000007	PC= %7	:: PROGRAM COUNTER
78			
79		:*PRIORITY LEVEL DEFINITIONS	
80	000000	PR0= 0	:: PRIORITY LEVEL 0
81	000040	PR1= 40	:: PRIORITY LEVEL 1
82	000100	PR2= 100	:: PRIORITY LEVEL 2
83	000140	PR3= 140	:: PRIORITY LEVEL 3
84	000200	PR4= 200	:: PRIORITY LEVEL 4
85	000240	PR5= 240	:: PRIORITY LEVEL 5
86	000300	PR6= 300	:: PRIORITY LEVEL 6
87	000340	PR7= 340	:: PRIORITY LEVEL 7
88			
89		:*"SWITCH REGISTER" SWITCH DEFINITIONS	
90	100000	SW15= 100000	
91	040000	SW14= 40000	
92	020000	SW13= 20000	
93	010000	SW12= 10000	
94	004000	SW11= 4000	
95	002000	SW10= 2000	
96	001000	SW09= 1000	
97	000400	SW08= 400	
98	000200	SW07= 200	
99	000100	SW06= 100	
100	000040	SW05= 40	
101	000020	SW04= 20	
102	000010	SW03= 10	
103	000004	SW02= 4	
104	000002	SW01= 2	
105	000001	SW00= 1	
106		.EQUIV SW09,SW9	
107		.EQUIV SW08,SW8	
108		.EQUIV SW07,SW7	
109		.EQUIV SW06,SW6	
110		.EQUIV SW05,SW5	
111		.EQUIV SW04,SW4	
112		.EQUIV SW03,SW3	



```

113 .EQUIV SW02,SW2
114 .EQUIV SW01,SW1
115 .EQUIV SW00,SW0
116
117 ;*DATA BIT DEFINITIONS (BIT00 TO BIT15)
118 100000 BIT15= 100000
119 040000 BIT14= 40000
120 020000 BIT13= 20000
121 010000 BIT12= 10000
122 004000 BIT11= 4000
123 002000 BIT10= 2000
124 001000 BIT09= 1000
125 000400 BIT08= 400
126 000200 BIT07= 200
127 000100 BIT06= 100
128 000040 BIT05= 40
129 000020 BIT04= 20
130 000010 BIT03= 10
131 000004 BIT02= 4
132 000002 BIT01= 2
133 000001 BIT00= 1
134 .EQUIV BIT09,BIT9
135 .EQUIV BIT08,BIT8
136 .EQUIV BIT07,BIT7
137 .EQUIV BIT06,BIT6
138 .EQUIV BIT05,BIT5
139 .EQUIV BIT04,BIT4
140 .EQUIV BIT03,BIT3
141 .EQUIV BIT02,BIT2
142 .EQUIV BIT01,BIT1
143 .EQUIV BIT00,BIT0
144
145 ;*BASIC "CPU" TRAP VECTOR ADDRESSES
146 000004 ERRVEC= 4 ;: TIME OUT AND OTHER ERRORS
147 000010 RESVEC= 10 ;: RESERVED AND ILLEGAL INSTRUCTIONS
148 000014 TBITVEC=14 ;: "T" BIT
149 000014 TRTVEC= 14 ;: TRACE TRAP
150 000014 BPTVEC= 14 ;: BREAKPOINT TRAP (BPT)
151 000020 IOTVEC= 20 ;: INPUT/OUTPUT TRAP (IOT) **SCOPE**
152 000024 PWRVEC= 24 ;: POWER FAIL
153 000030 EMTVEC= 30 ;: EMULATOR TRAP (EMT) **ERROR**
154 000034 TRAPVEC=34 ;: "TRAP" TRAP
155 000060 TKVEC= 60 ;: TTY KEYBOARD VECTOR
156 000064 TPVEC= 64 ;: TTY PRINTER VECTOR
157 000240 PIRQVEC=240 ;: PROGRAM INTERRUPT REQUEST VECTOR
158 .SBTTL TRAP CATCHER
159
160 000000 .=0
161 ;*ALL UNUSED LOCATIONS FROM 4 - 776 CONTAIN A ".+2,HALT"
162 ;*SEQUENCE TO CATCH ILLEGAL TRAPS AND INTERRUPTS
163 ;*LOCATION 0 CONTAINS 0 TO CATCH IMPROPERLY LOADED VECTORS
164 000174 .=174
165 000174 000000 DISPREG: .WORD 0 ;: SOFTWARE DISPLAY REGISTER
166 000176 000000 SWREG: .WORD 0 ;: SOFTWARE SWITCH REGISTER
167 .SBTTL STARTING ADDRESS(ES)
168 000200 000137 001542 JMP 2#START ;: JUMP TO STARTING ADDRESS OF PROGRAM

```

169  
170  
171  
172  
173  
174  
175  
176  
177  
178  
179

000204  
000046  
005370  
000052  
000052  
000000  
000204

.SBTTL ACT11 HOOKS

::\*\*\*\*\*  
:HOOKS REQUIRED BY ACT11

SSVPC=  
.=46  
SENDAD  
.=52  
.WORD 0  
.=SSVPC

;SAVE PC  
::1)SET LOC.46 TO ADDRESS OF SENDAD IN .SEOP  
::2)SET LOC.52 TO ZERO  
:: RESTORE PC

.SBTTL COMMON TAGS

180  
181  
182  
183  
184  
185  
186  
187  
188  
189  
190  
191  
192  
193  
194  
195  
196  
197  
198  
199  
200  
201  
202  
203  
204  
205  
206  
207  
208  
209  
210  
211  
212  
213  
214  
215  
216  
217  
218  
219  
220  
221  
222  
223  
224  
225  
226  
227  
228  
229  
230  
231  
232  
233  
234  
235

001100  
001100 000000  
001102 000  
001103 000  
001104 000000  
001106 000000  
001110 000000  
001112 000000  
001114 000  
001115 001  
001116 000000  
001120 000000  
001122 000000  
001124 000000  
001126 000000  
001130 000000  
001132 000000  
001134 000  
001135 000  
001136 000000  
001140 177570  
001142 177570  
001144 177560  
001146 177562  
001150 177564  
001152 177566  
001154 000  
001155 002  
001156 012  
001157 000  
001160 000000  
001162 000000  
001164 000000  
001166 000000  
001170 000000  
001172 000000  
001174 000000  
001176 000000  
001200 000000  
001202 000000  
001204 000000  
001206 000000  
001210 000000  
001212 077  
001213 015  
001214 000012  
001216 005015 047103 020124

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

.=1100  
\$CMTAG: .WORD 0  
\$PASS: .WORD 0  
\$STNM: .BYTE 00  
\$ERFLG: .BYTE 00  
\$ICNT: .WORD 00  
\$LPADR: .WORD 0  
\$LPERR: .WORD 00  
\$ERTTL: .WORD 00  
\$ITEMB: .BYTE 0  
\$ERMAX: .BYTE 1  
\$ERRPC: .WORD 0  
\$GDADR: .WORD 0  
\$BDADR: .WORD 0  
\$GDDAT: .WORD 0  
\$BDDAT: .WORD 0  
\$AUTOB: .BYTE 0  
\$INTAG: .BYTE 0  
\$SWR: .WORD DSWR  
\$DISPLAY: .WORD DDISP  
\$TKS: 177560  
\$TKB: 177562  
\$TPS: 177564  
\$TPB: 177566  
\$NULL: .BYTE 0  
\$FILLS: .BYTE 2  
\$FILLC: .BYTE 12  
\$TPFLG: .BYTE 0  
\$REGAD: .WORD 0  
\$REG0: .WORD 0  
\$REG1: .WORD 0  
\$REG2: .WORD 0  
\$REG3: .WORD 0  
\$REG4: .WORD 0  
\$REG5: .WORD 0  
\$REG6: .WORD 0  
\$REG7: .WORD 0  
\$REG10: .WORD 0  
\$REG11: .WORD 0  
\$TIMES: 0  
\$ESCAPE: 0  
\$QUES: .ASCII /?/  
\$CRLF: .ASCII <15>  
\$LF: .ASCII <12>  
MSG3: .ASCII <15><12>/CNT RDY DIDN'T SET/

:: START OF COMMON TAGS  
:: CONTAINS PASS COUNT  
:: CONTAINS THE TEST NUMBER  
:: CONTAINS ERROR FLAG  
:: CONTAINS SUBTEST ITERATION COUNT  
:: CONTAINS SCOPE LOOP ADDRESS  
:: CONTAINS SCOPE RETURN FOR ERRORS  
:: CONTAINS TOTAL ERRORS DETECTED  
:: CONTAINS ITEM CONTROL BYTE  
:: CONTAINS MAX. ERRORS PER TEST  
:: CONTAINS PC OF LAST ERROR INSTRUCTION  
:: CONTAINS ADDRESS OF 'GOOD' DATA  
:: CONTAINS ADDRESS OF 'BAD' DATA  
:: CONTAINS 'GOOD' DATA  
:: CONTAINS 'BAD' DATA  
:: RESERVED--NOT TO BE USED  
:: AUTOMATIC MODE INDICATOR  
:: INTERRUPT MODE INDICATOR  
:: ADDRESS OF SWITCH REGISTER  
:: ADDRESS OF DISPLAY REGISTER  
:: TTY KBD STATUS  
:: TTY KBD BUFFER  
:: TTY PRINTER STATUS REG. ADDRESS  
:: TTY PRINTER BUFFER REG. ADDRESS  
:: CONTAINS NULL CHARACTER FOR FILLS  
:: CONTAINS # OF FILLER CHARACTERS REQUIRED  
:: INSERT FILL CHARS. AFTER A "LINE FEED"  
:: "TERMINAL AVAILABLE" FLAG (BIT<07>=0=YES)  
:: CONTAINS THE ADDRESS FROM  
:: WHICH (\$REG0) WAS OBTAINED  
:: CONTAINS ((\$REGAD)+0)  
:: CONTAINS ((\$REGAD)+2)  
:: CONTAINS ((\$REGAD)+4)  
:: CONTAINS ((\$REGAD)+6)  
:: CONTAINS ((\$REGAD)+10)  
:: CONTAINS ((\$REGAD)+12)  
:: CONTAINS ((\$REGAD)+14)  
:: CONTAINS ((\$REGAD)+16)  
:: CONTAINS ((\$REGAD)+20)  
:: CONTAINS ((\$REGAD)+22)  
:: MAX. NUMBER OF ITERATIONS  
:: ESCAPE ON ERROR ADDRESS  
:: QUESTION MARK  
:: CARRIAGE RETURN  
:: LINE FEED

236	001224	042122	020131	044504
237	001232	047104	052047	051440
238	001240	052105	000	
239		001244		
240				
241				
242				
243				
244				
245				
246				
247	001244	177400		
248	001246	177402		
249	001250	177404		
250	001252	177406		
251	001254	177410		
252	001256	177412		
253	001260	177416		
254				
255				
256				
257				
258				
259				
260	001262	000000		
261	001264	000000		
262	001266	000200		
263				
264				
265				
266				
267	001270	000220		
268				
269				

.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

270  
271  
272  
273  
274  
275  
276  
277  
278  
279  
280  
281  
282  
283  
284  
285  
286  
287  
288  
289  
290  
291  
292  
293  
294  
295  
296  
297  
298  
299  
300  
301  
302  
303  
304  
305  
306  
307  
308  
309  
310  
311  
312  
313  
314  
315  
316  
317  
318  
319  
320  
321  
322  
323  
324  
325

001272

.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

SERRTB:

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

```
TIME OUT ON RK11 REG
PC      REG
***** 177400
```

NOTE THAT \*\*\*\*\* WOULD BE THE ACTUAL PC WHERE 'ERROR 1' IS LOCATED.

THE ERROR HANDLER IS LOCATED AT 'SERROR'. 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

```

326 ;ITEM 1
327
328 001272 010316 EM1 ;TIME OUT ON RK11 REG
329 001274 011526 DH1 ;PC REG
330 001276 011456 DT1 ;$ERRPC $REGO
331 001300 000000 0
332
333 ;ITEM 2
334
335 001302 010350 EM2 ;REGISTER NOT CLEARED
336 001304 011546 DH2 ;PC REGADD RECVD
337 001306 011464 DT2 ;$ERRPC $REGO $REG1
338 001310 000000 0
339
340 ;ITEM 3
341
342 001312 010375 EM3 ;RKCS ERROR
343 001314 011623 DH3 ;PC WROTE READ
344 001316 011464 DT2 ;$ERRPC $REGO $REG1
345 001320 000000 0
346
347 ;ITEM 4
348
349 001322 010410 EM4 ;RKCS ERROR-ON WRITING READ ONLY BITS
350 001324 011575 DH4 ;PC EXPCT RECVD
351 001326 011464 DT2 ;$ERRPC $REGO $REG1
352 001330 000000 0
353
354 ;ITEM 5
355
356 001332 010455 EM5 ;BUS INIT DID NOT CLEAR RKCS
357 001334 011650 DH5 ;PC RECVD
358 001336 011456 DT1 ;$ERRPC $REGO
359 001340 000000 0
360
361 ;ITEM 6
362
363 001342 010511 EM6 ;'CNTRL RESET' DIDN'T CLEAR RKCS, ON SETING GO
364 001344 011650 DH5 ;PC RECVD
365 001346 011456 DT1 ;$ERRPC $REGO
366 001350 000000 0
367
368 ;ITEM 7
369
370 001352 010567 EM7 ;'CNTRL ROY' DIDN'T SET AFTER CONTROL RESET
371 001354 012201 DH30 ;PC RKCS RKER RKDS
372 001356 011514 DT26 ;$ERRPC $REGO $REG1 $REG2
373 001360 000000 0
374
375 ;ITEM 10
376
377 001362 010636 EM10 ;REGISTER NOT CLEARED
378 001364 011546 DH2 ;PC REGADD RECVD
379 001366 011464 DT2 ;$ERRPC $REGO $REG1
380 001370 000000 0
381

```

382			;ITEM 11			
383						
384	001372	010663		EM11	;RKWC ERROR	
385	001374	011666		DH11	;PC WROTE	READ
386	001376	011464		DT2	;SERRPC \$REG0	\$REG1
387	001400	000000		0		
388						
389			;ITEM 12			
390						
391	001402	011423		EM43	;UNEXPECTED RK11 INTERRUPT	
392	001404	011713		DH21	;PC	
393	001406	011510		DT21	;SERRPC	
394	001410	000000		0		
395						
396			;ITEM 13			
397						
398	001412	010676		EM13	;RKBA ERROR	
399	001414	011666		DH11	;PC WROT	READ
400	001416	011464		DT2	;SERRPC \$REG0	\$REG1
401	001420	000000		0		
402						
403			;ITEM 14			
404						
405	001422	010711		EM14	;CNTRL RESET DID NOT CLEAR REGISTER	
406	001424	011546		DH2	;PC REGADD	RECVD
407	001426	011464		DT2	;SERRPC \$REG0	\$REG1
408	001430	000000		0		
409						
410			;ITEM 15			
411						
412	001432	010753		EM15	;RKDA ERROR	
413	001434	011666		DH11	;PC WROTE	READ
414	001436	011464		DT2	;SERRPC \$REG0	\$REG1
415	001440	000000		0		
416						
417			;ITEM 16			
418						
419	001442	011301		EM26	;RKCS ALTERED ON CLEARING 'REG-BYTE'	
420	001444	012067		DH26	;PC REG-BYTE (RKCS)EXP (RKCS)RECVD	
421	001446	011514		DT26	;SERRPC \$REG0	\$REG1 \$REG2
422	001450	000000		0		
423						
424			;ITEM 17			
425						
426	001452	010766		EM17	;BUS INIT DIDN'T CLEAR REGISTER	
427	001454	011546		DH2	;PC REGADD	RECVD
428	001456	011464		DT2	;SERRPC \$REG0	\$REG1
429	001460	000000		0		
430						
431			;ITEM 20			
432						
433	001462	011022		EM20	;ADDRESSING ERROR-TRIED TO ADDRESS REG1, GOT REG2	
434	001464	011720		DH20	;PC REG1 REG2 (REG1) (REG2)	
435	001466	011474		DT20	;SERRPC \$REG0	\$REG1 \$REG2 \$REG3
436	001470	000000		0		
437						

```

438 ;ITEM 21
439
440 001472 011345 EM27 ;TRIED TO CLEAR 'REG-BYTE' CHANGED 'REG-BYT2'
441 001474 012131 DH27 ;PC REG-BYT1 REG-BYT2 BYT2-EXPCT BYT2-RECVD
442 001476 011474 DT20 ;SERRPC $REG0 $REG1 $REG2 $REG3
443 001500 000000 0
444
445 ;ITEM 22
446
447 001502 011103 EM22 ;DID NOT CLEAR RKCS LO BYTE
448 001504 011575 DH4 ;PC EXPCT RECVD
449 001506 011464 DT2 ;SERRPC $REG0 $REG1
450 001510 000000 0
451
452 ;IEM 23
453
454 001512 011136 EM23 ;DID NOT CLEAR RKCS HI BYTE
455 001514 011575 DH4 ;PC EXPCT RECVD
456 001516 011464 DT2 ;SERRPC $REG0 $REG1
457 001520 000000 0
458
459 ;ITEM 24
460
461 001522 011172 EM24 ;TRIED TO CLEAR RKCS 'BYTE', CHANGED 'REGIS'
462 001524 011767 DH24 ;PC BYTE REGIS (REG)EXP (REG)RECVD
463 001526 011474 DT20 ;SERRPC $REG0 $REG1 $REG2 $REG3
464 001530 000000 0
465
466 ;ITEM 25
467
468 001532 011246 EM25 ;FAILED TO CLEAR 'REG-BYTE'
469 001534 012041 DH25 ;PC REG-BYTE RECVD
470 001536 011464 DT2 ;SERRPC $REG0 $REG1
471 001540 000000 0
472
473
474
475
476 001542 000005 START: RESET ;CLEAR THE BUS
477 .SBTTL INITIALIZE THE COMMON TAGS
478 ;;CLEAR THE COMMON TAGS ($CMTAG) AREA
479 001544 012706 001100 MOV #CMTAG,R6 ;;FIRST LOCATION TO BE CLEARED
480 001550 005026 CLR (R6)+ ;;CLEAR MEMORY LOCATION
481 001552 022706 001140 CMP #SWR,R6 ;;DONE?
482 001556 001374 BNE -6 ;;LOOP BACK IF NO
483 001560 012706 001100 MOV #STACK,SP ;;SETUP THE STACK POINTER
484 ;;INITIALIZE A FEW VECTORS
485 001564 012737 005632 000020 MOV #SCOPE,@IOTVEC ;;IOT VECTOR FOR SCOPE ROUTINE
486 001572 012737 000340 000022 MOV #340,@IOTVEC+2 ;;LEVEL 7
487 001600 012737 006104 000030 MOV #ERROR,@EMTVEC ;;EMT VECTOR FOR ERROR ROUTINE
488 001606 012737 000340 000032 MOV #340,@EMTVEC+2 ;;LEVEL 7
489 001614 012737 010046 000034 MOV #STRAP,@TRAPVEC ;;TRAP VECTOR FOR TRAP CALLS
490 001622 012737 000340 000036 MOV #340,@TRAPVEC+2 ;;LEVEL 7
491 001630 012737 010134 000024 MOV #SPWRDN,@PWRVEC ;;POWER FAILURE VECTOR
492 001636 012737 000340 000026 MOV #340,@PWRVEC+2 ;;LEVEL 7
493 001644 005037 001206 CLR $TIMES ;;INITIALIZE NUMBER OF ITERATIONS

```



```

494 001650 005037 001210          CLR      $ESCAPE          ;; CLEAR THE ESCAPE ON ERROR ADDRESS
495 001654 112737 000001 001115      MOV      #1,$ERMAX        ;; ALLOW ONE ERROR PER TEST
496 001662 012737 001662 001106      MOV      #,$SLPADR        ;; INITIALIZE THE LOOP ADDRESS FOR SCOPE
497 001670 012737 001670 001110      MOV      #,$SLPERR        ;; SETUP THE ERROR LOOP ADDRESS
498
499                                     ;; SIZE FOR A HARDWARE SWITCH REGISTER. IF NOT FOUND OR IT IS
500                                     ;; EQUAL TO A "-1" SETUP FOR A SOFTWARE SWITCH REGISTER.
501 001676 013746 000004          MOV      @#ERRVEC, -(SP)   ;; SAVE ERROR VECTOR
502 001702 012737 001736 000004      MOV      #64$, @#ERRVEC   ;; SET UP ERROR VECTOR
503 001710 012737 177570 001140      MOV      #DSWR, SWR        ;; SETUP FOR A HARDWARE SWICH REGISTER
504 001716 012737 177570 001142      MOV      #DDISP, DISPLAY   ;; AND A HARDWARE DISPLAY REGISTER
505 001724 022777 177777 177206      CMP      #-1, @SWR         ;; TRY TO REFERENCE HARDWARE SWR
506 001732 001012          BNE      66$              ;; BRANCH IF NO TIMEOUT TRAP OCCURRED
507                                     ;; AND THE HARDWARE SWR IS NOT = -1
508 001734 000403          BR       65$              ;; BRANCH IF NO TIMEOUT
509 001736 012716 001744          64$:   MOV      #65$, (SP)   ;; SET UP FOR TRAP RETURN
510 001742 000002          RTI
511 001744 012737 000176 001140      65$:   MOV      #SWREG, SWR    ;; POINT TO SOFTWARE SWR
512 001752 012737 000174 001142      MOV      #DISPREG, DISPLAY
513 001760 012637 000004          66$:   MOV      (SP)+, @#ERRVEC ;; RESTORE ERROR VECTOR
514
515 .SBTTL TYPE PROGRAM NAME
516 ;;TYPE THE NAME OF THE PROGRAM IF FIRST PASS
517 001764 005227 177777          INC      #-1              ;; FIRST TIME?
518 001770 001043          BNE      67$              ;; BRANCH IF NO
519 001772 104401 002030          TYPE    ,68$              ;; TYPE ASCIZ STRING
520 .SBTTL GET VALUE FOR SOFTWARE SWITCH REGISTER
521 001776 005737 000042          TST     @#42              ;; ARE WE RUNNING UNDER XXDP/ACT?
522 002002 001006          BNE      69$              ;; BRANCH IF YES
523 002004 023727 001140 000176      CMP     SWR, #SWREG        ;; SOFTWARE SWITCH REG SELECTED?
524 002012 001005          BNE      70$              ;; BRANCH IF NO
525 002014 104406          GTSWR                    ;; GET SOFT-SWR SETTINGS
526 002016 000403          BR       70$
527 002020 112737 000001 001134      69$:   MOV      #1,$AUTOB      ;; SET AUTO-MODE INDICATOR
528 002026 000424          70$:   BR       67$
529                                     ;; GET OVER THE ASCIZ
530 002100          ;;68$: .ASCIZ <CRLF>/RK11 LOGIC TEST I/<15><12>/MAINDEC-11-DZRKJ-D/<CRLF>
531 67$:
532 002100 012737 002116 000004      START1: MOV     #BADTMO, @#4 ;; SET TIME OUT VECTOR FOR UNEXPECTED
533                                     ;; TIME OUTS
534 002106 012777 002202 177154      MOV     #BADINT, @RKVEC   ;; SET UP RK11 INTERRUPT VECTOR FOR
535                                     ;; UNEXPECTED INTERRUPTS FROM RK11
536 002114 000516          BR      TST1              ;; GO TO TEST 1
537
538
539
540
541                                     ; THIS ROUTINE HANDLES UNEXPECTED TIME OUTS
542
543 002116 011600          BADTMO: MOV     (SP), RO    ;; SAVE PC WHERE TIME OUT OCCURED
544 002120 005740          TST     -(RO)
545 002122 022626          CMP     (SP)+, (SP)+     ;; RESTORE STACK POINTER
546 002124 104401 002132          TYPE    ,65$            ;; TYPE ASCIZ STRING
547 002130 000417          BR      64$              ;; GET OVER THE ASCIZ
548                                     ;;65$: .ASCIZ <15><12>/UNEXPECTED TIME OUT AT PC=/
549 002170          64$:

```

```

550 002170 010046      MOV      RO,-(SP)      ;SET UP FOR TYPING OUT PC
551 002172 104402      TYPOC                    ;GO TYPE OUT OCTAL PC
552 002174 000000      HALT
553 002176 000137 001542    JMP      @#START
554
555                      ;THIS ROUTINE HANDLES UNEXPECTED INTERRUPTS FROM RK11
556                      ;SW 9 AND 10 FOR LOOPING ON ERROR
557                      ;AND LOOPING ON TEST IN WHICH TIMEOUT
558                      ;OCCURRED, ARE PROVIDED.
559
560 002202 011600      BADINT: MOV      (SP),RO      ;SAVE PC WHERE INTERRUPT OCCURED
561 002204 005740      TST      -(RO)
562 002206 032777 020000 176724  BIT      #20000,@SWR      ;INHIBIT ERROR TYPEOUT?
563 002214 001015      BNE      1$              ;YES, DON'T TYPE OUT
564 002216 104401      TYPE
565 002220 001213      $CRLF
566 002222 104401      TYPE
567 002224 011423      EM43                    ;TYPE 'UNEXPEXED RK11 INTERRUPT'
568                      ;TYPE ' AT PC='
569 002226 104401 002234      TYPE      65$           ;:TYPE ASCIZ STRING
570 002232 000404      BR      64$             ;:GET OVER THE ASCIZ
571                      ;:65$: .ASCIZ / AT PC=/
572                      64$:
573 002244 010046      MOV      RO,-(SP)      ;SET UP FOR TYPING OUT PC
574 002246 104402      TYPOC                    ;GO TYPE OCTAL PC WHERE BAD
575                      ;INTERUPT OCCURED
576 002250 032777 001000 176662 1$: BIT      #1000,@SWR      ;LOOP ON ERROR?
577 002256 001403      BEQ      2$              ;NO, BRANCH
578 002260 022626      CMP      (SP)+,(SP)+    ;YES, REPOSITION STACK
579 002262 000177 176620      JMP      @SLPADR        ;GO TO THE STARTING ADDRESS OF
580                      ;THE TEST THAT GAVE UNEXPECTED INTERRUPT
581 002266 032777 040000 176644 2$: BIT      #40000,@SWR      ;LOOP ON TEST?
582 002274 001401      BEQ      3$              ;NO, BRANCH
583 002276 000002      RTI
584 002300 000000      3$: HALT                ;YES, LOOP. GO BACK WHER U INTERRUPTED FROM.
585                      ;UNEXPEXED INTERRUPT OCCURED AS
586                      ;INDICATED IN THE TYPE OUT.U CAN LOOP
587                      ;ON ERROR, TEST,OR INHIBIT TYPEOUT BY
588 002302 000137 001542    JMP      @#START        ;SETTING APPROPRIATE SWITCHES.
589                      ;GO BACK TO THE START OF THE
590                      ;PROGRAM. THUS PRESSING CONTINUE
591                      ;AFTER THE ABOVE HALT WILL
592                      ;RESTART THE PROGRAM
593
594
595                      ;RESTART AFTER POWER FAIL
596                      ;THE PROGRAM WOULD RESTART HERE IF POWER CAME BACK AFTER A FALIURE.
597
598 002306 002314      PFSTRT:
599 002306 104401 002314      TYPE      65$           ;:TYPE ASCIZ STRING
600 002312 000411      BR      64$             ;:GET OVER THE ASCIZ
601                      ;:65$: .ASCIZ <15><12>/PWR UP,RESTART/
602                      64$:
603 002336 005000      CLR      RO
604 002340 005001      CLR      R1
605 002342 005201      1$: INC      R1

```



```

662 002524 001004          BNE      3$          ;BRANCH IF ERROR
663 002526 000411          BR       4$
664 002530 005771 000000 2$:  TST     2(R1)      ;CHECK THAT REST OF REGISTERS
665                                ;ARE CLEAR
666 002534 001406          BEQ     4$          ;BRANCH IF REGISTER IS CLR
667 002536 011137 001162 3$:  MOV     2R1,$REG0   ;GET ADRES OF REGISTER
668 002542 017137 000000 001164 MOV     2(R1),$REG1 ;GET CONTENTS OF REGISTER
669 002550 104002          ERROR    2          ;RK11 REGISTER WAS FOUND TO B NOT CLEAR
670 002552 005721 4$:  TST     (R1)+      ;INCREMNT POINTER TO NXT REG
671 002554 005200          INC     R0          ;CHKD ALL REGS?
672 002556 001354          BNE     1$          ;IF NOT LUP BAK
673                                ;
674                                ;*****
675                                ;*TEST 3      TEST RKCS FUNCTION BITS - 1,2,3
676                                ;*THIS TEST CHECKS IF THE FUNCTION BIT-1,2,3 IN RKCS CAN BE WRITTEN &
677                                ;*READ BACK.  R0 CONTAINS THE BIT THAT WAS WRITTEN.  R1 CONTAINS THE
678                                ;*BITS THAT WAS/WERE READ BACK.
679                                ;*****
680 002560 000004          †ST3:  SCOPE
681 002562 012737 002574 001110 MOV     #1$,$LPERR  ;SET RETURN ADRES FOR LOPING ON
682                                ;ERROR (SW 9)
683 002570 012700 000002 1$:  MOV     #2,R0      ;INITIALIZE BIT TO BE WRITTEN IN RKCS
684 002574 010077 176450  MOV     R0,2RKCS   ;WRITE THAT BIT IN RKCS
685 002600 017701 176444  MOV     2RKCS,R1   ;GET RKCS
686 002604 042701 000200  BIC     #200,R1    ;MASK OUT CNTRL RDY BIT
687 002610 020001  CMP     R0,R1      ;WAS RECVD BIT SAME AS WRITTEN BIT
688 002612 001406          BEQ     2$          ;YES, BRANCH OTHERWISE REPORT ERROR
689 002614 010037 001162  MOV     R0,$REG0   ;GET EXPCTD RKCS, BIT THAT WAS WRITTEN
690 002620 017737 176424 001164 MOV     2RKCS,$REG1 ;GET RECVD RKCS, BIT READ
691 002626 104003          ERROR    3          ;BIT THAT WAS WRITTEN WAS NOT READ BACK
692 002630 006300 2$:  ASL     R0          ;SHIFT TO WRITE NEXT BIT
693 002632 020027 000020  CMP     R0,#20     ;HAVE U CHKD ALL 3 BITS 1, 2, 3
694 002636 001356          BNE     1$          ;IF NOT, LOOP BACK & CHECK THE NXT BIT
695                                ;
696                                ;*****
697                                ;*TEST 4      TEST RKCS EXTENDED MEMORY BITS - 4,5
698                                ;*THIS TEST CHECKS IF THE 'EXTENDED MEMORY' BITS CAN BE WRITTEN
699                                ;*AND READ BACK CORRECTLY.
700                                ;*****
701 002640 000004          †ST4:  SCOPE
702 002642 012737 002654 001110 MOV     #1$,$LPERR  ;SET RETURN ADRES FOR LUPING
703                                ;ON EROR (SW 9)
704 002650 012700 000020 1$:  MOV     #20,R0     ;INITIALIZE BIT TO BE WRITTEN IN RKCS
705 002654 010077 176370  MOV     R0,2RKCS   ;WRITE THAT BIT IN RKCS
706 002660 017701 176364  MOV     2RKCS,R1   ;GET RKCS
707 002664 042701 000200  BIC     #200,R1    ;MASK OUT CNTRL RDY BIT
708 002670 020001  CMP     R0,R1      ;WAS RECVD BIT SAME AS WRITTEN BIT
709 002672 001406          BEQ     2$          ;YES, BRANCH
710 002674 010037 001162  MOV     R0,$REG0   ;GET EXPCTD RKCS, BIT THAT WAS WRITTEN
711 002700 017737 176344 001164 MOV     2RKCS,$REG1 ;GET RKCD RECVD, BIT THAT WAS READ
712 002706 104003          ERROR    3          ;BIT THAT WAS WRITTEN WAS NOT READ BACK
713 002710 006300 2$:  ASL     R0          ;SHIFT TO WRITE NEXT BIT
714 002712 022700 000100  CMP     #100,R0    ;HAVE U CHKD BOTH BITS, 4, 5
715 002716 001356          BNE     1$          ;IF NOT, LOOP BACK & CHECK THE NXT BIT
716                                ;
717                                ;*****

```

718  
719  
720  
721  
722  
723  
724  
725  
726  
727  
728  
729  
730  
731  
732  
733  
734  
735  
736  
737  
738  
739  
740  
741  
742  
743  
744  
745  
746  
747  
748  
749  
750  
751  
752  
753  
754  
755  
756  
757  
758  
759  
760  
761  
762  
763  
764  
765  
766  
767  
768  
769  
770  
771  
772  
773

;\*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.

\*\*\*\*\*

↑STS: SCOPE  
MOV #340,-(SP)  
MOV #645,-(SP)  
RTI

645: MOV RKCS,RO  
MOV #100,ARO  
CMP #300,ARO  
BEQ 15  
MOV #300,\$REGO  
MOV ARO,\$REGI  
ERROR 3

SET THE IDE BIT  
WAS IT WRITTEN CORRECTLY  
YES, BRANCH OTHERWISE REPORT ERROR  
GET EXPCT RKCS  
GET RKCS RECVD  
IDE BIT WAS WRITTEN, BUT WAS NOT  
READ BACK

15: CNT.RESET

CONTROL RESET, CLEAR IDE  
THIS IS A CALL FOR THE 'CNTRL-  
RESET' ROUTINE. A CONTROL RESET  
IS ISSUED AND AFTER A CERTAIN TIME  
IF THE 'CNTRL RDY' DOES NOT SET  
AN ERROR IS REPORTED. NOTE THAT  
THE PC IN ERROR MESSAGE IS THE  
PC WHERE 'CNT.RESET' IS LOCATED.

CMP #200,ARO  
BEQ 25  
MOV RO,\$REGO  
MOV ARO,\$REGI  
ERROR 10

DID IDE BIT GET CLRD?  
YES, BRANCH  
GET ADRES OF RKCS  
GET RKCS  
IDE BIT COULD NOT B CLRD  
EXIT

25: DELAY ,10

WAIT FOR AT LEAST 60 US  
ON 11/20 12 US FOR 11/45  
SET RK11 INTERUPT VECTOR TO  
WHICH RK11 CAN INTERUPT IF THERE  
IS FAULTY LOGIC

MOV #35,ARKVEC

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

LOWER CPU PRIORITY SO THAT  
RK11 CAN POSSIBLY INTERRUPT IF  
THER IS MALFUNCTIONINIG LOGIC  
THE INTERRUPT WOULD OCCUR

45:

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

655:

MOV #BADINT,ARKVEC  
BR TST6  
35: CMP (SP)+,(SP)+  
ERROR 12

PRIORITY  
SET UP UNEXPCTD INTRUPT VECTOR  
EXIT  
RESTORE STACK  
AN UNEXPECTED RK11 INTERRUPT  
OCCURED PROBABLY DUE TO FAULTY LOGIC

;

```

774
775
776
777
778
779 003072 000004
780 003074 012701 000400
781 003100 013702 001250
782 003104 010112
783 003106 011200
784 003110 042700 000200
785 003114 020100
786
787 003116 001405
788 003120 010137 001162
789 003124 011237 001164
790 003130 104003
791
792 003132 006301
793 003134 022701 010000
794 003140 001361
795
796
797
798
799
800
801
802 003142 000004
803 003144 013700 001250
804 003150 012710 170200
805
806 003154 011001
807 003156 042701 010000
808 003162 022701 000200
809
810 003166 001406
811 003170 012737 000200 001162
812 003176 011037 001164
813 003202 104004
814
815
816
817
818
819
820
821
822
823
824 003204 000004
825 003206 012746 000340
826 003212 012746 003220
827 003216 000002
828 003220
829 003220 013701 001250

```

```

*****
;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
*****
TST6: SCOPE
MOV #400,R1 ;INITIALIZE BIT TO BE WRITTEN IN RKCS
MOV RKCS,R2
1S: 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 2S ;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
2S: ASL R1 ;SHIFT TO WRITE NEXT BIT
CMP #10000,R1 ;HAVE U CHECKED ALL BITS 8, 9, 10, 11
BNE 1S ;IF NOT, LOOP BACK & CHECK THE NXT BIT

*****
;TEST 7 CHECK READ ONLY BITS OF RKCS
;THIS TEST CHECKS THAT TRYING TO SET THE UNUSED BIT OR THE READ ONLY
;BITS DOES NOT SET THEM OR AFFECT ANY OTHER BITS IN RKCS
*****
TST7: 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,$REG0 ;GET EXPCTD RKCS
MOV R0,$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.
*****
TST10: SCOPE
MOV #340,-(SP)
MOV #64$,-(SP)
RTI
64$: MOV RKCS,R1

```

```

830 003224 012711 007576      MOV      #7576,R1      ;SET ALL BITS EXCEPT GO
831 003230 005000              CLR      RO           ;
832 003232 000005              RESET          ;ISSUE BUS INIT
833 003234 022711 000200      CMP      #200,R1     ;CHECK IF RKCS WAS CLEARED?
834 003240 001403              BEQ      1$         ;YES, BRANCH OTHERWISE REPORT ERROR
835 003242 011137 001162      MOV      R1,$REGO    ;GET RKCS
836 003246 104005              ERROR      5        ;BUS INIT DID NOT CLEAR RKCS
837 003250 012711 005000      1$:      MOV      #5000,R1 ;SET IBA & EXB IN RKCS
838 003254 005211              INC      R1         ;SET GO, CONTROL RESET
839 003256 005200      2$:      INC      RO         ;KEEP TIME
840 003260 105700              TSTB     RO         ;HAVE U WAITED LONG FOR CNTRL RDY
841                          ;TO SET?
842 003262 100411              BMI      3$         ;IF YES, BRANCH & REPORT ERROR
843 003264 105711              TSTB     R1         ;WAS CNTRL RDY SET?
844 003266 100373              BPL      2$         ;IF NOT LOOP BACK & WAIT FOR IT
845 003270 022711 000200      CMP      #200,R1     ;IF CNTRL RDY WAS SET, CHK IF 'CNTRL
846                          ;RESET' CLEARED IBA & EXB BITS
847 003274 001407              BEQ      TST11      ;IF YES, EXIT. OTHERWISE ERROR
848 003276 011137 001162      MOV      R1,$REGO    ;GET RKCS
849 003302 104006              ERROR      6        ;GO BIT COULD NOT BE SET OR FAULT IN
850                          ;THE 'INIT L' GENERATING LOGIC
851 003304 000403              BR       TST11      ;EXIT
852 003306 004737 005424      3$:      JSR      PC,GT3RG   ;GO, GET RKCS, ER, DS
853 003312 104007              ERROR      7        ;CONTROL READY DID NOT SET AFTER
854                          ;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 UNDESIREED OPERATION.
: *R1 CONTAINS THE COUNT PATTERN THAT WAS WRITTEN
:*****

```

```

863 003314 000004      TST11:  SCOPE
864 003316 012737 000010 001206      MOV      #10,$TIMES  ;;DO 10 ITERATIONS
865 003324 012746 000340      MOV      #340,-(SP)
866 003330 012746 003336      MOV      #64$,-(SP)
867 003334 000002              RTI
868 003336      64$:
869 003336 013700 001250      MOV      RKCS,RO
870 003342 012710 007576      MOV      #7576,R0    ;SET ALL BITS IN RKCS EXCEPT GO
871 003346 005010              CLR      R0         ;CLEAR RKCS
872 003350 022710 000200      CMP      #200,R0    ;WAS IT CLEARED
873 003354 001405              BEQ      1$         ;YES, BRANCH
874 003356 010037 001162      MOV      RO,$REGO   ;GET ADRES OF RKCS
875 003362 011037 001164      MOV      R0,$REG1   ;NO, GET RKCS
876 003366 104010              ERROR      10       ;RKCS COULD NOT BE CLEARED
877 003370 012701 000002      1$:      MOV      #2,R1      ;WRITE THIS BIT IN RKCS
878 003374 012737 003406 001110      MOV      #2,$SLPERR
879 003402 012705 177773      MOV      #-5,R5
880 003406 010110      2$:      MOV      R1,R0      ;WRITE IT
881 003410 010102              MOV      R1,R2      ;GET BIT THAT WAS WRITTEN
882 003412 052702 000200      BIS      #200,R2    ;SET CNTRL RDY BIT
883 003416 011003              MOV      R0,R3
884 003420 020203      CMP      R2,R3
885 003422 001407              BEQ      3$         ;WAS THAT BIT WRITTEN CORRECTLY?
                        ;YES, BRANCH

```

F03

MAINDEC-11-DZRKJ-D MACY11 27(1006) 04-OCT-76 13:08 PAGE 18  
 DZRKJD.P11 30-AUG-76 14:48 T11 CHECK RKCS WITH A COUNT PATTERN

SEQ 0031

```

886 003424 010237 001162      MOV     R2,$REGO      ;GET EXPCTD WORD
887 003430 010337 001164      MOV     R3,$REG1      ;GET RKCS RECVD
888 003434 104003                ERROR    3             ;DID NOT READ BAK THE BIT THAT
889                                ;WAS WRITTEN
890 003436 005205                INC     R5
891 003440 001405                BEQ     TST12         ;:EXIT
892 003442 062701 000002 3$:  ADD     #2,R1         ;GENERATE NXT PATTERN TO BE WRITTEN
893 003446 022701 010000      CMP     #10000,R1     ;ALL PATTERNS WRITTEN?
894 003452 001355                BNE     2$           ;IF NOT, LUP BAK & CHK NXT PATTERN
895
896 ;*****
897 ;*TEST 12      CHECK THAT RKWC BIT 0-15 CAN BE SET
898 ;*THIS TEST FLOATS A '1' THROUGH RKWC BIT 0-15 AND CHECKS THAT IT
899 ;*CAN BE READ BACK CORRECTLY. R0 CONTAINS THE WORD THAT IS WRITTEN
900 ;*****
901 003454 000004                TST12: SCOPE
902 003456 012700 000001      MOV     #1,R0         ;INITIALIZE R0 FOR THE BIT TO BE WRITTEN IN RKWC
903 003462 013701 001252      MOV     RKWC,R1
904 003466 012737 003474 001110  MOV     #1$,$LPERR    ;SET UP RETURN ADRES FOR
905                                ;LUPING ON ERROR (SW 9)
906 003474 010011 1$:  MOV     R0,R1         ;WRITE THAT BIT IN RKWC
907 003476 011102      MOV     R1,R2
908 003500 020002      CMP     R0,R2         ;WAS IT WRITTEN CORRECTLY
909 003502 001405                BEQ     2$           ;YES, BRANCH. OTHERWISE, ERROR
910 003504 010037 001162      MOV     R0,$REGO     ;GET EXPCTD RKWC BIT THAT WAS WRITTEN
911 003510 010237 001164      MOV     R2,$REG1     ;GET RKWC (THAT WAS READ BACK)
912 003514 104011                ERROR    11         ;DID NOT READ BACK THE BIT THAT
913                                ;WAS WRITTEN IN RKWC
914 003516 006300 2$:  ASL     R0           ;SHIFT TO WRITE THE NXT BIT
915 003520 001365                BNE     1$           ;IF ALL THE BITS HAVE NOT BEEN
916                                ;DONE, LOOP BACK
917
918 ;*****
919 ;*TEST 13      CHECK RKWC WITH A COUNT PATTERN
920 ;*THIS TEST CHECKS THAT RKWC CAN BE CLEARED, THEN A COUNT PATTERN
921 ;*FROM 0 TO 177777 IS WRITTEN & CHECKED IF IT WAS WRITTEN CORRECTLY
922 ;*****
923 003522 000004                TST13: SCOPE
924 003524 012737 000010 001206  MOV     #10,$TIMES   ;;DO 10 ITERATIONS
925 003532 013700 001252      MOV     RKWC,R0
926 003536 012710 177777      MOV     #177777,R0   ;SET ALL BITS IN RKWC
927 003542 005010                CLR     R0           ;CLEAR RKWC
928 003544 005710                TST     R0           ;WAS IT CLEARED?
929 003546 001405                BEQ     1$           ;YES, BRANCH
930 003550 010037 001162      MOV     R0,$REGO     ;GET ADRES OF RKWC
931 003554 011037 001164      MOV     R0,$REG1     ;NO, GET RKWC
932 003560 104010                ERROR    10         ;RKWC COULD NOT BE CLEARED
933
934 003562 005001 1$:  CLR     R1           ;INITIALIZE COUNT PATTERN
935 003564 012705 177773      MOV     #-5,R5
936 003570 012737 003576 001110  MOV     #2$,$LPERR
937 003576 010110 2$:  MOV     R1,R0         ;WRITE THE PATTERN IN THE REGISTER
938 003600 011002      MOV     R0,R2
939 003602 020102      CMP     R1,R2         ;WAS IT WRITTEN CORRECTLY?
940 003604 001407                BEQ     3$           ;YES, BRANCH
941 003606 010137 001162      MOV     R1,$REGO     ;GET EXPECTED WORD
    
```



```

942 003612 010237 001164      MOV      R2,$REG1      ;GET WORD THAT WAS RECD
943 003616 104011              ERROR      11          ;DID NOT READ BACK THE PATTERN THAT
944                                ;WAS WRITTEN INTO THE REGISTER
945 003620 005205              INC      R5
946 003622 001402              BEQ      TST14        ;:EXIT
947 003624 005201      3$:      INC      R1          ;INCREMENT COUNT PATTERN
948 003626 001363              BNE      2$          ;LUP BAK & WRITE NXT PATTERN IF NOT
949                                ;DONE WITH ALL
950

```

```

;*****
;:TEST 14      CHECK THAT RKBA CAN BE SET
;:THIS TEST FLOATS A '1' THROUGH RKBA BITS 0-15 AND CHECKS THAT
;:IT CAN BE READ BACK CORRECTLY. R0 CONTAINS THE WORD THAT WAS
;:WRITTEN.
;*****

```

```

957 003630 000004      TST14:  SCOPE
958 003632 012700 000001      MOV      #1,R0          ;INITIALIZE R0 FOR BIT TO BE
959                                ;WRITTEN IN RKBA
960 003636 013701 001254      MOV      RKBA,R1
961 003642 012737 003650 001110  MOV      #1$,$LPERR     ;SET UP RETURN ADRES FOR
962                                ;LUPING ON EROR (SW 12)
963 003650 010011      1$:      MOV      R0,$R1        ;WRITE THAT BIT IN RKBA
964 003652 011102              MOV      $R1,R2
965 003654 020002              CMP      R0,R2          ;WAS IT WRITTEN CORRECTLY?
966 003656 001405              BEQ      2$            ;YES, BRANCH
967 003660 010037 001162      MOV      R0,$REG0      ;GET EXPCTD RKBA (BIT WRITTEN)
968 003664 010237 001164      MOV      R2,$REG1      ;GET RKBA (BIT READ BACK)
969 003670 104013              ERROR      13          ;DID NOT READ BACK TME BIT THAT
970                                ;WAS WRITTEN IN RKBA
971 003672 006300      2$:      ASL      R0            ;SHIFT R0, TO WRITE NEXT BIT
972 003674 001365              BNE      1$            ;IF ALL BITS ARE NOT CHKD, LOOP
973                                ;BACK & WRITE NXT BIT
974

```

```

;*****
;:TEST 15      CHECK RKBA WITH A COUNT PATTERN
;:THIS TEST CHECKS THAT RKBA CAN BE CLEARED, THEN IT RUNS A COUNT
;:PATTERN FROM 0 TO 177777. R1 CONTAINS THE COUNT PATTERN TO BE WRITTEN.
;*****

```

```

979
980 003676 000004      TST15:  SCOPE
981 003700 012737 000010 001206  MOV      #10,$TIMES    ;;DO 10 ITERATIONS
982 003706 013700 001254      MOV      RKBA,R0
983 003712 012710 177777      MOV      #177777,$R0   ;SET ALL BITS IN RKBA
984 003716 005010              CLR      $R0           ;CLEAR RKBA
985 003720 005710              TST      $R0           ;WAS IT CLEARED?
986 003722 001405              BEQ      1$            ;YES, BRANCH
987 003724 010037 001162      MOV      R0,$REG0      ;GET ADRES OF RKBA
988 003730 011037 001164      MOV      $R0,$REG1     ;NO, GET RKBA
989 003734 104010              ERROR      10          ;RKBA COULD NOT BE CLEARED
990
991 003736 005001      1$:      CLR      R1            ;INITIALIZE COUNT PATTERN
992 003740 012705 177773      MOV      #-5,R5
993 003744 012737 003752 001110  MOV      #2$,$LPERR
994 003752 010110      2$:      MOV      R1,$R0        ;WRITE THE PATTERN IN THE
995                                ;REGISTER
996 003754 011002              MOV      $R0,R2
997 003756 020102              CMP      R1,R2          ;WAS IT WRITTEN CORRECTLY?

```

```

998 003760 001407 BEQ 3$ ;YES, BRANCH
999 003762 010137 001162 MOV R1,$REG0 ;GET EXPECTED WORD
1000 003766 011037 001164 MOV JRO,$REG1 ;GET WORD THAT WAS RECVD
1001 003772 104013 ERROR 13 ;DID NOT READ BACK THE PATTERN THAT
1002 ; WAS WRITTEN INTO THE REGISTER
1003 003774 005205 INC R5
1004 003776 001402 BEQ TST16 ;EXIT
1005 004000 005201 3$: INC R1 ;INCREMENT COUNT PATTERN
1006 004002 001363 BNE 2$ ;LUP BAK & WRITE NXT PATTERN IF NOT
1007 ; DONE WITH ALL
1008
1009 ;*****
1010 ;*TEST 16 CHECK THAT RKDA CAN BE SET
1011 ;*THIS TEST FLOATS A '1' THROUGH RKDA AND CHECKS THAT THE WORD WHICH WAS
1012 ;*WRITTEN IS READ BACK CORRECTLY
1013 ;*****
1014 004004 000004 TST16: SCOPE
1015 004006 005077 175236 CLR JRKCS ;CLEAR RKCS
1016 004012 012700 000001 MOV #1,R0 ;INITIALIZE R0 FOR BIT TO BE WRITTEN IN RKDA
1017 004016 013701 001256 MOV RKDA,R1
1018 004022 012737 004030 001110 MOV #1$,$LPERR ;SET UP RETURN ADRES
1019 ; FOR LUPING ON EROR
1020 004030 010011 1$: MOV R0,JR1 ;WRITE THAT BIT IN RKDA
1021 004032 011102 MOV JR1,R2
1022 004034 020002 CMP R0,R2 ;WAS IT WRITTEN CORRECTLY
1023 004036 001405 BEQ 2$ ;YES, BRANCH
1024 004040 010037 001162 MOV R0,$REG0 ;NO, GET EXPCTD RKDA (BIT WRITTEN)
1025 004044 010237 001164 MOV R2,$REG1 ;GET RKDA (BIT READ BACK)
1026 004050 104015 ERROR 1$ ;DID NOT READ BACK THE BIT THAT
1027 ; WAS WRITTEN IN RKDA
1028 004052 006300 2$: ASL R0 ;SHIFT R0, TOWRITE NXT BIT
1029 004054 001365 BNE 1$ ;IF ALL BITS ARE NOT CHKD, LOOP
1030 ; BACK & WRITE NXT BIT
1031
1032 ;*****
1033 ;*TEST 17 CHECK RKDA WITH A COUNT PATTERN
1034 ;*THIS TEST CHECKS THAT RKDA CAN BE CLEARED, THEN A COUNT PATTERN
1035 ;*FROM 0 TO 177777 IS WRITTEN AND CHECKED. R1 CONTAINS THE COUNT
1036 ;*PATTERN TO BE WRITTEN
1037 ;*****
1038 004056 000004 TST17: SCOPE
1039 004060 012737 000010 001206 MOV #10,$TIMES ;;DO 10 ITERATIONS
1040 004066 013700 001256 MOV RKDA,R0
1041 004072 012710 177777 MOV #177777,JRO ;SET ALL BITS IN RKDA
1042 004076 005010 CLR JRO ;CLEAR RKDA
1043 004100 005710 TST JRO ;WAS IT CLEARED?
1044 004102 001405 BEQ 1$ ;YES, BRANCH
1045 004104 010037 001162 MOV R0,$REG0 ;GET ADRES OF RKDA
1046 004110 011037 001164 MOV JRO,$REG1 ;GO, GET RKDA
1047 004114 104010 ERROR 10 ;RKDA COULD NOT BE CLEARED
1048
1049 004116 005001 1$: CLR R1 ;INITIALIZE COUNT PATTERN
1050 004120 012705 177773 MOV #-5,R5
1051 004124 012737 004132 001110 MOV #2$,$LPERR
1052 004132 010110 2$: MOV R1,JRO ;WRITE THE PATTERN IN THE REGISTER
1053 004134 011002 MOV JRO,R2

```

```

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

```

```

1110 004304 104007          ERROR 7          ;CNTRL RDY DID NOT SET
1111                          ;AFTER CNTRL RESET
1112 004306 022710 000200 2$:  CMP      #200, R0      ;DID CNTRL RESET CLEAR RKCS
1113 004312 001405          BEQ      3$          ;YES, BRANCH
1114 004314 010037 001162  MOV     R0, $REG0    ;GET ADRES OF RKCS
1115 004320 011037 001164  MOV     R0, $REG1    ;GET CONTENTS OF RKCS
1116 004324 104014          ERROR 14          ;CONTROL RESET DID NOT CLEAR RKCS
1117 004326 013702 001252 3$:  MOV     RKWC, R2
1118 004332 010204          MOV     R2, R4
1119 004334 012701 177777  MOV     #177777, R1
1120 004340 010122          MOV     R1, (R2)+
1121 004342 010122          MOV     R1, (R2)+
1122 004344 010122          MOV     R1, (R2)+
1123 004346 104412          CNT.RESET
1124                          ;SET ALL BITS IN RKWC
1125                          ;RKBA
1126                          ;RKDA
1127                          ;GO, DO CONTROL RESET
1128                          ;THIS IS A CALL FOR THE 'CNTRL-
1129                          ;RESET' ROUTINE. A CONTROL RESET IS
1130                          ;DONE & AFTER A CERTAIN TIME IF
1131                          ;'CNTRL RDY' DOES NOT SET AN ERROR IS
1132                          ;REPORTED. NOTE THAT THE PC IN ERROR
1133                          ;IS THE PC WHERE CNT.RESET IS
1134                          ;LOCATED. THIS IS A VERY BASIC ERROR &
1135                          ;IF IT OCCURS GO BACK TO TEST 10.
1136 004350 012703 177775 4$:  MOV     #-3, R3
1137 004354 005714          TST     (R4)
1138 004356 001405          BEQ     5$          ;WAS THE REGISTER CLEARED?
1139 004360 010437 001162  MOV     R4, $REG0    ;YES, BRANCH
1140 004364 011437 001164  MOV     R4, $REG1    ;GET ADRES OF REGISTER IN ERROR
1141 004370 104014          ERROR 14          ;GET CONTENTS OF THAT REGISTER
1142                          ;CONTROL RESET DID NOT CLEAR THE
1143                          ;REGISTER WHOOSE ADRES IS IN R4
1144 004372 005724 5$:  TST     (R4)+
1145 004374 005203          INC     R3
1146 004376 001366          BNE     4$          ;INCREMENT POINTER TO NXT REGISTER
1147                          ;CHKD ALL REGS?
1148                          ;IF NOT, LUP BAK & CHK THE NXT REG
1149
1150 *****
1151 ;*TEST 22 CHECK THAT EACH RK11 REGISTER IS UNIQUELY ADRESSED
1152 ;*THIS TEST CHECKS THAT EACH RK11 REGISTER CAN BE UNIQUELY ADRESSED
1153 ;*1) RKCS RKWC, RKBA, RKDA ARE FIRST SET TO 177777 (17576 FOR RKCS)
1154 ;*2) REGISTER WHOOSE ADDRESS IS IN R0 IS CLEARED
1155 ;*3) EVERY OTHER REGISTER IS CHECKED FOR ERRONEOUS CLEARING BECAUSE OF
1156 ;*ADDRESSING ERROR. IF SO THE MULTIPLE ADDRESSING ERROR IS REPORTED
1157 ;*4) ADDRESS IN R0 IS CHANGED TO THE NEXT REGISTER & THE PROCESS IS
1158 ;* REPEATED.
1159 *****
1160 ;*ST22: SCOPE
1161 004400 000004          MOV     #340, -(SP)
1162 004402 012746 000340  MOV     #64$, -(SP)
1163 004406 012746 004414  RTI
1164 004412 000002          RTI
1165 004414 012705 177771 64$: MOV     #-7, R5
1166                          ;SET UP COUNT FOR THE # OF
1167                          ;REGISTERS TO BE UNIQELY ADDRSD
1168                          ;INITIALIZE POINTER TO REGIS TO BE SET
1169 004420 013700 001244 1$:  MOV     RKDS, R0
1170 004424 012701 177774  MOV     #-4, R1
1171 004430 013702 001250  MOV     RKCS, R2
1172 004434 012722 017576 2$:  MOV     #17576, (R2)+
1173 004440 012722 177777  MOV     #177777, (R2)+
1174 004444 005201          INC     R1
1175                          ;SET BITS IN RKCS
1176                          ;SET BITS IN RKWC
1177                          ;RKBA

```

K03

MAINDEC-11-DZRKJ-D  
DZRKJD.P11 30-AUG-76

MACY11 27(1006)  
14:48

04-OCT-76 13:08 PAGE 23  
T22

CHECK THAT EACH RK11 REGISTER IS UNIQUELY ADDRESSED

SEQ 0036

```

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

```

```

1222
1223 004570 000004
1224 004572 012702 177776
1225 004576 012700 177775
1226 004602 012746 000340
1227 004606 012746 004614
1228 004612 000002
1229 004614
1230 004614 013701 001250
1231 004620 012721 007576
1232 004624 012721 177777
1233 004630 005200
1234 004632 001374
1235 004634 013701 001250
1236 004640 105011
1237 004642 010104
1238 004644 017703 174400
1239 004650 022702 177776
1240 004654 001011
1241 004656 042703 177600
1242 004662 001417
1243 004664 005037 001162
1244 004670 010337 001164
1245 004674 104022
1246
1247
1248 004676 000411
1249 004700 042703 000377
1250 004704 001406
1251 004706 005037 001162
1252
1253 004712 000303
1254 004714 010337 001164
1255 004720 104023
1256
1257
1258 004722 012700 177774
1259 004726 013705 001250
1260 004732 005725
1261 004734 005200
1262 004736 001417
1263 004740 011503
1264
1265 004742 005103
1266
1267 004744 001772
1268
1269 004746 010137 001162
1270
1271 004752 010537 001164
1272
1273 004756 012737 177777 001166
1274
1275 004764 005103
1276 004766 010337 001170
1277 004772 104024

```

```

*****
TST23: SCOPE
MOV #-2,R2 ;SET COUNT FOR 2BYTES-HI & LO
MOV #-3,R0 ;SET COUNT
MOV #340,-(SP)
MOV #64$,-(SP)
RTI

64$: MOV RKCS,R1 ;INITIALIZE PTR. TO RKCS
MOV #7576,(R1)+ ;SET ALL BITS IN RKCS
1$: MOV #177777,(R1)+ ;SET ALL BITS IN RKWC
INC R0 ; RKBA
BNE 1$ ; RKDA
MOV RKCS,R1 ;INITIALIZE PTR TO RKCS LO BYTE
2$: CLRB @R1 ;CLER RKCS LO OR HI BYTE
MOV R1,R4
MOV @RKCS,R3 ;GET RKCS WORD
CMP #-2,R2 ;R U CHKING HI OR LO BYTE?
BNE 3$ ;BRANCH IF HI BYTE
BIC #177600,R3 ;MASK HI BYTE
BEQ 4$ ;OK IF LO BYTE WAS CLEARED
CLR $REG0 ;GET EXPCTD RKCS.
MOV R3,$REG1 ;GET RKCS RECVD, LO BYTE
ERROR 22 ;ALL RK11 REGISTER'S WERE LOADED WITH
;1'S THEN TRIED TO ADRES & CLR RKCS
;LO BYTE, IT COULD NOT BE CLEARED.

3$: BR 4$
BIC #377,R3 ;MASK LO BYTE
BEQ 4$ ;OK IF HI BYTE WAS CLEARED
CLR $REG0 ;GET EXPCTD RKCS, HI BYTE
;GET WHAT WAS ACTUALLY RECVD

SWAB R3
MOV R3,$REG1
ERROR 23 ;ALL RK11 REGISTER'S WERE LOADED WITH
;1'S THEN TRIED TO ADRES & CLR RKCS
;HI BYTE IT COULD NOT BE CLEARED.
;INITIALIZE COUNT FOR REST OF REGISTERS
5$: TST (R5)+ ;INITIALIZE POINTER TO RKCS
INC R0 ;INCREMENT PTR TO NXT REGIS
BEQ 6$ ;ALL REGS DONE?
MOV @R5,R3 ;IF YES, GO & ADRES TO CLEAR RKCS HI BYTE
;IF NOT, GET CONTENTS OF THE REGIS
;BEING CHKD
COM R3 ;COMPLEMENT THE CONTENTS, SHOULD BE
;0 FOR IT WAS
BEQ 5$ ;PREVIOUSLY SET TO ALL 1'S IF IT'S
;0 BRANCH, IF NOT REPORT ERROR
MOV R1,$REG0 ;GET RKCS-BYTE-ADRES WHICH WAS TRIED
;TO ADRES
MOV R5,$REG1 ;GET ADRES OF REGIS WHICH GOT ADRESED
;INSTEAD
MOV #177777,$REG2 ;GET EXPCTD CONTENTS OF REGISTER
;THAT GO ADRESED
COM R3 ;GET CONTENTS RECVD FROM THAT REGIS
MOV R3,$REG3
ERROR 24 ;ALL RK11 REGISTERS WERE LOADED

```

```

1278                                     ;WITH 1'S. RKCS
1279                                     ;BYTE (ADRES UNDER 'BYTE' IN ER
1280                                     ;MSGE) WAS ADRESED
1281                                     ;USING 'CLRB', BUT REGISTER (ADRES
1282                                     ;UNDER 'REGIS' IN
1283                                     ;ER MSG) GOT CHANGED AS A RESULT.
1284 004774 000756                       BR      5$
1285 004776 005201                       6$: INC   R1
1286 005000 005202                       INC   R2
1287                                     ;POSITION PTR TO RKCS HI BYTE
1288 005002 001316                       BNE   2$
1289                                     ;CHK IF BOTH HI & LO BYTES (RKCS)
1290                                     ;WERE CLEARED
1291                                     ;IF NOT BRANCH BACK
1292                                     ;*****
1293                                     ;*TEST 24 CHECK THAT HI & LO BYTES OF RKWC,BA,DA CAN BE ADDRESSED
1294                                     ;*THIS TEST CHECKS THAT BYTE OPERATIONS ON RKWC, RKBA & RKDA CAN BE DONE
1295                                     ;*CORRECTLY. FIRST RKWC, RKCS, RKBA, RKDA ARE SET TO 177777.
1296                                     ;*1) REGISTER BYTE POINTED TO BY R2 IS CLEARED USING 'CLRB'
1297                                     ;*2) IT IS CHECKED THAT ONLY THAT BYTE AND NO OTHER BYTES GET CLEARED
1298                                     ;*3) POINTER R2 IS INCREMENTED TO THE NEXT REGISTER-BYTE & THE PROCESS
1299                                     ;*IS REPEATED. LO BYTE IS DONE FIRST, THEN HI-BYTE IS DONE.
1300 005004 000004 000340                 TST24: SCOPE
1301 005006 012746 005020                 MOV    #340,-(SP)
1302 005012 012746 005020                 MOV    #64$,-(SP)
1303 005016 000002                       RTI
1304 005020 012704 177772                 64$: MOV    #-6,R4
1305 005024 013702 001252                 MOV    RKWC,R2
1306 005030 012700 177775                 1$: MOV    #-3,R0
1307 005034 013701 001250                 MOV    RKCS,R1
1308 005040 012721 007576                 MOV    #7576,(R1)+
1309 005044 012721 177777                 2$: MOV    #177777,(R1)+
1310 005050 005200                       INC    R0
1311 005052 001374                       BNE   2$
1312                                     ;SET RKCS BITS
1313 005054 105012                       CLR B  2R2
1314                                     ;ADDRESS & CLEAR REGIS BYTE UNDER TEST
1315 005056 010200                       MOV    R2,R0
1316 005060 042700 000001                 BIC    #1,R0
1317                                     ;CONVERT THE BYTE ADRES INTO WORD
1318 005064 011001                       MOV    2R0,R1
1319 005066 032702 000001                 BIT    #1,R2
1320 005072 001003                       BNE   3$
1321 005074 042701 177400                 BIC    #177400,R1
1322 005100 000402                       BR     4$
1323 005102 042701 000377                 3$: BIC    #377,R1
1324 005106 001411                       4$: BEQ    5$
1325 005110 010237 001162                 MOV    R2,$REGO
1326                                     ;MASK LO BYTE
1327 005114 032702 000001                 BIT    #1,R2
1328 005120 001401                       BEQ   11$
1329 005122 000301                       SWAB  R1
1330 005124 010137 001164                 11$: MOV    R1,$REG1
1331 005130 104025                       ERROR  2$
1332                                     ;GET CONTENTS OF REG-BYTE
1333                                     ;TRIED TO ADRES & CLR A REGISTER BYTE
1334                                     ;(ADRES UNDER 'REG-BYTE' IN ER MSGE), COULD
1335                                     ;NOT CLEAR IT.

```

```

1334 005132 017701 174112      5$:  MOV    3RKCS,R1      ;
1335 005136 022701 007776      CMP    #7776,R1      ; WAS RKCS ERRONEOUSLY CLRD?
1336 005142 001410                BEQ    6$              ; NO BRANCH
1337 005144 010237 001162      MOV    R2,$REGO      ; GET ADRES OF REG-BYTE THAT WAS
1338                                ; TRIED TO B ADRESED & CLEARED.
1339 005150 012737 007776 001164  MOV    #7776,$REG1   ; GET EXPCTD RKCS
1340 005156 010137 001166      MOV    R1,$REG2      ; GET RKCS RECVD
1341 005162 104016                ERROR   1$           ; ALL RK11 REGISTRS WERE LOADED WITH 1'S
1342                                ; TRIED TO ADRES & CLR 'REGIS-BYTE' (IN ER MSGE)
1343                                ; RKCS GOT CHANGED AS A RESULT. '(RKCS)EXP'
1344                                ; CONTAINS EXPCTD RKCS. 'RKCS (RECVD)' CONTAINS
1345                                ; RKCS RECVD.
1346 005164 012700 177772      6$:  MOV    #-6,R0      ; SET COUNT FOR BYTES
1347 005170 013701 001252      MOV    RKWC,R1        ; INITIALIZE PTR TO RKWC
1348 005174 020102                7$:  CMP    R1,R2        ; WAS THE CLEARED BYTE (PTD TO BY R2)
1349                                ; SAME AS BYTE TO BE CHKD (PTD TO BY R1)?
1350                                ; IF YES, DO NOT CHK THIS BYTE
1351 005176 001433                BEQ    10$            ;
1352 005200 010105                MOV    R1,R5          ;
1353 005202 042705 000001      BIC    #1,R5          ; STRIP WORD ADDRESS FROM BYTE ADDR
1354 005206 011503                MOV    3R5,R3         ;
1355 005210 032701 000001      BIT    #1,R1          ; IS THE BYTE TO B CHKD LO OR HI BYTE?
1356 005214 001003                BNE    8$              ; HI BYTE-BRANCH
1357 005216 042703 177400      BIC    #177400,R3    ; MASK HI BYTE
1358 005222 000402                BR     9$              ;
1359 005224 042703 000377      8$:  BIC    #377,R3      ; MASK LO BYTE
1360 005230 001016                9$:  BNE    10$          ;
1361 005232 010237 001162      MOV    R2,$REGO      ; GET ADRES OF REG-BYTE THAT WAS
1362                                ; TRIED TO B ADRESED & CLEARED
1363 005236 010137 001164      MOV    R1,$REG1      ; GET ADRS OF REG-BYTE THAT GOT
1364                                ; ADRESED INSTEAD
1365 005242 012737 000377 001166  MOV    #377,$REG2    ; GET EXPCTD CONTENTS OF REG-BYTE
1366                                ; THAT GOT ADRESED
1367 005250 032701 000001      BIT    #1,R1          ;
1368 005254 001401                BEQ    12$            ;
1369 005256 000303                SWAB   R3              ;
1370 005260 010337 001170      12$: MOV    R3,$REG3     ; GET CONTENTS RECVD FOR THAT REG-BYTE
1371 005264 104021                ERROR   21           ; ALL RK11 REGISTRS WERE LOADED WITH 1'S.
1372                                ; TRIED TO ADRES & CLR 'REG-BYT1' (IN ER MSGE)
1373                                ; 'REG-BYT2' GOT CHANGD AS A RESULT.
1374                                ; 'BYT2-EXPCT' (IN ER MSGE) IS THE EXPCTD CONTENTS
1375                                ; OF REG-BYT2. 'BYT2-RECVD' IS THE CONTENTS RECVD.
1376 005266 005201                10$: INC    R1          ; INCREMENT PTR TO NXT REG BYTE
1377 005270 005200                INC    R0              ; ALL BYTES CHKD?
1378 005272 001340                BNE    7$              ; NO, LOOP BACK
1379 005274 005202                INC    R2              ; INCREMENT PTR TO NXT REG BYTE TO B CLRD
1380 005276 005204                INC    R4              ; ALL BYTES CLRD?
1381 005300 001253                BNE    1$              ; LOOP BACK
1382
1383
1384
1385      .SBTTL  END OF PASS ROUTINE
1386
1387      ;*****
1388      ;*INCREMENT THE PASS NUMBER ($PASS)
1389      ;*INDICATE END-OF-PROGRAM AFTER 1 PASSES THRU THE PROGRAM

```



```

1390          ;*TYPE "END PASS #XXXXX" (WHERE XXXXX IS A DECIMAL NUMBER)
1391          ;*IF THERES A MONITOR GO TO IT
1392          ;*IF THERE ISN'T JUMP TO START1
1393
1394 005302      SEOP:
1395 005302      SCOPE
1396 005304      CLR          $TSTNM          ;; ZERO THE TEST NUMBER
1397 005310      CLR          $TIMES          ;; ZERO THE NUMBER OF ITERATIONS
1398 005314      INC          $PASS          ;; INCREMENT THE PASS NUMBER
1399 005320      BIC          #100000,$PASS  ;; DON'T ALLOW A NEG. NUMBER
1400 005326      DEC          (PC)+          ;; LOOP?
1401 005330      SEOPCT: .WORD 1
1402 005332      BGT          $DOAGN          ;; YES
1403 005334      MOV          (PC)+,2(PC)+  ;; RESTORE COUNTER
1404 005336      SENDCT: .WORD 1
1405 005340      SEOPCT
1406 005342      TYPE          $SENDMG          ;; TYPE "END PASS #"
1407 005346      MOV          $PASS,-(SP)    ;; SAVE $PASS FOR TYPEOUT
1408 005352      TYPDS          ;; GO TYPE--DECIMAL ASCII WITH SIGN
1409 005354      TYPE          $ENULL          ;; TYPE A NULL CHARACTER
1410 005360      SGET42: MOV     2#42,R0     ;; GET MONITOR ADDRESS
1411 005364      BEQ          $DOAGN          ;; BRANCH IF NO MONITOR
1412 005366      RESET          ;; CLEAR THE WORLD
1413 005370      SENDAD: JSR     PC,(R0)     ;; GO TO MONITOR
1414 005372      NOP          ;; SAVE ROOM
1415 005374      NOP          ;; FOR
1416 005376      NOP          ;; ACT11
1417 005400      SDOAGN:
1418 005400      JMP          2(PC)+          ;; RETURN
1419 005402      SRTNAD: .WORD  START1
1420 005404      SNULL:  .BYTE  -1,-1,0     ;; NULL CHARACTER STRING
1421 005407      SENDMG: .ASCIZ  <15><12>/END PASS #/
1422 005414
1423 005422
1424
1425
1426
1427
1428
1429
1430
1431          .SBTTL  GT3RG:  ROUTINE FOR GETTING RKCS, RKER, RKDS
1432
1433          ;GT3RG
1434          ;SUBROUTINE FOR TRANSFERRING THE CONTENTS OF RKCS, RKER, RKDS
1435          ;TO $REG0, $REG1, $REG2 RESPECTIVELY BEFORE TYPING OUT AN
1436          ;ERROR MESSAGE.
1437          ;CALL:  JSR      PC,GT3RG
1438
1439 005424      GT3RG:  MOV     2RKCS,$REG0   ;GET RKCS
1440 005432      MOV     2RKER,$REG1       ;GET RKER
1441 005440      MOV     2RKDS,$REG2       ;GET RKDS
1442 005446      RTS      PC               ;EXIT FROM THIS SUBROUTINE
1443
1444
1445

```

```

1446
1447          .SBTTL  GT4RG:  ROUTINE FOR GETTING RKCS, RKER, RKDS, RKDA
1448
1449          :GT4RG
1450          :SUBROUTINE FOR TRANSFERRING CONTENTS OF RKCS, RKER, RKDS
1451          :RKDA TO $REG0, $REG1, $REG2, $REG3 RESPECTIVELY BEFORE
1452          :TYPING OUT AN ERROR MESSAGE.
1453          :CALL:  JSR      PC,GT4RG
1454          :GT4RG:  JSR      PC,GT3RG
1455          :          MOV      @RKDA,$REG3
1456          :          RTS      PC
1457
1458          .SBTTL  DELAY:  TIME DELAY ROUTINE
1459
1460          :DELAY
1461          :THIS ROUTINE PROVIDES A VARIABLE TIME DELAY. THE CALL FOR THIS
1462          :ROUTINE IS AN ENCODED 'TRAP' INSTRUCTION.
1463          :CALL:      DELAY  ,N      N IS ANY OCTAL NO. FROM 1 TO 177777
1464          :THE DELAY PROVIDED IS 7.5N US (CONVERT N TO DECIMAL) FOR 11/20
1465          :1.5N US FOR 11/45
1466          :IF THE USER WANTS TO CHANGE THE DELAY TIME (EXMP: SHORTER DELAY TO
1467          :GET A TIGHTER SCOPE LOOP) THE VARIABLE 'N' FOLLOWING 'DELAY' SHOULD
1468          :BE CHANGED TO SUIT THE INDIVIDUAL NEED.
1469
1470          :
1471          :
1472          :
1473          :
1474          :
1475          :
1476          :
1477          :
1478          :
1479          :
1480          :
1481          :
1482          :
1483          :
1484          :
1485          :
1486          :
1487          :
1488          :
1489          :
1490          :
1491          :
1492          :
1493          :
1494          :
1495          :
1496          :
1497          :
1498          :
1499          :
1500          :
1501          :

```

005450	004737	005424					
005454	017737	173576	001170				
005462	000207						

```

1471 005464 017637 000000 001264 DELA.Y: MOV      @2(SP),TIMER      ;GET 'AMOUNT' (N) FOR WHICH
1472 005472 062716 000002          ADD      @2,(SP)      ;DELAY IS TO BE PROVIDED
1473          :          ;ADJUST STACK POINTER TO SKIP OVER 'N'
1474 005476 005337 001264          IS:   DEC      TIMER      ;COUNT DOWN TO 0
1475 005502 001375          :          BNE      IS
1476 005504 000002          :          RTI          ;RETURN TO MAIN PROGRAM
1477
1478
1479
1480          .SBTTL  CON.RESET:  CONTROL REST ROUTINE
1481
1482          :CON.RESET
1483          :THIS ROUTINE ISSUES A CONTROL RESET AND WAITS FOR
1484          :THE 'CNTRL RDY' FLAG TO SET. WHEN THE FLAG SETS
1485          :AN EXIT IS MADE OUT OF THE ROUTINE. IF 'CNTRL-RDY'
1486          :DOES NOT SET WITHIN A CERTAIN TIME AN ERROR MESSAGE
1487          :
1488          :          CNT RDY DIDN'T SET
1489          :          PC=XXXXXX RKCS=YYYYYY
1490          :IS GIVEN. NOTE THAT XXXXXX IS THE PC WHERE 'CNT.RESET' OR 'CNT.RDY'
1491          :IS CALLED.
1492          :
1493          :CALL:  CNT.RESET
1494
1495
1496
1497          .SBTTL  CNT.RDY:  WAIT FOR CONTROL READY ROUTINE
1498
1499          :CN.RDY
1500          :THIS ROUTINE WAITS FOR THE CONTROL READY BIT TO SET AND WHEN IT
1501

```

```

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

```

```

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

```

;;\*\*\*\*\*

.SBTTL ERROR HANDLER ROUTINE

;\*SW15=1 HALT ON ERROR  
;\*SW13=1 INHIBIT ERROR TYPEOUTS  
;\*SW10=1 BELL ON ERROR

1605  
1606  
1607  
1608  
1609  
1610  
1611  
1612  
1613

```

1614 ;#SW09=1          LOOP ON ERROR
1615 ;#SW12=1          CYCLE ON ERROR TO PREVIOUS 'SCOPE'
1616 ;#GO TO SERRTYP ON ERROR
1617
1618 $ERROR: CKSWR          ;CHECK FOR SOFTWARE SWITCH ENTRY REQUEST
1619 006104 104407          ;SET THE ERROR FLAG
1620 006106 105237 001103 7$: INCB          SERFLG
1621 006112 001775          ;DON'T LET THE FLAG GO TO ZERO
1622 006114 013777 001102 173020 MOV          $TSTNM,$DISPLAY
1623 006122 005237 001112 1$: INC          SERRTL
1624 006126 011637 001116          ;COUNT THE NUMBER OF ERRORS
1625 006132 162737 000002 001116 SUB          (SP),SERRPC
1626 006140 117737 172752 001114 MOV          #2,SERRPC
1627 006146 032777 020000 172764 MOV          @SERRPC,$ITEMB
1628 006154 001004          ;STRIP AND SAVE THE ERROR ITEM CODE
1629 006156 004737 006244          ;SKIP TYPEOUT IF SET
1630 006162 104401 001213          ;SKIP TYPEOUTS
1631 006166 005777 172746          ;GO TO USER ERROR ROUTINE
1632 006172 100002          ;HALT ON ERROR
1633 006174 000000          ;SKIP IF CONTINUE
1634 006176 104407          ;HALT ON ERROR!
1635 006200 032777 010000 172732 3$: BIT          #SW12,@SWR
1636 006206 001402          ;CHECK FOR SOFTWARE SWITCH ENTRY REQUEST
1637 006210 013716 001106          ;SW 12 SET?
1638 006214 032777 001000 172716 BIT          $LPADR,(SP)
1639 006222 001402          ;NO, BRANCH
1640 006224 013716 001110          ;ADJUST RETURN ADRES FOR SW12
1641 006230 005737 001210 4$: TST          $SERRPC
1642 006234 001402          ;LOOP ON ERROR SWITCH SET?
1643 006236 013716 001210          ;BR IF NO
1644 006242 000002          ;FUDGE RETURN FOR LOOPING
1645          ;CHECK FOR AN ESCAPE ADDRESS
1646          ;BR IF NONE
1647          ;FUDGE RETURN ADDRESS FOR ESCAPE
1648          ;RETURN
1649          ;
1650          ;
1651          ;
1652          ;
1653          ;
1654          ;
1655          ;
1656          ;
1657          ;
1658          ;
1659          ;
1660          ;
1661          ;
1662          ;
1663          ;
1664          ;
1665          ;
1666          ;
1667          ;
1668          ;
1669          ;
    
```

.SBTTL ERROR MESSAGE TYPEOUT ROUTINE

```

;*****
;THIS ROUTINE USES THE "ITEM CONTROL BYTE" ($ITEMB) TO DETERMINE WHICH
;ERROR IS TO BE REPORTED. IT THEN OBTAINS, FROM THE "ERROR TABLE" ($ERRTB),
;AND REPORTS THE APPROPRIATE INFORMATION CONCERNING THE ERROR.
    
```

```

1652 006244          SERRTYP:
1653 006244 104401 001213          TYPE          $CRLF
1654 006250 010046          MOV          RO,-(SP)
1655 006252 005000          CLR          RO
1656 006254 153700 001114          BISB          @#$ITEMB,RO
1657 006260 001004          BNE          1$
1658          ;"CARRIAGE RETURN" & "LINE FEED"
1659 006262 013746 001116          MOV          SERRPC,-(SP)
1660          ;SAVE RO
1661          ;PICKUP THE ITEM INDEX
1662          ;
1663          ;
1664          ;
1665          ;
1666          ;
1667          ;
1668          ;
1669          ;
1670          ;
1671          ;
1672          ;
1673          ;
1674          ;
1675          ;
1676          ;
1677          ;
1678          ;
1679          ;
1680          ;
1681          ;
1682          ;
1683          ;
1684          ;
1685          ;
1686          ;
1687          ;
1688          ;
1689          ;
1690          ;
1691          ;
1692          ;
1693          ;
1694          ;
1695          ;
1696          ;
1697          ;
1698          ;
1699          ;
1700          ;
1701          ;
1702          ;
1703          ;
1704          ;
1705          ;
1706          ;
1707          ;
1708          ;
1709          ;
1710          ;
1711          ;
1712          ;
1713          ;
1714          ;
1715          ;
1716          ;
1717          ;
1718          ;
1719          ;
1720          ;
1721          ;
1722          ;
1723          ;
1724          ;
1725          ;
1726          ;
1727          ;
1728          ;
1729          ;
1730          ;
1731          ;
1732          ;
1733          ;
1734          ;
1735          ;
1736          ;
1737          ;
1738          ;
1739          ;
1740          ;
1741          ;
1742          ;
1743          ;
1744          ;
1745          ;
1746          ;
1747          ;
1748          ;
1749          ;
1750          ;
1751          ;
1752          ;
1753          ;
1754          ;
1755          ;
1756          ;
1757          ;
1758          ;
1759          ;
1760          ;
1761          ;
1762          ;
1763          ;
1764          ;
1765          ;
1766          ;
1767          ;
1768          ;
1769          ;
1770          ;
1771          ;
1772          ;
1773          ;
1774          ;
1775          ;
1776          ;
1777          ;
1778          ;
1779          ;
1780          ;
1781          ;
1782          ;
1783          ;
1784          ;
1785          ;
1786          ;
1787          ;
1788          ;
1789          ;
1790          ;
1791          ;
1792          ;
1793          ;
1794          ;
1795          ;
1796          ;
1797          ;
1798          ;
1799          ;
1800          ;
1801          ;
1802          ;
1803          ;
1804          ;
1805          ;
1806          ;
1807          ;
1808          ;
1809          ;
1810          ;
1811          ;
1812          ;
1813          ;
1814          ;
1815          ;
1816          ;
1817          ;
1818          ;
1819          ;
1820          ;
1821          ;
1822          ;
1823          ;
1824          ;
1825          ;
1826          ;
1827          ;
1828          ;
1829          ;
1830          ;
1831          ;
1832          ;
1833          ;
1834          ;
1835          ;
1836          ;
1837          ;
1838          ;
1839          ;
1840          ;
1841          ;
1842          ;
1843          ;
1844          ;
1845          ;
1846          ;
1847          ;
1848          ;
1849          ;
1850          ;
1851          ;
1852          ;
1853          ;
1854          ;
1855          ;
1856          ;
1857          ;
1858          ;
1859          ;
1860          ;
1861          ;
1862          ;
1863          ;
1864          ;
1865          ;
1866          ;
1867          ;
1868          ;
1869          ;
1870          ;
1871          ;
1872          ;
1873          ;
1874          ;
1875          ;
1876          ;
1877          ;
1878          ;
1879          ;
1880          ;
1881          ;
1882          ;
1883          ;
1884          ;
1885          ;
1886          ;
1887          ;
1888          ;
1889          ;
1890          ;
1891          ;
1892          ;
1893          ;
1894          ;
1895          ;
1896          ;
1897          ;
1898          ;
1899          ;
1900          ;
1901          ;
1902          ;
1903          ;
1904          ;
1905          ;
1906          ;
1907          ;
1908          ;
1909          ;
1910          ;
1911          ;
1912          ;
1913          ;
1914          ;
1915          ;
1916          ;
1917          ;
1918          ;
1919          ;
1920          ;
1921          ;
1922          ;
1923          ;
1924          ;
1925          ;
1926          ;
1927          ;
1928          ;
1929          ;
1930          ;
1931          ;
1932          ;
1933          ;
1934          ;
1935          ;
1936          ;
1937          ;
1938          ;
1939          ;
1940          ;
1941          ;
1942          ;
1943          ;
1944          ;
1945          ;
1946          ;
1947          ;
1948          ;
1949          ;
1950          ;
1951          ;
1952          ;
1953          ;
1954          ;
1955          ;
1956          ;
1957          ;
1958          ;
1959          ;
1960          ;
1961          ;
1962          ;
1963          ;
1964          ;
1965          ;
1966          ;
1967          ;
1968          ;
1969          ;
1970          ;
1971          ;
1972          ;
1973          ;
1974          ;
1975          ;
1976          ;
1977          ;
1978          ;
1979          ;
1980          ;
1981          ;
1982          ;
1983          ;
1984          ;
1985          ;
1986          ;
1987          ;
1988          ;
1989          ;
1990          ;
1991          ;
1992          ;
1993          ;
1994          ;
1995          ;
1996          ;
1997          ;
1998          ;
1999          ;
2000          ;
    
```

```

1670 006314 104401
1671 006316 000000
1672 006320 104401 001213
1673 006324 012037 006334
1674 006330 001404
1675 006332 104401
1676 006334 000000
1677 006336 104401 001213
1678 006342 011000
1679 006344 001004
1680 006346 012600
1681 006350 104401 001213
1682 006354 000207
1683 006356
1684 006356 013046
1685 006360 104402
1686 006362 005710
1687 006364 001770
1688 006366 104401 006374
1689 006372 000771
1690 006374 020040 000
1691 006400
1692
1693
1694
1695
1696
1697
1698
1699
1700
1701
1702
1703 006400 022737 000176 001140
1704 006406 001074
1705 006410 105777 172530
1706 006414 100071
1707 006416 117746 172524
1708 006422 042716 177600
1709 006426 022726 000007
1710 006432 001062
1711 006434 123727 001134 000001
1712 006442 001456
1713
1714 006444 104401 007125
1715 006450 104401 007132
1716 006454 013746 000176
1717 006460 104402
1718 006462 104401 007143
1719 006466 005046
1720 006470 005046
1721 006472 105777 172446
1722 006476 100375
1723
1724 006500 117746 172442
1725 006504 042716 177600

```

```

2$: TYPE
WORD 0
TYPE ,SCRLF
3$: MOV (RO)+,4$
BEQ 5$
TYPE
WORD 0
TYPE ,SCRLF
5$: MOV (RO),RO
BNE 7$
6$: MOV (SP)+,RO
TYPE ,SCRLF
RTS PC
7$: MOV 2(RO)+,-(SP)
TYPOC
TST (RO)
BEQ 6$
TYPE ,8$
BR 7$
8$: .ASCIZ / /
.EVEN

```

```

::TYPE THE "ERROR MESSAGE"
::"ERROR MESSAGE" POINTER GOES HERE
::"CARRIAGE RETURN" & "LINE FEED"
::PICKUP "DATA HEADER" POINTER
::SKIP TYPEOUT IF 0
::TYPE THE "DATA HEADER"
::"DATA HEADER" POINTER GOES HERE
::"CARRIAGE RETURN" & "LINE FEED"
::PICKUP "DATA TABLE" POINTER
::GO TYPE THE DATA
::RESTORE RO
::"CARRIAGE RETURN" & "LINE FEED"
::RETURN
::SAVE 2(RO)+ FOR TYPEOUT
::GO TYPE--OCTAL ASCII(ALL DIGITS)
::IS THERE ANOTHER NUMBER?
::BR IF NO
::TYPE TWO(2) SPACES
::LOOP
::TWO(2) SPACES

```

.SBTTL TTY INPUT ROUTINE

```

::*****
.ENABL LSB
::*****
::SOFTWARE SWITCH REGISTER CHANGE ROUTINE.
::ROUTINE IS ENTERED FROM THE TRAP HANDLER, AND WILL
::SERVICE THE TEST FOR CHANGE IN SOFTWARE SWITCH REGISTER TRAP CALL
::WHEN OPERATING IN TTY FLAG MODE.

```

```

$CKSWR: CMP #SWREG,SWR
BNE 15$
TSTB 2$TKS
BPL 15$
MOVB 2$TKB,-(SP)
BIC #1C177,(SP)
CMP #7,(SP)+
BNE 15$
CMPB $AUTOB,#1
BEQ 15$
19$: CLR -(SP)
CLR -(SP)
7$: TSTB 2$TKS
BPL 7$
MOVB 2$TKB,-(SP)
BIC #1C177,(SP)

```

```

::IS THE SOFT-SWR SELECTED?
::BRANCH IF NO
::CHAR THERE?
::IF NO, DON'T WAIT AROUND
::SAVE THE CHAR
::STRIP-OFF THE ASCII
::IS IT A CONTROL G?
::NO, RETURN TO USER
::ARE WE RUNNING IN AUTO-MODE?
::BRANCH IF YES
::ECHO THE CONTROL-G (1G)
::TYPE CURRENT CONTENTS
::SAVE SWREG FOR TYPEOUT
::GO TYPE--OCTAL ASCII(ALL DIGITS)
::PROMPT FOR NEW SWR
::CLEAR COUNTER
::THE NEW SWR
::CHAR THERE?
::IF NOT TRY AGAIN
::PICK UP CHAR
::MAKE IT 7-BIT ASCII

```

```

1726
1727
1728
1729 006510 021627 000025 9$: CMP (SP),#25 ;; IS IT A CONTROL-U?
1730 006514 001005 BNE 10$ ;; BRANCH IF NOT
1731 006516 104401 007120 TYPE ,SCNTLU ;; YES, ECHO CONTROL-U (↑U)
1732 006522 062706 000006 20$: ADD #6,SP ;; IGNORE PREVIOUS INPUT
1733 006526 000757 BR 19$ ;; LET'S TRY IT AGAIN
1734
1735
1736 006530 021627 000015 10$: CMP (SP),#15 ;; IS IT A <CR>?
1737 006534 001022 BNE 16$ ;; BRANCH IF NO
1738 006536 005766 000004 TST 4(SP) ;; YES, IS IT THE FIRST CHAR?
1739 006542 001403 BEQ 11$ ;; BRANCH IF YES
1740 006544 016677 000002 172366 MOV 2(SP),@SWR ;; SAVE NEW SWR
1741 006552 062706 000006 11$: ADD #6,SP ;; CLEAR UP STACK
1742 006556 104401 001213 14$: TYPE $CRLF ;; ECHO <CR> AND <LF>
1743 006562 123727 001135 000001 CMPB $INTAG,#1 ;; RE-ENABLE TTY KBD INTERRUPTS?
1744 006570 001003 BNE 15$ ;; BRANCH IF NOT
1745 006572 012777 000100 172344 MOV #100,@STKS ;; RE-ENABLE TTY KBD INTERRUPTS
1746 006600 000002 15$: RTI ;; RETURN
1747 006602 004737 007324 16$: JSR PC,$TYPEC ;; ECHO CHAR
1748 006606 021627 000060 CMP (SP),#60 ;; CHAR < 0?
1749 006612 002420 BLT 18$ ;; BRANCH IF YES
1750 006614 021627 000067 CMP (SP),#67 ;; CHAR > 7?
1751 006620 003015 BGT 18$ ;; BRANCH IF YES
1752 006622 042726 000060 BIC #60,(SP)+ ;; STRIP-OFF ASCII
1753 006626 005766 000002 TST 2(SP) ;; IS THIS THE FIRST CHAR
1754 006632 001403 BEQ 17$ ;; BRANCH IF YES
1755 006634 006316 ASL (SP) ;; NO, SHIFT PRESENT
1756 006636 006316 ASL (SP) ;; CHAR OVER TO MAKE
1757 006640 006316 ASL (SP) ;; ROOM FOR NEW ONE.
1758 006642 005266 000002 17$: INC 2(SP) ;; KEEP COUNT OF CHAR
1759 006646 056616 177776 BIS -2(SP),(SP) ;; SET IN NEW CHAR
1760 006652 000707 BR 7$ ;; GET THE NEXT ONE
1761 006654 104401 001212 18$: TYPE $QUES ;; TYPE ?<CR><LF>
1762 006660 000720 BR 20$ ;; SIMULATE CONTROL-U
1763 .DSABL LSB

```

```

1764
1765
1766 ;;*****
1767 ;;*THIS ROUTINE WILL INPUT A SINGLE CHARACTER FROM THE TTY
1768 ;;*CALL:
1769 ;;* RDCHR ;; INPUT A SINGLE CHARACTER FROM THE TTY
1770 ;;* RETURN HERE ;; CHARACTER IS ON THE STACK
1771 ;;* ;; WITH PARITY BIT STRIPPED OFF
1772
1773
1774 006662 011646 000004 000002 $RDCHR: MOV (SP),-(SP) ;; PUSH DOWN THE PC
1775 006664 016666 000004 000002 MOV 4(SP),2(SP) ;; SAVE THE PS
1776 006672 105777 172246 1$: TSTB @STKS ;; WAIT FOR
1777 006676 100375 BPL 1$ ;; A CHARACTER
1778 006700 117766 172242 000004 MOVB @STKB,4(SP) ;; READ THE TTY
1779 006706 042766 177600 000004 BIC #1C<177>,4(SP) ;; GET RID OF JUNK IF ANY
1780 006714 026627 000004 000023 CMP 4(SP),#23 ;; IS IT A CONTROL-S?
1781 006722 001013 BNE 3$ ;; BRANCH IF NO

```

```

1782 006724 105777 172214 2$: TSTB 2$TKS ;:WAIT FOR A CHARACTER
1783 006730 100375 ;:BPL 2$ ;:LOOP UNTIL ITS THERE
1784 006732 117746 172210 ;:MOVB 2$TKB,-(SP) ;:GET CHARACTER
1785 006736 042716 177600 ;:BIC #177,(SP) ;:MAKE IT 7-BIT ASCII
1786 006742 022627 000021 ;:CMP (SP)+,#21 ;:IS IT A CONTROL-Q?
1787 006746 001366 ;:BNE 2$ ;:IF NOT DISCARD IT
1788 006750 000750 ;:BR 1$ ;:YES, RESUME
1789 006752 026627 000004 000140 3$: CMP 4(SP),#140 ;:IS IT UPPER CASE?
1790 006760 002407 ;:BLT 4$ ;:BRANCH IF YES
1791 006762 026627 000004 000175 ;:CMP 4(SP),#175 ;:IS IT A SPECIAL CHAR?
1792 006770 003003 ;:BGT 4$ ;:BRANCH IF YES
1793 006772 042766 000040 000004 ;:BIC #40,4(SP) ;:MAKE IT UPPER CASE
1794 007000 000002 4$: RTI ;:GO BACK TO USER
1795 ;:*****
1796 ;:*THIS ROUTINE WILL INPUT A STRING FROM THE TTY
1797 ;:*CALL:
1798 ;:* RDLIN ;:INPUT A STRING FROM THE TTY
1799 ;:* RETURN HERE ;:ADDRESS OF FIRST CHARACTER WILL BE ON THE STACK
1800 ;:* ;:TERMINATOR WILL BE A BYTE OF ALL 0'S
1801
1802 007002 010346 $RDLIN: MOV R3,-(SP) ;:SAVE R3
1803 007004 012703 007110 1$: MOV #STTYIN,R3 ;:GET ADDRESS
1804 007010 022703 007120 2$: CMP #STTYIN+8.,R3 ;:BUFFER FULL?
1805 007014 101405 ;:BLOS 4$ ;:BR IF YES
1806 007016 104410 ;:RDCHR ;:GO READ ONE CHARACTER FROM THE TTY
1807 007020 112613 ;:MOVB (SP)+,(R3) ;:GET CHARACTER
1808 007022 122713 000177 10$: CMPB #177,(R3) ;:IS IT A RUBOUT
1809 007026 001003 ;:BNE 3$ ;:SKIP IF NOT
1810 007030 104401 001212 4$: TYPE ,SQUES ;:TYPE A '?'
1811 007034 000763 ;:BR 1$ ;:CLEAR THE BUFFER AND LOOP
1812 007036 111337 007106 3$: MOVB (R3),9$ ;:ECHO THE CHARACTER
1813 007042 104401 007106 ;:TYPE ,9$
1814 007046 122723 000015 ;:CMPB #15,(R3)+ ;:CHECK FOR RETURN
1815 007052 001356 ;:BNE 2$ ;:LOOP IF NOT RETURN
1816 007054 105063 177777 ;:CLRB -1(R3) ;:CLEAR RETURN (THE 15)
1817 007060 104401 001214 ;:TYPE ,SLF ;:TYPE A LINE FEED
1818 007064 012603 ;:MOV (SP)+,R3 ;:RESTORE R3
1819 007066 011646 ;:MOV (SP),-(SP) ;:ADJUST THE STACK AND PUT ADDRESS OF THE
1820 007070 016666 000004 000002 ;:MOV 4(SP),2(SP) ;:FIRST ASCII CHARACTER ON IT
1821 007076 012766 007110 000004 ;:MOV #STTYIN,4(SP)
1822 007104 000002 ;:RTI ;:RETURN
1823 007106 000 ;:9$: .BYTE 0 ;:STORAGE FOR ASCII CHAR. TO TYPE
1824 007107 000 ;: .BYTE 0 ;:TERMINATOR
1825 007110 000010 ;:$TTYIN: .BLKB 8. ;:RESERVE 8 BYTES FOR TTY INPUT
1826 007120 052536 005015 000 ;:$CNTLU: .ASCIZ /!U/<15><12> ;:CONTROL "U"
1827 007125 136 006507 000012 ;:$CNTLG: .ASCIZ /!G/<15><12> ;:CONTROL "G"
1828 007132 005015 053523 020122 ;:$MSWR: .ASCIZ <15><12>/SWR = /
1829 007140 020075 000
1830 007143 040 047040 053505 ;:$MNEW: .ASCIZ / NEW = /
1831 007150 036440 000040
1832
1833 ;.SBTTL TYPE ROUTINE
1834
1835 ;:*****
1836 ;:*ROUTINE TO TYPE ASCIZ MESSAGE. MESSAGE MUST TERMINATE WITH A 0 BYTE.
1837 ;:*THE ROUTINE WILL INSERT A NUMBER OF NULL CHARACTERS AFTER A LINE FEED.

```



```

1838 ;*NOTE1: $NULL CONTAINS THE CHARACTER TO BE USED AS THE FILLER CHARACTER.
1839 ;*NOTE2: $FILLS CONTAINS THE NUMBER OF FILLER CHARACTERS REQUIRED.
1840 ;*NOTE3: $FILLC CONTAINS THE CHARACTER TO FILL AFTER.
1841 ;*
1842 ;*CALL:
1843 ;*1) USING A TRAP INSTRUCTION
1844 ;* TYPE ,MESADR ;;MESADR IS FIRST ADDRESS OF AN ASCIZ STRING
1845 ;*OR
1846 ;* TYPE
1847 ;* MESADR
1848 ;*
1849 ;*
1850 007154 105737 001157 $TYPE: TSTB $TPFLG ;; IS THERE A TERMINAL?
1851 007160 100002 BPL 1$ ;; BR IF YES
1852 007162 000000 HALT ;; HALT HERE IF NO TERMINAL
1853 007164 000407 BR 3$ ;; LEAVE
1854 007166 010046 1$: MOV RO,-(SP) ;; SAVE RO
1855 007170 017600 000002 MOV 2(SP),RO ;; GET ADDRESS OF ASCIZ STRING
1856 007174 112046 2$: MOVB (RO)+,-(SP) ;; PUSH CHARACTER TO BE TYPED ONTO STACK
1857 007176 001005 BNE 4$ ;; BR IF IT ISN'T THE TERMINATOR
1858 007200 005726 TST (SP)+ ;; IF TERMINATOR POP IT OFF THE STACK
1859 007202 012600 60$: MOV (SP)+,RO ;; RESTORE RO
1860 007204 062716 000002 3$: ADD #2,(SP) ;; ADJUST RETURN PC
1861 007210 000002 RTI ;; RETURN
1862 007212 122716 000011 4$: CMPB #HT,(SP) ;; BRANCH IF <HT>
1863 007216 001430 BEQ 8$
1864 007220 122716 000200 CMPB #CRLF,(SP) ;; BRANCH IF NOT <CRLF>
1865 007224 001006 BNE 5$
1866 007226 005726 TST (SP)+ ;; POP <CR><LF> EQUIV
1867 007230 104401 TYPE ;; TYPE A CR AND LF
1868 007232 001213 $CRLF
1869 007234 105037 007370 CLRB $CHARCNT ;; CLEAR CHARACTER COUNT
1870 007240 000755 BR 2$ ;; GET NEXT CHARACTER
1871 007242 004737 007324 5$: JSR PC,$TYPEC ;; GO TYPE THIS CHARACTER
1872 007246 123726 001156 6$: CMPB $FILLC,(SP)+ ;; IS IT TIME FOR FILLER CHARS.?
1873 007252 001350 BNE 2$ ;; IF NO GO GET NEXT CHAR.
1874 007254 013746 001154 MOV $NULL,-(SP) ;; GET # OF FILLER CHARS. NEEDED
1875 ;; AND THE NULL CHAR.
1876 007260 105366 000001 7$: DECB 1(SP) ;; DOES A NULL NEED TO BE TYPED?
1877 007264 002770 BLT 6$ ;; BR IF NO--GO POP THE NULL OFF OF STACK
1878 007266 004737 007324 JSR PC,$TYPEC ;; GO TYPE A NULL
1879 007272 105337 007370 DECB $CHARCNT ;; DO NOT COUNT AS A COUNT
1880 007276 000770 BR 7$ ;; LOOP
1881
1882 ;HORIZONTAL TAB PROCESSOR
1883
1884 007300 112716 000040 8$: MOVB #' ,(SP) ;; REPLACE TAB WITH SPACE
1885 007304 004737 007324 9$: JSR PC,$TYPEC ;; TYPE A SPACE
1886 007310 132737 000007 007370 BITB #7,$CHARCNT ;; BRANCH IF NOT AT
1887 007316 001372 BNE 9$ ;; TAB STOP
1888 007320 005726 TST (SP)+ ;; POP SPACE OFF STACK
1889 007322 000724 BR 2$ ;; GET NEXT CHARACTER
1890 007324 105777 171620 $TYPEC: TSTB 2$TPS ;; WAIT UNTIL PRINTER IS READY
1891 007330 100375 BPL $TYPEC
1892 007332 116677 000002 171612 MOVB 2(SP),2$TPB ;; LOAD CHAR TO BE TYPED INTO DATA REG.
1893 007340 122766 000015 000002 CMPB #CR,2(SP) ;; IS CHARACTER A CARRIAGE RETURN?

```

```

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

```

1950	007526	003002			BGT	B5	::GO TO EXIT
1951	007530	010502			MOV	R5,R2	::GET THE LSD
1952	007532	000764			BR	65	::GO CHANGE TO ASCII
1953	007534	105726			85:	TSTB (SP)+	::WAS THE LSD THE FIRST NON-ZERO?
1954	007536	100003			BPL	95	::BR IF NO
1955	007540	116663	177777	177776	MOV	-1(SP),-2(R3)	::YES--SET THE SIGN FOR TYPING
1956	007546	105013			95:	CLRB (R3)	::SET THE TERMINATOR
1957	007550	012605			MOV	(SP)+,R5	::POP STACK INTO R5
1958	007552	012603			MOV	(SP)+,R3	::POP STACK INTO R3
1959	007554	012602			MOV	(SP)+,R2	::POP STACK INTO R2
1960	007556	012601			MOV	(SP)+,R1	::POP STACK INTO R1
1961	007560	012600			MOV	(SP)+,R0	::POP STACK INTO R0
1962	007562	104401	007610		TYPE	\$DBLK	::NOW TYPE THE NUMBER
1963	007566	016666	000002	000004	MOV	2(SP),4(SP)	::ADJUST THE STACK
1964	007574	012616			MOV	(SP)+,(SP)	
1965	007576	000002			RTI		::RETURN TO USER
1966	007600	023420			SDBLK:	10000.	
1967	007602	001750				1000.	
1968	007604	000144				100.	
1969	007606	000012				10.	
1970	007610	000004			SDBLK:	.BLKW 4	

.SBTTL BINARY TO OCTAL (ASCII) AND TYPE

::\*\*\*\*\*  
 ::THIS ROUTINE IS USED TO CHANGE A 16-BIT BINARY NUMBER TO A 6-DIGIT  
 ::OCTAL (ASCII) NUMBER AND TYPE IT.  
 ::\$TYPOS---ENTER HERE TO SETUP SUPPRESS ZEROS AND NUMBER OF DIGITS TO TYPE

\*CALL:  
 \* MOV NUM,-(SP) ;:NUMBER TO BE TYPED  
 \* TYPOS ;:CALL FOR TYPEOUT  
 \* .BYTE N ;:N=1 TO 6 FOR NUMBER OF DIGITS TO TYPE  
 \* .BYTE M ;:M=1 OR 0  
 \* ;:1=TYPE LEADING ZEROS  
 \* ;:0=SUPPRESS LEADING ZEROS

\*\$TYPON----ENTER HERE TO TYPE OUT WITH THE SAME PARAMETERS AS THE LAST  
 \*\$TYPOS OR \$TYPOC

\*CALL:  
 \* MOV NUM,-(SP) ;:NUMBER TO BE TYPED  
 \* TYPON ;:CALL FOR TYPEOUT

\*\$TYPOC---ENTER HERE FOR TYPEOUT OF A 16 BIT NUMBER  
 \*CALL:  
 \* MOV NUM,-(SP) ;:NUMBER TO BE TYPED  
 \* TYPOC ;:CALL FOR TYPEOUT

1997	007620	017646	000000		STYPOS:	MOV	2(SP),-(SP)	::PICKUP THE MODE
1998	007624	116637	000001	010043	MOV	1(SP),\$OFILL	::LOAD ZERO FILL SWITCH	
1999	007632	112637	010045		MOV	(SP)+,\$OMODE+1	::NUMBER OF DIGITS TO TYPE	
2000	007636	062716	000002		ADD	#2,(SP)	::ADJUST RETURN ADDRESS	
2001	007642	000406			BR	\$TYPON		
2002	007644	112737	000001	010043	STYPOC:	MOV	#1,\$OFILL	::SET THE ZERO FILL SWITCH
2003	007652	112737	000006	010045	MOV	#6,\$OMODE+1	::SET FOR SIX(6) DIGITS	
2004	007660	112737	000005	010042	STYPON:	MOV	#5,\$OCNT	::SET THE ITERATION COUNT
2005	007666	010346			MOV	R3,-(SP)	::SAVE R3	

```

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

.SBTTL TRAP DECODER

2051
2052  ;;*****
2053  ;;*THIS ROUTINE WILL PICKUP THE LOWER BYTE OF THE "TRAP" INSTRUCTION
2054  ;;*AND USE IT TO INDEX THROUGH THE TRAP TABLE FOR THE STARTING ADDRESS
2055  ;;*OF THE DESIRED ROUTINE. THEN USING THE ADDRESS OBTAINED IT WILL
2056  ;;*GO TO THAT ROUTINE.
2057
2058 010046 010046          STRAP:  MOV      R0,-(SP) ;;SAVE R0
2059 010050 016600 000002  MOV      2(SP),R0     ;;GET TRAP ADDRESS
2060 010054 005740          TST      -(R0)       ;;BACKUP BY 2
2061 010056 111000          MOVB     (R0),R0     ;;GET RIGHT BYTE OF TRAP

```

```

2062 010060 006300          ASL      RD      ;; POSITION FOR INDEXING
2063 010062 016000 010102  MOV     $TRPAD(RD),RD  ;; INDEX TO TABLE
2064 010066 000200          RTS      RD      ;; GO TO ROUTINE

```

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

```

2069 010070 011646          STRAP2: MOV     (SP),-(SP)  ;; MOVE THE PC DOWN
2070 010072 016666 000004 000002  MOV     4(SP),2(SP)    ;; MOVE THE PSW DOWN
2071 010100 000002          RTI      ;; RESTORE THE PSW

```

.SBTTL TRAP TABLE

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

```

; ROUTINE
; -----
2080 010102 010070          $TRPAD: .WORD  $TRAP2
2081 010104 007154          $TYPE  ;; CALL=TYPE      TRAP+1(104401) TTY TYPEOUT ROUTINE
2082 010106 007644          $TYPOC ;; CALL=TYPOC     TRAP+2(104402) TYPE OCTAL NUMBER (WITH LEADING ZEROS)
2083 010110 007620          $TYPOS ;; CALL=TYPOS     TRAP+3(104403) TYPE OCTAL NUMBER (NO LEADING ZEROS)
2084 010112 007660          $TYPON ;; CALL=TYPON      TRAP+4(104404) TYPE OCTAL NUMBER (AS PER LAST CALL)
2085 010114 007374          $TYPDS ;; CALL=TYPDS     TRAP+5(104405) TYPE DECIMAL NUMBER (WITH SIGN)
2086
2087 010116 006450          $GTSWR ;; CALL=GTSWR     TRAP+6(104406) GET SOFT-SWR SETTING
2088
2089 010120 006400          $CKSWR ;; CALL=CKSWR     TRAP+7(104407) TEST FOR CHANGE IN SOFT-SWR
2090 010122 006662          $RDCHR ;; CALL=RDCHR    TRAP+10(104410) TTY TYPEIN CHARACTER ROUTINE
2091 010124 007002          $RDLIN ;; CALL=RDLIN     TRAP+11(104411) TTY TYPEIN STRING ROUTINE
2092
2093 010126 005506          CN.RST ;; CALL=CNTR.RESET TRAP+12(104412) CONTROL RESET ROUTINE
2094
2095 010130 005524          CN.RDY ;; CALL=CNTR.RDY  TRAP+13(104413) WAIT FOR CNTRL RDY TO SET
2096
2097 010132 005464          DELA.Y ;; CALL=DELAY   TRAP+14(104414) TIME DELAY ROUTINE
2098
2099

```

.SBTTL POWER DOWN AND UP ROUTINES

```

;*****
; POWER DOWN ROUTINE
2104 010134 012737 010300 000024 $PWRDN: MOV     #SILLUP,@#PWRVEC ;; SET FOR FAST UP
2105 010142 012737 000340 000026  MOV     #340,@#PWRVEC+2 ;; PRIO:7
2106 010150 010046          MOV     RD,-(SP)      ;; PUSH RD ON STACK
2107 010152 010146          MOV     R1,-(SP)     ;; PUSH R1 ON STACK
2108 010154 010246          MOV     R2,-(SP)     ;; PUSH R2 ON STACK
2109 010156 010346          MOV     R3,-(SP)     ;; PUSH R3 ON STACK
2110 010160 010446          MOV     R4,-(SP)     ;; PUSH R4 ON STACK
2111 010162 010546          MOV     R5,-(SP)     ;; PUSH R5 ON STACK
2112 010164 017746 170750  MOV     @SWR,-(SP)    ;; PUSH @SWR ON STACK
2113 010170 010637 010304  MOV     SP,$$AVR6    ;; SAVE SP
2114 010174 012737 010206 000024  MOV     #PWRUP,@#PWRVEC ;; SET UP VECTOR
2115 010202 000000          HALT
2116 010204 000776          BR      -2          ;; HANG UP
2117

```

```

2118 ::*****
2119 ::POWER UP ROUTINE
2120 010206 012737 010300 000024 $PWRUP: MOV $SILLUP,2#PWRVEC ::SET FOR FAST DOWN
2121 010214 013706 010304 MOV $SAVR6,SP ::GET SP
2122 010220 005037 010304 CLR $SAVR6 ::WAIT LOOP FOR THE TTY
2123 010224 005237 010304 1$: INC $SAVR6 ::WAIT FOR THE INC
2124 010230 001375 BNE 1$ ::OF WORD
2125 010232 012677 170702 MOV (SP)+,2SWR ::POP STACK INTO 2SWR
2126 010236 012605 MOV (SP)+,R5 ::POP STACK INTO R5
2127 010240 012604 MOV (SP)+,R4 ::POP STACK INTO R4
2128 010242 012603 MOV (SP)+,R3 ::POP STACK INTO R3
2129 010244 012602 MOV (SP)+,R2 ::POP STACK INTO R2
2130 010246 012601 MOV (SP)+,R1 ::POP STACK INTO R1
2131 010250 012600 MOV (SP)+,R0 ::POP STACK INTO R0
2132 010252 012737 010134 000024 MOV $SPWRDN,2#PWRVEC ::SET UP THE POWER DOWN VECTOR
2133 010260 012737 000340 000026 MOV #340,2#PWRVEC+2 ::PRIO:7
2134 010266 104401 TYPE ::REPORT THE POWER FAILURE
2135 010270 010306 $PWRMG: .WORD $POWER ::POWER FAIL MESSAGE POINTER
2136 010272 012716 MOV (PC)+,(SP) ::RESTART AT PFSTR
2137 010274 002306 $PWRAD: .WORD PFSTR ::RESTART ADDRESS
2138 010276 000002 RTI
2139 010300 000000 $SILLUP: HALT ::THE POWER UP SEQUENCE WAS STARTED
2140 010302 000776 BR .-2 ::BEFORE THE POWER DOWN WAS COMPLETE
2141 010304 000000 $SAVR6: 0 ::PUT THE SP HERE
2142 010306 005015 047520 042527 $POWER: .ASCIZ <15><12>"POWER"
2143 010314 000122 .EVEN
2144
2145
2146 ;ERROR MESSAGES
2147
2148 .SBTTL ERROR MESSAGES
2149
2150 010316 044524 042515 047440 EM1: .ASCIZ /TIME OUT ON RK11 REGISTER/
2151 010324 052125 047440 020116
2152 010332 045522 030461 051040
2153 010340 043505 051511 042524
2154 010346 000122
2155
2156 010350 042522 044507 052123 EM2: .ASCIZ /REGISTER NOT CLEARED/
2157 010356 051105 047040 052117
2158 010364 041440 042514 051101
2159 010372 042105 000
2160
2161 010375 122 041513 020123 EM3: .ASCIZ /RKCS ERROR/
2162 010402 051105 047522 000122
2163
2164 010410 045522 051503 042440 EM4: .ASCIZ /RKCS ERROR-ON WRITING READ ONLY BITS/
2165 010416 051122 051117 047455
2166 010424 020116 051127 052111
2167 010432 047111 020107 042522
2168 010440 042101 047440 046116
2169 010446 020131 044502 051524
2170 010454 000
2171
2172 010455 102 051525 044440 EM5: .ASCIZ /BUS INIT DID NOT CLEAR RKCS/
2173 010462 044516 020124 044504

```

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

2230	011074	020124	042522	031107	
2231	011102	000			
2232					
2233	011103	104	042111	023516	EM22: .ASCIZ /DIDN'T CLEAR RKCS LOW BYTE/
2234	011110	020124	046103	040505	
2235	011116	020122	045522	051503	
2236	011124	046040	053517	041040	
2237	011132	052131	000105		
2238					
2239	011136	044504	047104	052047	EM23: .ASCIZ /DIDN'T CLEAR RKCS HIGH BYTE/
2240	011144	041440	042514	051101	
2241	011152	051040	041513	020123	
2242	011160	044510	044107	041040	
2243	011166	052131	000105		
2244					
2245	011172	051124	042511	020104	EM24: .ASCIZ /TRIED TO CLEAR RKCS 'BYTE', CHANGED 'REGIS'/
2246	011200	047524	041440	042514	
2247	011206	051101	051040	041513	
2248	011214	020123	041047	052131	
2249	011222	023505	020054	044103	
2250	011230	047101	042507	020104	
2251	011236	051047	043505	051511	
2252	011244	000047			
2253					
2254	011246	040506	046111	042105	EM25: .ASCIZ /FAILED TO CLEAR 'REG-BYTE'/
2255	011254	052040	020117	046103	
2256	011262	040505	020122	051047	
2257	011270	043505	041055	052131	
2258	011276	023505	000		
2259					
2260	011301	122	041513	020123	EM26: .ASCIZ /RKCS ALTERED ON CLEARING 'REG-BYTE'/
2261	011306	046101	042524	042522	
2262	011314	020104	047117	041440	
2263	011322	042514	051101	047111	
2264	011330	020107	051047	043505	
2265	011336	041055	052131	023505	
2266	011344	000			
2267					
2268	011345	124	044522	042105	EM27: .ASCIZ /TRIED TO CLEAR 'REG-BYT1', CHANGED 'REG-BYT2'/
2269	011352	052040	020117	046103	
2270	011360	040505	020122	051047	
2271	011366	043505	041055	052131	
2272	011374	023461	020054	044103	
2273	011402	047101	042507	020104	
2274	011410	051047	043505	041055	
2275	011416	052131	023462	000	
2276					
2277	011423	125	042516	050130	EM43: .ASCIZ /UNEXPECTED RK11 INTERRUPT/
2278	011430	041505	042524	020104	
2279	011436	045522	030461	044440	
2280	011444	052116	051105	052522	
2281	011452	052120	000		
2282					
2283		011456			.EVEN
2284					.SBTTL ERROR DATA POINTERS
2285					





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

.END

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

H05

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

SEQ 0059

\$TIMES	001206	\$TPFLG	001157	\$TRPAD	010102	\$TYPEC	007324	\$XTSTR	005644
\$TKB	001146	\$TPS	001150	\$STSTM	001102	\$TYPEX	007372	\$SGET4=	000000
\$TKS	001144	\$TRAP	010046	\$TTYIN	007110	\$TYPC	007644	\$OFILL	010043
\$TN =	000025	\$TRAP2	010070	\$TYPDS	007374	\$TYPON	007660	.	= 012237
\$TPB	001152	\$TRP =	000015	\$TYPE	007154	\$TYPOS	007620		

. ABS. 012237 000

ERRORS DETECTED: 0  
DEFAULT GLOBALS GENERATED: 0

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

