

# DM11

LOGIC TESTS  
MD-11-DZDMA-C

EP-DZDMA-C-DL  
COPYRIGHT ©72-77  
FICHE 1 OF 1

JAN 1978  
**digital**  
MADE IN USA

REM %

IDENTIFICATION

PRODUCT CODE: MAINDEC-11-DZDMA-C-D  
PRODUCT NAME: DM11 LOGIC TESTS  
DATE RELEASED: NOVEMBER, 1977  
MAINTAINER: DIAGNOSTIC GROUP

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 DOCUMENT.

NO RESPONSIBILITY IS ASSUMED FOR THE USE OR RELIABILITY OF SOFTWARE ON EQUIPMENT THAT IS NOT SUPPLIED BY DIGITAL OR ITS AFFILIATED COMPANIES.

COPYRIGHT (C) 1972, 1977 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL  
DEC

PDP  
DECUS

UNIBUS  
DECTAPE

MASSBUS

1. ABSTRACT

TWO SEPARATE DIAGNOSTIC PROGRAMS ARE PROVIDED FOR TESTING THE DM11 (ASYNCHRONOUS DATA MULTIPLEXER), MAINDEC-11-DZDMA (DM11 LOGIC TESTS), AND MAINDEC-11-DZDMB (DM11 MULTIPLE LINE DATA TESTS). THE LOGIC TESTS INDIVIDUALLY TEST EACH OF THE 16 DM11 LINES AND ALL COMMON LOGIC. THE MULTIPLE LINE DATA TESTS RUN SEVERAL LINES CONCURRENTLY AND ARE USED TO TEST LINE INTERACTION AND DATA TRANSMISSION/RECEPTION RELIABILITY. THIS DOCUMENT DESCRIBES THE LOGIC TESTS.

THE AVAILABLE TESTS ARE:

- PRGO - LOGIC TEST
- PRG1 - TRANSMITTER SCOPE LOOP
- PRG2 - TRANSMIT/RECEIVE SCOPE LOOP

2. REQUIREMENTS

2.1 EQUIPMENT

- A. PDP 11 FAMILY PROCESSOR
- B. DM11
- C. JUMPERS CONNECTING 16 TRANSMITTERS TO THEIR RESPECTIVE RECEIVERS.

2.2 STORAGE

THIS PROGRAM USES ALL OF CORE (SK) EXCEPT THAT AREA RESERVED FOR THE LOADERS.

3. LOADING PROCEDURE

THE ABSOLUTE LOADER IS USED TO LOAD THE PROGRAM.

4. USE PROCEDURE

4.1 STARTING PROCEDURE

BEFORE STARTING MAKE SURE THAT THE TTY IS IN REMOTE MODE, AND THE JUMPERS ARE INSTALLED. THREE STARTING ADDRESSES ARE PROVIDED.

0200 - THIS STARTING ADDRESS REQUESTS DM11 PARAMETERS, AND MUST BE USED TO INITIALLY START THE PROGRAM, AND WHENEVER ANY OF THE PARAMETERS LISTED BELOW IS CHANGED.

A. VECTOR ADDRESS ?

RESPONSE: TYPE IN THE VECTOR ADDRESS OF THE DM11 RECEIVER UNDER TEST. CARRIAGE RETURN SELECTS 0300

B. UNIT #(8)?

RESPONSE: THE DM11 UNIT NUMBER CORRESPONDS TO THE ADDRESS TO WHICH THE CLOCK STATUS REGISTER (CSR) RESPONDS.

CSR ADDRESS	DM11 UNIT #	CSR ADDRESS	DM11 UNIT #
175000	0	175100	10
175010	1	175110	11
175020	2	175120	12
175030	3	175130	13
175040	4	175140	14
175050	5	175150	15
175060	6	175160	16
175070	7	175170	17

CARRIAGE RETURN SELECTS UNIT # 0.

C. WHAT IS THE CHARACTER LENGTH?

RESPONSE: CHARACTER LENGTH REFERS TO THE NUMBER OF DATA BITS PER CHARACTER (5-8). CARRIAGE RETURN SELECTS 8 DATA BITS PER CHARACTER.

D. PRG #

RESPONSE: TYPE PROGRAM NUMBER OF PROGRAM YOU WISH TO RUN. CARRIAGE RETURN SELECTS PROGRAM # 0.

NOTES:

CARRIAGE RETURN TERMINATES ALL RESPONSES.  
ANY UNACCEPTABLE RESPONSE WILL RESULT IN A ? TYPEOUT AND THE PARAMETER WILL AGAIN BE REQUESTED.

0204 - THIS STARTING ADDRESS USES PREVIOUSLY DEFINED DM11 PARAMETERS AND REQUESTS THE PROGRAM NUMBER OF THE PROGRAM YOU WISH TO RUN.

0210 - THIS STARTING ADDRESS STARTS THE PREVIOUSLY SELECTED PROGRAM USING PREVIOUSLY SELECTED PARAMETERS

#### 4.2 SWITCH SETTINGS

THE FOLLOWING SWITCH SETTINGS APPLY TO PROGRAM #0.

SR 0-6	ROUTINE TO BE RUN (IF ENABLED BY SR-9)
SR 8	RING BELL ON ERROR
SR 9	LOOP SELECTED ROUTINE
SR 11	INHIBIT ITERATION (DO EACH ROUTINE ONCE)
SR 13	INHIBIT PRINTOUT
SR 14	SCOPE (LOOP ROUTINE)
SR 15	HALT ON ERROR

THIS PROGRAM HAS BEEN MODIFIED TO RUN ON A PROCESSOR WITH OR WITHOUT A HARDWARE SWITCH REGISTER. WHEN FIRST EXECUTED THE PROGRAM TESTS THE EXISTENCE OF A HARDWARE SWITCH REGISTER. IF NOT FOUND A SOFTWARE SWITCH REGISTER LOCATION (SWREG=LOC. 176 ) IS DEFAULTED TO. IF THIS IS THE CASE, UPON EXECUTION THE CONTENTS OF THE SWREG ARE DUMPED IN OCTAL ON THE CONSOLE TTY AND ANY CHANGES ARE REQUESTED

(IE) SWR=XXXXXX NEW=

POSSIBLE RESPONSES ARE:

1. <CR> IF NO CHANGES ARE TO BE MADE
2. 6 DIGITS 0-7 TO REPRESENT IN OCTAL THE NEW SWITCH REGISTER VALUE ; LAST DIGIT FOLLOWED BY <CR>.
3. U TO ALLOW REENTERING VALUE IF ERROR IS COMMITTED KEYING IN SWREG VALUE.

BUILT INTO THE PROGRAM IS THE ABILITY TO DYNAMICALLY CHANGE THE CONTENTS OF SWREG DURING PROGRAM EXECUTION. BY STRIKING G (CNTRL G) ON CONSOLE TTY THE OPERATOR SETS A REQUEST FLAG TO CHANGE THE CONTENTS OF SWREG, WHICH IS PROCESSED IN KEY AREAS OF THE PROGRAM CODE (IE) ERROR ROUTINES, AFTER HALTS END OF PASS, AND OTHER APPLICABLE AREAS.

5.0 PROGRAM DESCRIPTION

5.1 PRGO - LOGIC TESTS

PRGO CONSISTS OF 152(8) INDEPENDENT ROUTINES WHICH TEST VARIOUS FUNCTIONS OF THE DM11 HARDWARE. ANY OF THESE ROUTINES MAY BE INDIVIDUALLY SELECTED AND RUN (SEE SEC. 4.2 FOR SWITCH SETTING)

5.1.1 ROUTINE DESCRIPTION

ROUTINE TESTS

RT0 TESTS THE ABILITY TO REFERENCE THE FOUR DM11 REGISTERS CONTROL STATUS REGISTER (CSR), BUFFER ACTIVE REGISTER (BAR), BREAK STATUS REGISTER (BKCSR), AND THE BASE REGISTER (BASREG). IF AN ILLEGAL REFERENCE OCCURS WHEN THE CSR IS REFERENCED THE PROGRAM WILL INDICATE AN ERROR, AND AUTOMATICALLY LOOP THE ERROR AS LONG AS THE ERROR CONDITION EXISTS.  
RT0 PC=XXXXXX

RT1-RT10 BIT 'BANGS' THE CSR (BITS 0,1,2,4,5,6,12,13), TESTING THAT EACH BIT IN THE CSR CAN BE INDIVIDUALLY SET AND CLEARED. TWO ERROR TYPES ARE DETECTED IN THESE TESTS, A BIT FAILED TO SET, AND/OR A BIT FAILED TO CLEAR. THE ERROR PRINTOUT SHOWS THE ROUTINE THAT FAILED AND THE PC WHERE THE ERROR WAS DETECTED.

RT11- TESTS THAT RESET AND CLEAR CLEAR ALL R/W BITS IN THE CSR. TWO ERROR TYPES ARE DETECTED IN THIS ROUTINE SHOWING THE CONTENTS OF THE CSR AFTER THE RESET & CLEAR INSTRUCTION. THE PROGRAM AUTOMATICALLY LOOPS IF AN ERROR OCCURS. SHOWN BELOW IS THE ERROR TYPEOUT.  
RT11 PC=XXXXXX ERR S/B: 00000 WAS: XXXXXX

RT12 LOADS A BINARY COUNT PATTERN INTO THE BKCSR AND READS BACK THE RESULTS. IF THE DATA READ BACK IS INCORRECT AN ERROR IS INDICATED. THE SCOPE SWITCH WILL CAUSE THE PROGRAM TO RELOAD THE BINARY NUMBER AND REPEAT THE TEST. THE ERROR TYPEOUT SHOWS CORRECT AND ACTUAL RESULTS.  
THE SECOND PORTION OF THE TEST CLEARS THE PREVIOUSLY LOADED NUMBER IF THE SCOPE SWITCH IS SET THE PROGRAM LOOPS BACK AND REPEATS THE CLEAR INSTRUCTION.

RT13 THIS ROUTINE LOADS RANDOM NUMBERS INTO THE BKCSR. IF A RANDOM NUMBER IS LOADED INCORRECTLY AN ERROR IS INDICATED SHOWING THE CORRECT AND ACTUAL RESULTS.

RT14 THIS ROUTINE TESTS THAT RESET WILL CLEAR ALL BREAK STATUS REGISTER (BKCSR) BITS. IF ALL BITS DO NOT CLEAR WHEN THE RESET IS GIVEN AN ERROR IS INDICATED. THE ERROR TYPEOUT SHOWS THE CORRECT RESULT (ALL 0'S) AND THE ACTUAL RESULT.

RT15-RT16 THESE ROUTINES ARE THE SAME AS RT12 & RT13 EXCEPT THAT THE BASE REGISTER IS TESTED.

RT17 THIS ROUTINE TESTS THAT ALL BAR BITS CAN BE INDIVIDUALLY SET AND CLEARED. THE ROUTINE SHIFTS A '1' THROUGH THE BAR THEREBY SETTING EACH BAR BIT AND THEN THE BAR BIT IS CLEARED. THE ERROR TYPEOUTS SHOW CORRECT AND ACTUAL RESULTS.

RT20 THIS ROUTINE TESTS THAT RESET AND CLEAR CLEAR ALL BAR BITS THE ERROR TYPEOUT SHOWS CORRECT AND ACTUAL RESULTS.

RT21-RT23 THESE ROUTINES TEST THAT THE CSR, BAR, AND BKCSR RESPOND PROPERLY TO BYTE COMMANDS. BOTH BYTES ARE REFERENCED IN THESE ROUTINES USING CLRB INSTRUCTIONS. THE ERROR TYPEOUT SHOWS CORRECT AND ACTUAL RESULTS.

RT24 THIS ROUTINE TESTS THAT THE DM11 CAN INTERRUPT THE PROCESSOR VIA THE OVER RUN BIT (CSR BIT 13). THE ERROR TYPEOUT SHOWS THE ROUTINE NUMBER AND THE PC WHERE THE ERROR WAS DETECTED.

RT25 THIS ROUTINE TESTS THAT THE DM11 INTERRUPTS THE PROCESSOR AT THE PROPER LEVEL.

RT26-RT45 THESE ROUTINES TEST THE BASIC TRANSMITTER FUNCTIONS ON EACH LINE

RT46-RT65 THESE ROUTINES TEST THE BASIC RECEIVER FUNCTIONS ON EACH LINE

RT66 THIS ROUTINE TESTS THAT THE DM11 WILL SET THE NEX BIT (CSR BIT 14). WHEN THE DM11 TRIES TO TRANSMIT FROM NON-EXISTANT MEMORY. ALL LINES ARE INDIVIDUALLY TRANSMITTED ON. THE ERROR TYPEOUT SHOWS THE FAILING LINE. ALSO TESTED IS THAT THE NEX BIT WHEN SET CAUSES AN INTERRUPT.

RT67 THIS ROUTINE TESTS THAT THE NEX BIT (CSR BIT 14) SETS WHEN THE DM11 TRIES TO REFERENCE THE TUMBLE TABLE THAT IS IN NON-EXISTANT MEMORY.

RT70 THIS ROUTINE TESTS THAT WHEN THE GO BIT (CSR BIT 0) IS CLEAR THAT NO DATA IS RECEIVED ON ANY LINE. ALL LINES ARE TRANSMITTED ON AND AFTER THE TRANSMISSION IS COMPLETE THE RECEIVER DONE FLAG IS TESTED. THE ERROR TYPEOUT SHOWS THE LINE ON WHICH DATA WAS RECEIVED.

THE TYPEOUT SHOWN BELOW SHOWS THAT DATA WAS RECEIVED ON LINE 0  
RT70 PC=XXXXXX ERRS/B: 000001 WAS: 000001

RT71 THIS ROUTINE TESTS THAT THE CURRENT ADDRESS IS INCREMENTED PROPERLY BY THE DM11. THE TABLE BELOW SHOWS THE ADDRESS LOADED INTO IN THE CURRENT ADDRESS TABLE BEFORE 2 CHARACTERS ARE TRANSMITTED AND THE RESULTANT ADDRESS AFTER THE CHARACTER IS TRANSMITTED

BEFORE	AFTER	BEFORE	AFTER
000000	000001	000777	001000
000001	000002	001777	002000
000003	000004	003777	004000
000007	000010	007777	010000
000017	000020	017777	020000
000037	000040	037777	040000
000077	000100	077777	100000
000177	000200	177777	000000

000377 000400

THE ERROR TYPEOUT SHOWS CORRECT AND ACTUAL CURRENT ADDRESS.

RT72 THIS ROUTINE TESTS THAT DATA CAN BE TRANSMITTED FROM ALL AVAILABLE CORE AND RECEIVED CORRECTLY. THIS IS DONE BY TRANSMITTING 1 CHARACTER FROM SEVERAL ADDRESSES IN EACH 4K BLOCK OF CORE ON LINE 0. THE ERROR TYPEOUT WILL SHOW TRANSMITTED AND ACTUAL RECEIVED DATA. IF A DATA ERROR RESULTED WHEN TRANSMITTING FROM THE FIRST 4K OF CORE EXAMINE THE CURRENT ADDRESS OF LINE 0 TO DETERMINE WHERE IN THE FIRST 4K OF CORE THE DM11 WAS TRANSMITTING FROM WHEN ERROR OCCURRED. FOR ERRORS IN OTHER 4K BLOCKS THE CORRECT RESULT CORRELATES TO THE ADDRESS WHERE THE ERROR OCCURRED. FOR EXAMPLE

RT72 PC=XXXXXX ERR S/B: 000001 WAS XXXXXX.  
INDICATES THAT THE DM11 FAILED TO TRANSMIT AND RECEIVE CORRECT DATA WHEN TRANSMITTING FROM LOCATION 20000.  
THE TEST IS ABORTED BEFORE TRANSMITTING IF THE CORE LOCATION IS NON-EXISTANT.



RT73 THIS ROUTINE TESTS THAT THE TRANSMITTER CAN TRANSMIT 100 CHARACTERS ON EACH LINE. THE ROUTINE TESTS THAT EXACTLY 100 CHARACTERS HAVE BEEN TRANSMITTED BEFORE READY (CSR BIT15) SETS AND THE BAR BIT CLEARS. THE ERROR TYPEOUT GIVES THE NUMBER OF CHARACTERS RECEIVED AT THE TIME OF AN ERROR, AND THE FAILING LINE NUMBER (X2).

RT74 THIS ROUTINE TESTS THAT THE DM11 WILL STORE DATA SEQUENTIALLY IN THE TUMBLE TABLE AND ALSO THAT THE POINTER RETURNS TO THE TOP OF THE TABLE WHEN 64 CHARACTERS HAVE BEEN RECEIVED.

RT75-114 THESE ROUTINES CHECK THAT A BREAK CAN BE TRANSMITTED AND RECEIVED ON ALL ALINES.

R115-R134 THESE ROUTINES INDIVIDUALLY TRANSMIT, RECEIVE AND CHECK DATA PLUS PARITY ON EACH OF THE 16 DM11 LINES. ONLY DATA AND PARITY ERRORS ARE REPORTED.

RT131 THIS ROUTINE SIMULTANEOUSLY TRANSMITS AND RECEIVES A CHARACTER (ALL 1'S) ON THE 16 DM11 LINES. THE FOLLOWING TESTS ARE PERFORMED:

- A: THERE ARE 16 DATA ENTRIES (1 PER LINE)
- B: THERE ISN'T A 17TH ENTRY
- C: DATA IS CORRECT
- D: ONE ENTRY FOR EACH LINE

RT136 THIS ROUTINE TRANSMITS A BREAK ON EACH LINE. TESTS PERFORMED ARE THE SAME AS IN RT135.

RT137-RT144 THESE ROUTINES TRANSMIT 64 CHARACTERS ON EACH LINE WITH A DELAY BEFORE BEGINNING TRANSMISSION ON THE NEXT SUCCESSIVE LINE. THE DELAY BEFORE TRANSMITTING ON THE NEXT LINE IS HALVED BY SUCCESSIVE TESTS. NO DATA CHECKING IS PERFORMED BY THESE TESTS. TESTED ARE THAT OVER RUN (CSR BIT13) AND NEX (CSR BIT14) ARE NOT SET DURING TRANSMISSION/RECEPTION.

RT145 THIS ROUTINE TESTS PROPER OPERATION OF THE HALF DUPLEX BIT (CSR BIT1)

RT146 THIS ROUTINE TESTS THAT THE DM11 COMES TO AN 'ORDERLY HALT' WHEN THE RESET INSTRUCTION IS GIVEN. 'ORDERLY HALT' IS DEFINED AS CSR, BAR, AND BKCS CLEAR IMMEDIATELY AFTER THE RESET INSTRUCTION AND STAY CLEARED.

- 5.2 PRG1- TRANSMITTER SCOPE LOOP  
PROGRAM 1 ALLOWS THE USER TO SCOPE THE DM11 TRANSMITTER FUNCTIONS WITH THE DM11 CONTINUOUSLY RUNNING UNDER USER SUPPLIED PARAMETERS.
- 5.3 PRG2- TRANSMITTER/RECEIVER SCOPE LOOP  
PROGRAM 2 ALLOWS THE USER TO SCOPE THE DM11 RECEIVER FUNCTIONS WITH THE DM11 CONTINUOUSLY RUNNING UNDER USER SUPPLIED PARAMETERS.

6.0 PROGRAM 1 AND PROGRAM 2 PARAMETERS  
WHEN PROGRAM 1 OR PROGRAM 2 ARE SELECTED ADDITIONAL PARAMETERS WILL  
BE REQUESTED BY EACH PROGRAM AS SHOWN BELOW.

A: TYPE LINES TO BE TESTED  
EXAMPLES:  
TYPE TO SELECT LINE(S)  
1 0  
3 1,0  
10 3  
17 3,2,1,0  
50 5,3  
3101 10,7,6,0  
17770 14,13,12,11,10,7,6,5,4,3  
177777 ALL  
NOTE, LINE NUMBERS ARE GIVEN IN OCTAL.

B: HOW MANY CHARACTERS  
TYPE THE NUMBER OF CHARACTERS YOU WISH TO TRANSMIT. NOTE,  
THE NUMBER OF CHARACTERS MUST BE LESS THAN 200, AND IS TAKEN  
IN OCTAL.

C: PUT CHARACTER IN SR (0-7); DELAY IN SR (8-15)  
SELF-EXPLANATORY. NOTE, THE DELAY REFERS TO A DELAY AFTER  
ALL THE CHARACTERS HAVE BEEN TRANSMITTED AND BEFORE A NEW  
TRANSMISSION PERIOD BEGINS.

7.0 PROGRAM LIMITATIONS  
BECAUSE THE DM11 DIAGNOSTICS ARE INSENSITIVE TO 'REAL' ELAPSED TIME  
THE DIAGNOSTIC DOES NOT 'KNOW' IF THE DM11 IS OPERATING AT THE COR-  
RECT FREQUENCY OR THAT THE STOP CODE SELECTION LOGIC IS CORRECT,  
THESE SHOULD BE CHECKED WITH A SCOPE.

8.0 PROGRAM NOTES  
IF THE POWER FAILS THE PROGRAM TYPES AN ERROR MESSAGE INDICATING THE  
ROUTINE THAT WAS RUNNING (PROG #0 ONLY) AND RESTARTS THE PROGRAM.

\*\*\*\*\* IMPORTANT NOTE \*\*\*\*\*

POWER FAIL TEST

A TEST OF THE POWER FAIL LOGIC SHOULD BE PERFORMED ON EACH UNIT.  
SELECT & RUN ROUTINE 144 (L.A. = 210 SR = 5144 PRESS START). TURN  
THE POWER OFF THEN ON. THE PROGRAM WILL TYPE OUT THE POWER FAIL  
ERROR

R144 PC=0J3622

AND CONTINUE RUNNING ROUTINE 144. LOWER SR 9 AND WAIT FOR END OF  
TEST MESSAGE. 'TEST DZDMA COMPLETE'

NOTE: IF THE POWER IS TURNED OFF DURING A RESET INSTRUCTION THE  
PROGRAM WILL HALT. PRESS CONTINUE AND REPEAT THE TEST

IF THE PROGRAM HANGS THE BUS EXAMINE THE CONTENTS OF RTNNO. THE  
CONTENTS OF RTNNO IS THE ROUTINE NUMBER THAT WAS RUNNING AT THE TIME  
OF THE FAILURE.

%

. TITLE MAINDEC-11-DZDMA-C DM11 LOGIC TESTS  
. NLIST MC,MD,CND  
. LIST ME  
. ENABLE ABS,AMA

;DM11 LOGIC TESTS DIAGNOSTIC (MAINDEC-11-DZDMA)  
;COPYRIGHT 1972,1977 DIGITAL EQUIPMENT CORP., MAYNARD, MASS. 01754  
;PRG0- INPUT-OUTPUT LOGIC TESTS  
;PRG1- TRANSMITTER SCOPE LOOP  
;PRG2- TRANSMIT/RECEIVE SCOPE LOOP  
;STANDARD SR SWITCH OPTIONS (SWITCH SET TO A 1 )  
;SR15- HALT ON ERROR  
;SR14- SCOPE.  
;SR13- INHIBIT PRINTOUT  
;SR12- INHIBIT TRACE (NOT USED)  
;SR11- INHIBIT ITERATION.  
;SR10- LOOP PROGRAM. (NOT USED)  
;SR9- LOOP ROUTINE.  
;SR8- RING BELL ON AN ERROR  
;SR6 THROUGH SR0 - NUMBER OF ROUTINE TO BE LOOPED.

;EQUATE STATEMENTS

177776  
177776  
000004  
000240  
000000  
100000  
100000  
040000  
020000  
010000  
004000  
002000  
001000  
000400  
000200  
000100  
000040  
000020  
000010  
000004  
000002  
000001  
100000  
040000  
020000  
010000  
004000  
002000  
001000  
000400  
000200  
000100

CC=177776  
PSW=177776  
ERRVEC=4  
NOP=240  
OPEN=0  
MANUAL=BIT15  
LBIT17=100000  
LBIT16=40000  
LBIT15=20000  
LBIT14=10000  
LBIT13=4000  
LBIT12=2000  
LBIT11=1000  
LBIT10=400  
LBIT7=200  
LBIT6=100  
LBIT5=40  
LBIT4=20  
LBIT3=10  
LBIT2=4  
LBIT1=2  
LBIT0=1  
BIT15=100000  
BIT14=40000  
BIT13=20000  
BIT12=10000  
BIT11=4000  
BIT10=2000  
BIT9=1000  
BIT8=400  
BIT7=200  
BIT6=100

;ADDRESS OF ERROR TRAP VECTOR

000040  
000020  
000010  
000004  
000002  
000001  
005726  
022626  
000340  
000300  
000240  
000200  
000140  
000100  
000040  
000000

BIT5=40  
BIT4=20  
BIT3=10  
BIT2=4  
BIT1=2  
BIT0=1  
POPSP=5726  
POPSP2=022626  
PRTY7=340  
PRTY6=300  
PRTY5=240  
PRTY4=200  
PRTY3=140  
PRTY2=100  
PRTY1=40  
PRTY0=0

; POP THE STACK. SAME AS TST (6)+  
; POP STACK TWICE. SAME AS CMP (6)+, (6)+  
; PRIORITY LEVEL DEFINITIONS

; LINE NUMBERS

000000  
000002  
000004  
000006  
000010  
000012  
000014  
000016  
000020  
000022  
000024  
000026  
000030  
000032  
000034  
000036  
000000  
000001  
000002  
000003  
000004  
000005  
000006  
000007

LINE0=0  
LINE1=2  
LINE2=4  
LINE3=6  
LINE4=10  
LINE5=12  
LINE6=14  
LINE7=16  
LINE10=20  
LINE11=22  
LINE12=24  
LINE13=26  
LINE14=30  
LINE15=32  
LINE16=34  
LINE17=36  
R0=%0  
R1=%1  
R2=%2  
R3=%3  
R4=%4  
R5=%5  
SP=%6  
PC=%7

; EMT CALLS

104000  
104001  
104002  
104003  
104004  
104005  
104006  
104007  
104010  
104011  
104012  
104013  
104014  
104015

TYPE=EMT+0  
ERROR=EMT+1  
DATCHK=EMT+2  
CHALT=EMT+3  
EHALT=EMT+4  
SRESET=EMT+5  
SCOPE=EMT+6  
SAVREG=EMT+7  
RSTREG=EMT+10  
ERROR1=EMT+11  
INITIALIZE=EMT+12  
SUSWR=EMT+13  
KBDIN=EMT+14  
CNTLIJ=EMT+15

104400	DELAY=TRAP+0	
000007	BELL=007	
177777	RTLAST=-1	
000000	Y=0	
177777	X=-1	
000000	A=0	
000000	. =0	
000000	. +2	; UNASSIGNED TRAP
000002	HALT	
000004	. +2	; SP OVERFLOW, BUS ERROR TRAP
000006	HALT	
000010	. +2	; RESERVED INSTRUCTION TRAP
000012	HALT	
000014	. +2	; TRACE TRAP
000016	HALT	
000020	. +2	; TRAP TO CALL IOX
000022	HALT	
000024	. +2	; POWER FAIL TRAP
000026	HALT	
000030	EMTINT	; EMT TRAP
000032	PRTY7	
000034	DLY	; TRAP TRAP. SIMILAR TO EMT
000036	PRTY7	
000040	. +2	
000042	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000044	. +2	
000046	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000050	. +2	
000052	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000054	. +2	
000056	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000060	. +2	
000062	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000064	. +2	
000066	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000070	. +2	
000072	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000074	. +2	
000076	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000100	. +2	
000102	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000104	. +2	
000106	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000110	. +2	
000112	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000114	. +2	
000116	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000120	. +2	
000122	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000124	. +2	
000126	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000130	. +2	
000132	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000134	. +2	
000136	HALT	; TRAPPED TO PREVIOUS ADDRESS.

MACHER:

000140	000142	.+2	
000142	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000144	000146	.+2	
000146	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000150	000152	.+2	
000152	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000154	000156	.+2	
000156	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000160	000162	.+2	
000162	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000164	000166	.+2	
000166	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000170	000172	.+2	
000172	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000174	000176	.+2	
000176	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000200	000202	.+2	
000202	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000204	000206	.+2	
000206	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000210	000212	.+2	
000212	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000214	000216	.+2	
000216	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000220	000222	.+2	
000222	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000224	000226	.+2	
000226	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000230	000232	.+2	
000232	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000234	000236	.+2	
000236	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000240	000242	.+2	
000242	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000244	000246	.+2	
000246	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000250	000252	.+2	
000252	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000254	000256	.+2	
000256	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000260	000262	.+2	
000262	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000264	000266	.+2	
000266	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000270	000272	.+2	
000272	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000274	000276	.+2	
000276	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000300	000302	.+2	
000302	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000304	000306	.+2	
000306	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000310	000312	.+2	
000312	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000314	000316	.+2	
000316	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.

000320	000322	. +2	
000322	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000324	000326	. +2	
000326	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000330	000332	. +2	
000332	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000334	000336	. +2	
000336	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000340	000342	. +2	
000342	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000344	000346	. +2	
000346	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000350	000352	. +2	
000352	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000354	000356	. +2	
000356	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000360	000362	. +2	
000362	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000364	000366	. +2	
000366	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000370	000372	. +2	
000372	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.
000374	000376	. +2	
000376	000000	HALT	; TRAPPED TO PREVIOUS ADDRESS.

000046	000046		. =46		
	002544		SENDAD		
	000052		. =52		
000052	060000		60000		
	000174		. =174		
000174	000000	DISPREG:	0		
000176	000000	SWREG:	0		
	000200		. =200		
000200	000137	002066	JMP	@#START	; GO TO START OF DIAGNOSTIC.
000204	000137	002134	JMP	@#RSTAT1	; GO GET PROGRAM # & RESTART PROGRAM
					; USING PREVIOUS DM11 PARAMETERS
000210	000137	002222	JMP	@#RSTAT2	; RESTART PREVIOUS PROGRAM USING
					; PREVIOUS DM11 PARAMETERS
	001100		. =1100		
001100	000000	SPBOT:	0		
001102	177570	SWR:	177570		
001104	177570	DISPLAY:	177570		
001106	000000	CAT:	OPEN		; STARTING ADDRESS OF
	001146		. =CAT+32.		; CURRENT ADDRESS TABLE
001146	000000	WCT:	OPEN		; STARTING ADDRESS OF
	001206		. =WCT+32.		; WORD COUNT TABLE
001206	000000	BAT:	OPEN		; STARTING ADDRESS OF
	001246		. =BAT+32.		; BIT ASSEMBLY TABLE
001246	000000	VAC:	OPEN		; 32. SPARE WORDS
001250	175000	CSR:	175000		; ADDRESS OF CLOCK STATUS REGISTER
001252	175002	BAR:	175002		; ADDRESS OF BUFFER ACTIVE REGISTER
001254	175004	BKCSR:	175004		; ADDRESS OF BREAK STATUS REGISTER
001256	175006	BASREG:	175006		; ADDRESS OF BASE REGISTER
001260	000000	CLKINT:	OPEN		; DM11 VECTOR ADDRESS (RECEIVER)
001262	000240	CLKLVL:	PRTY5		; PRIORITY LEVEL
001264	000000	XMTINT:	OPEN		; DM11 VECTOR ADDRESS (TRANSMITTER)
001266	000240	XMTLVL:	PRTY5		; TRANSMITTER PRIORITY LEVEL
001270	000000	TTDAT:	OPEN		; TUMBLE TABLE DATA
001272	000000	LINBIT:	OPEN		; LINE BIT (FOR BAR)
001274	000000	RCVDAT:	OPEN		
001276	000000	XMTDAT:	OPEN		
001300	000000	CARMSK:	OPEN		
	001306		. =VAC+32.		
001306	000000	TUMTAB:	OPEN		; STARTING ADDRESS OF
	001506		. =TUMTAB+128.		; TUMBLE TABLE
001506	000000	KSTART:	OPEN		; CURRENT PROGRAM START ADDRESS.
001510	000000	CURTST:	OPEN		; CONTAINS ADDR OF CURRENT TEST.
001512	000000	RTNNO:	OPEN		; CONTAINS CURRENT TEST #.
001514	000000	NXTST:	OPEN		; CONTAINS ADDR OF NEXT TEST.
001516	000000	ICTR:	OPEN		; CONTAINS CURRENT ITERATION COUNT
001520	000000	SCOPTR:	OPEN		; CONTAINS CURRENT SCOPE POINTER.
001522	000000	PRGLIM:	OPEN		
001524	006604	PRGTAB:	PRGO		; PRGO START ADDRESS
001526	016222		PRG1		; PRG1 START ADDRESS
001530	016270		PRG2		; PRG2 START ADDRESS



001532	006626	RSTART:	PRGOR	; PRGO RESTART ADDRESS
001534	016232		PRG1R	; PRG1 " "
001536	016300		PRG2R	; PRG2 " "
001540	003030	EMTTAB:	TYP	; POINTER TO TIMEOUT ROUTINE
001542	001654		ERR	; POINTER TO ERROR ROUTINE
001544	001634		DTCHK	; POINTER TO DATA COMPARISON ROUTINE
001546	000000		0	
001550	000000		0	
001552	002740		SRSETT	; POINTER TO RESET ROUTINE
001554	002406		ESCOPE	; POINTER TO SCOPE ROUTINE
001556	002640		SAVRG	; POINTER TO SAVE REGISTERS ROUTINE
001560	002700		RSTRG	; POINTER TO RESTORE REGISTERS ROUTINE
001562	001672		ERR1	; POINTER TO ERROR1 ROUTINE
001564	003142		INIT	; POINTER TO INITIALIZE ROUTINE
001566	017074		SUSWRP	
001570	017154		KBDINTT	
001572	017230		CNTLUU	
001574	177560	TKCSR:	177560	
001576	177562	TKDBR:	177562	
001600	177564	TPCSR:	177564	
001602	177566	TPDBR:	177566	
001604	000000	COUNT:	OPEN	
001606	000000	PCADD:	OPEN	
001610	000000	APCADD:	OPEN	
001612	000000	PRVCNT:	OPEN	
001614	175001	.CSR:	175001	
001616	175003	.BAR:	175003	
001620	175005	.BKCSR:	175005	
001622	175007	.BASREG:	175007	
001624	000000	PASS:	OPEN	

```

;ROUTINE TO TYPE OUT INCORRECT ROUTINE SELECTED
001626 104000 INCRTN: TYPE
001630 017444 M1 ;TYPE INCORRECT ROUTINE SELECTED.
001632 000207 RTS %7 ;EXIT.

;DATA COMPARISON ROUTINE.
001634 123737 001274 001276 DTCHK: CMPB RCVDAT,XMTDAT ;COMPARE RECEIVED & TRANSMITTED DATA
001642 001403 BEQ 1$ ;CHARS. BRANCH IF SAME.
001644 004737 002040 JSR 7,CNVDAT ;CONVERT RCVDAT & XMTDAT TO ASCII
001650 104011 ERROR1
001652 000002 1$: RTI ;EXIT.

;ERROR ROUTINE WHENEVER THE PROGRAM DETECTS AN ERROR THE ERROR
;AND ERROR1 EMT INSTRUCTIONS ENTER HERE. ERROR AT ERR:,AND
;ERROR1 AT ERR1:
001654 012737 000402 001754 ERR: MOV #402,ERRB ;MOV BR.+6 TO ERRB
001662 013737 001606 001610 MOV PCADD,APCADD ;GET PC WHERE ERROR OCCURRED
001670 000410 BR ERRA
001672 012737 000240 001754 ERR1: MOV #240,ERRB ;MOVE NOP TO ERRB
001700 013737 001606 001610 MOV PCADD,APCADD ;GET PC WHERE ERROR OCCURRED
001706 004737 002040 JSR 7,CNVDAT ;CONVERT RCVDAT & XMIT DAT TO ASCII
001712 104014 ERRA: KBDIN ;GO CHECK FOR G
001714 032777 020000 177160 BIT #BIT13,@SWR ;ERROR PRINTOUT DESIRED
001722 001017 BNE ERRC ;BRANCH IF NO PRINTOUT
001724 004537 005662 JSR 5,OACNV ;CONVERT
001730 001610 APCADD ;DATA
001732 017650 APC ;TO
001734 000006 6 ;ASCII
001736 004537 005662 JSR 5,OACNV ;FOR
001742 001512 RTNNO ;PRINTOUT
001744 017642 ATNUMB
001746 000003 3
001750 104000 TYPE ;TYPE ERROR
001752 017637 EMO ;MESSAGE
001754 000000 ERRB: OPEN ;NOP IF ERROR1, BR.+6 IF ERROR
001756 104000 TYPE ;TYPE ANOTHER MESSAGE
001760 017401 ERDAT ;IF ERROR 1
001762 032777 000400 177112 ERRC: BIT #BITS,@SWR ;RING BELL ON ERROR?
001770 001411 BEQ ERRD ;BRANCH IF NO BELL ON ERROR
001772 105777 177602 TSTB @TPCSR ;TELEPRINTER
001776 100375 BPL -4 ;READY?
002000 012777 000007 177574 MOV #BELL,@TPDBR ;RING THE BELL
002006 105777 177566 TSTB @TPCSR ;WAIT FOR THE BELL TO RING
002012 100375 BPL -4
002014 023737 000042 000046 ERRD: CMP @#42,@#46 ;ACT11?
002022 001403 BEQ ERRHLT
002024 005777 177052 TST @SWR ;HALT ON ERROR
002030 100001 BPL ERREX ;GO TO EXIT IF NO HALT ON ERROR
002032 000000 ERRHLT: HALT
002034 104014 ERREX: KBDIN ;CHECK FOR G
002036 000002 RTI ;RETURN

```

;SUBROUTINE TO CONVERT RCVDAT AND XMTDAT TO ASCII AND PLACE

```
002040 004537 005662 ; IN MESSAGE.  
002044 001276 CNVDAT: JSR 5, OACNV  
002046 017413 XMTDAT  
002050 000006 AASB  
002052 004537 005662 6  
002056 001274 JSR 5, OACNV  
002060 017427 RCVDAT  
002062 000006 AWAS  
002064 000207 6  
RTS 7 ;EXIT
```

; THE FIRST PART OF THE START ROUTINE CONTAINS A SHORT  
 ; ROUTINE TO CHECK FOR MEMORY MANAGEMENT. ALTHOUGH THIS  
 ; DIAGNOSTIC DOES NOT USE MEMORY MANAGEMENT, ITS PRESENCE  
 ; INDICATES THAT OVER 28K OF MEMORY MAY BE PRESENT IN WHICH  
 ; CASE TESTS RT66 AND RT67 MAY FAIL. IF MEM. MAN. IS  
 ; PRESENT THESE TESTS ARE SKIPPED BY THE PROGRAM.

```

002066 012706 001100      START:  MOV    #SPBOT,%6
002072 104013              SUSWR                ;SEE IF SWITCH-LESS PROCESSOR
002074 012737 002116 000004      MOV    #1$,@#ERRVEC ;SET UP FOR ERROR TRAP
002102 005737 172300              TST    @#172300     ;TEST FOR KT11
002106 012737 012424 012036      MOV    #RT70,@#RT65+2 ;KT11 PRESENT, SET UP TO
                                ;SKIP RT66 AND RT67

002114 000402              BR     +6
002116 012706 001100      1$:   MOV    #SPBOT,%6 ;TRAP OCCURRED, NO KT11 PRESENT,
                                ;RESET STACK
002122 012737 000006 000004      MOV    #ERRVEC+2,@#ERRVEC ;RESET ERROR TRAP

                                ;OUT INTERRUPTS (SET PRIORITY LEVEL 7)
                                ;GET DM11 PARAMETERS
002130 004737 003172      RSTAT1: JSR    7,@#DMPAR
002134 012706 001100      MOV    #SPBOT,%6
002140 104012              INITIALIZE
002142 023737 000042 000046      CMP    @#42,@#46    ;ACT11?
002150 001405              BEQ    PRGNUM+2
002152 104000              TYPE
002154 017436              M3
002156 004537 004124      PRGNUM: JSR    5,RECD    ;GET PRGNUM AND PUT IT
                                ;HERE
002162 000000              MOV    0
002164 043737 001522 002162      BIC    PRGLIM,PRGNUM ;MASK OFF UNUSED BITS
002172 006337 002162      ASL    PRGNUM       ;SHIFT PROGRAM #
002176 012737 004304 000024      MOV    #PFAIL,24
002204 012737 000340 000026      MOV    #PTY7,26
002212 013700 002162      MOV    PRGNUM,%0    ;GET PROGRAM #
002216 000170 001524      JMP    @PRGTAB(0)   ;GO START PROGRAM
002222 012737 004304 000024      RSTAT2: MOV    #PFAIL,24
002230 012737 000340 000026      MOV    #PTY7,26
002236 012706 001100      MOV    #SPBOT,%6
002242 104012              INITIALIZE
002244 013700 002162      MOV    PRGNUM,%0    ;GET PROGRAM #
002250 000170 001532      JMP    @RSTART(0)  ;GO RESTART PROGRAM
002254 022737 000176 001102      SRSET: CMP    #SWREG,SWR
002262 001410              BEQ    1$
002264 023737 000042 000046      CMP    @#42,@#46    ;ACT11?
002272 001405              BEQ    GETRDY
002274 104000              TYPE                ;TYPE MESSAGE TO REQUEST SWITCH
002276 017454              M3                  ;REGISTER SETTINGS
002300 000000              HALT                ;WAIT FOR OPERATOR TO SET SWITCHES
002302 000401              BR     GETRDY
002304 104015      1$:   CNTLU
002306 013737 001506 001514      GETRDY: MOV    KSTART,NXTST ;ADDR OF 1ST ROUTINE TO NXTST
002314 012737 000006 000004      GTRDYX: MOV    #6,@#ERRVEC ;RESET ERROR TRAP VECTOR
002322 104012              INITIALIZE
002324 004737 002554      GTRDYA: JSR    %7,FORWD ;ROLL FORWARD TO "NEXT" ROUTINE.
002330 032777 001000 176544      BIT    #BIT9,@SWR   ;CHECK SELECT ROUTINE SWITCH
002336 001003      BNE    GTRDYC      ;BRANCH IF SELECT ROUTINE SWITCH IS SET.
  
```

```

002340 000177 177144          JMP      @CURTST      ;GO RUN CURRENT ROUTINE.
002344 000457                BR       SCOPED      ;NO GO. MANUAL RTN BYPASSED.
002346 017700 176530          GTRDYC: MOV      @SWR,%0 ;(SR) TO R0
002352 042700 177600          BIC     #177600,%0  ;MASK UNDESIRED BITS
002356 123700 001512          CMPB   RTNNO,%0    ;COMPARE RTNNO TO (R0)
002362 001002                BNE    GTRDYD      ;BRANCH IF ROUTINE NOT FOUND YET.
002364 000177 177120          JMP     @CURTST      ;GO RUN ROUTINE.
002370 022737 177777 001514  GTRDYD: CMP     #-1,NXTST ;NO. CHECK FOR LAST ROUTINE.
002376 001352                BNE    GTRDYA      ;BRANCH IF NOT LAST ROUTINE.
002400 004737 001626          JSR    %7,INCRTN   ;YES. INCORRECT ROUTINE SELECTED.
002404 000740                BR     GETRDY      ;START OVER.

;SCOPE SERVICE ROUTINE
002406 000240          ESCOPE: NOP
002410 104014          KBDIN
002412 005077 176634          CLR    @BAR        ;CHECK FOR G
002416 005077 176626          CLR    @CSR        ;CLEAR ALL DM11 REGISTERS
002422 005077 176626          CLR    @BKCSR      ;AND SET BASE REGISTER
002426 104012          INITIALIZE        ;AT THE STARTING ADDRESS
002430 013716 001520          MOV    SCOPTR,(SP)
002434 032777 040000 176440  BIT    #BIT14,@SWR ;CHECK FOR SCOPE OPTION.
002442 001402                BEQ    SCOPEB      ;BRANCH IF SCOPE SW NOT SET.
002444 000176 000000          SCOPEA: JMP     @(SP)   ;RETURN TO ROUTINE
002450 032777 004000 176424  SCOPEB: BIT    #BIT11,@SWR ;TEST INHIBIT ITERATION SWITCH
002456 001012                BNE    SCOPED      ;BRANCH IF INHIBIT ITERATION SW SET.
002460 023737 000042 000046  CMP    @#42,@#46   ;ACT11?
002466 001003                BNE    1$         ;BR IF NO
002470 005737 001624          TST    @#PASS      ;1ST PASS?
002474 001403                BEQ    SCOPED      ;BR IF YES
002476 005337 001516          1$:   DEC    ICTR     ;DECREMENT ITERATION COUNT.
002502 001360                BNE    SCOPEA      ;BRANCH IF COUNT NOT 0.
002504 032777 001000 176370  SCOPED: BIT    #BIT9,@SWR ;CHECK SELECT ROUTINE SWITCH
002512 001275                BNE    GETRDY      ;BRANCH IF SELECT RTN SW SET
002514 022737 177777 001514  CMP    #-1,NXTST   ;LAST TEST?
002522 001274                BNE    GTRDYX      ;BRANCH IF NOT LAST TEST.
002524 005237 001624          INC    @#PASS
002530 104000          TYPE
002532 017447          M2
002534 013700 000042          MOV    @#42,%0    ;CHECK XXDP/ACT11 MONITOR HOOK
002540 001662                BEQ    GETRDY
002542 000005          RESET
002544 004710          SENDAD: JSR    7,(0) ;RETURN TO XXDP/ACT11 MONITOR
002546 000240          NOP
002550 000240          NOP
002552 000240          NOP
002554 013705 001514          FORWD: MOV    NXTST,%5 ;ADDR. OF NEXT ROUTINE TO R5.
002560 012537 001512          MOV    (5)+,RTNNO ;GET NEXT ROUTINE NUMBER.
002564 012537 001514          MOV    (5)+,NXTST ;GET ADDR OF NEXT "NEXT" ROUTINE.
002570 012537 001516          MOV    (5)+,ICTR  ;GET ITERATION COUNT.
002574 012537 001520          MOV    (5)+,SCOPT ;GET SCOPE LOOP ENTRY POINTER.
002600 010537 001510          FORWDA: MOV    %5,CURTST ;ADDR OF NOW CURRENT TEST TO CURTST.
002604 000207          RTS    %7        ;EXIT FORWD SUBROUTINE.

;EMT TRAP INTERPRETER
002606 011646          EMTINT: MOV    (6),-(6) ;GET PC OF NEXT INSTRUCTION
    
```

```

002610 162716 000002      SUB      #2,(6)      ;FORM PC OF EMT INSTRUCTION
002614 011637 001606      MOV      (6),PCADD  ;GET PC OF EMT INSTRUCTION
002620 017616 000000      MOV      @ (6),(6)  ;GET EMT INSTRUCTION
002624 105066 000001      CLRB    1(6)       ;CLEAR MSH OF EMT INSTRUCTION
002630 006316          ASL      (6)       ;SHIFT EMT IDENTIFIER
002632 062716 001540      ADD      #EMTTAB,(6)
002636 013607          MOV      @ (6)+,%7  ;GO TO PROPER EMT
  
```

```

;SAVE REGS 0 TO 4 SUBROUTINE.
002640 012637 002674      SAVRG:  MOV      (6)+,SVRPC  ;SAVE PC AND PSW
002644 012637 002676      MOV      (6)+,SVRPSW
002650 010446          MOV      %4,-(6)    ;SAVE REGS 0 - 4
002652 010346          MOV      %3,-(6)    ;IN STACK
002654 010246          MOV      %2,-(6)
002656 010146          MOV      %1,-(6)
002660 010046          MOV      %0,-(6)
002662 013746 002676      MOV      SVRPSW,-(6) ;RESTORE PC AND PSW
002666 013746 002674      MOV      SVRPC,-(6)
002672 000002          RTI
002674 000000          SVRPC: OPEN
002676 000000          SVRPSW: OPEN
  
```

```

;RESTORE REGS 0 TO 4 SUBROUTINE
002700 012637 002734      RSTRG:  MOV      (6)+,RSTPC  ;SAVE PC AND PSW
002704 012637 002736      MOV      (6)+,RSTPSW
002710 012600          MOV      (6)+,%0    ;PESTORE REGS 0 - 4
002712 012601          MOV      (6)+,%1    ;FROM STACK
002714 012602          MOV      (6)+,%2
002716 012603          MOV      (6)+,%3
002720 012604          MOV      (6)+,%4
  
```

```

002722 013746 002736      MOV      RSTPSW,-(6) ;RESTORE PC AND PSW
002726 013746 002734      MOV      RSTPC,-(6)
002732 000002          RTI
002734 000000          RSTPC: OPEN
002736 000000          RSTPSW: OPEN
  
```

```

;ROUTINE TO ISSUE RESET.
002740 012700 052525      SPSETT: MOV      #52525,%0  ;DATA TO R0
002744 005100          COM      %0          ;COMPLEMENT (R0)
002746 010037 002742      MOV      %0,SRSETT+2 ;(R0) TO SRSETT+2
002752 000005          RESET
002754 000002          RTI                ;ISSUE RESET. (R0) IS
                                ;DISPLAYED. EXIT
  
```

```

;RANDOM NUMBER GENERATOR. ROUTINE EXITS WITH NUMBER IN REGISTER 0.
002756 013700 003024      RNGEN:  MOV      RP1,%0
002762 006100          ROL      %0
002764 006100          ROL      %0
002766 063700 003026      ADD      RP2,%0
002772 010037 003024      MOV      %0,RP1
002776 006100          ROL      %0
003000 006100          ROL      %0
003002 063700 003026      ADD      RP2,%0
003006 006100          ROL      %0
003010 006100          ROL      %0
003012 010037 003026      MOV      %0,RP2
  
```

```

003016 013700 003024      MOV      RP1,%0
003022 000207      RTS      %7          ;EXIT. NUMBER IN RO
003024 001233      RP1:    1233
003026 007622      RP2:    7622
;SUBROUTINE TO OUTPUT ASCII MESSAGE ON TELETYPE PRINTER.
TYP:  MOV      @%6,%0      ;GET ADDRESS THAT CONTAINS MESSAGE ADDRESS.
      ADD      #2,@%6      ;SET UP EXIT.
      MOV      @%0,%0      ;ADDRESS OF MESSAGE TO RO.
TYPA: MOVB     (0)+,TYPDAT  ;GET CHARACTER
      CMPB    #100,TYPDAT  ;CHECK FOR"@"CHARACTER
      BNE     TYPC         ;BRANCH IF NOT"@".
      RTI     TYPD         ;TERMINATOR CHAR. DONE. EXIT.
TYPC: CMPB    #45,TYPDAT  ;CHECK FOR"%".
      BEQ     TYPF         ;BRANCH IF"%".
      JSR    %7,TYPD      ;TYPE CHAR IN TYPDAT
      BR     TYPA
TYPD: MOVB    TYPDAT,@TPDBR ;OUTPUT CHARACTER TO PRINTER
      TSTB   @TPCSR      ;WAIT FOR DONE FLAG.
      BPL    -4
      RTS    %7          ;EXIT
TYPF: MOVB    #15,TYPDAT  ;MOVE CARRIAGE RETURN CODE TO TYPDAT
      JSR    %7,TYPD      ;GO TYPE CHAR.
TYPG: MOVB    #12,TYPDAT  ;MOVE LF CODE TO TYPDAT.
      JSR    %7,TYPD      ;GO TYPE CHAR.
      BR     TYPA
TYPDAT: OPEN
  
```

;SUBROUTINE TO INITIALIZE STACK POINTER AND SET PROCESSOR PRIORITY  
 ;LEVEL 7

```

003142 012777 001106 176106 INIT:  MOV      #CAT,@BASREG  ;INITIALIZE THE BASE REGISTER
003150 012737 000340 177776      MOV      #PTY7,PSW      ;SET PRIORITY LEVEL 7
003156 011637 001100      MOV      (SP),SPBOT     ;GET RETURN ADDRESS
003162 012706 001100      MOV      #SPBOT,SP      ;SET BOTTOM OF THE STACK
003166 000176 000000      JMP      @SP            ;RETURN
  
```

;SUBROUTINE TO GET DM11 PARAMETERS  
 ;VECTOR ADDRESS

```

003172 023737 000042 000046 DMPAR: CMP      @#42,@#46      ;ACT1!?
003200 001060      BNE     6$             ;BR IF NO
;SIZE FOR INTERRUPT VECTOR IN AUTO MODE
      MOV      #302,RO    ;SET UP FLOATING VECT AREA
4$:   MOV      RO,-2(RO)
      MOV      #3,(RO)+
      TST     (RO)+
      CMP     #776,RO
      BPL    4$
      MOV     #55,@#14    ;SET BPT VECT
      MOV     #340,@#16   ;& PSW
3$:   MOV     #-1,WCT      ;SET TO XMIT 1 CHAR
      MOV     #OUTBUF,CAT
      MOV     #BIT6+BIT2+BIT0,@CSR ;SET IE
      CLR    @#PSW        ;LVL 0
      MOV     #LBIT0,@BAR  ;XMIT
      MOV     #-1,COUNT    ;WAIT
  
```

```

003304 005337 001604      25:  DEC      COUNT
003310 001375              BNE      25
003312 104001              ERROR
003314 000752              BR       35
003316 162716 000004      55:  SUB      #1, (SP)
003322 011637 003356      MOV      (SP), @#VECTOR
003326 012737 000016 000014  MOV      #16, @#14
003334 004737 004100      JSR      7, OVLAY
003340 000415              BR       VECOK
003342 004737 004100      65:  JSR      7, OVLAY
003346 104000              TYPE
003350 017310              WHERE
003352 004537 004124      JSR      5, RECD
003356 000000      VECTOR:  0
003360 005737 003356      TST      VECTOR
003364 001003              BNE      VECOK
003366 012737 000300 003356  MOV      #300, VECTOR
003374 023727 003356 000300  VECOK:  CMP      VECTOR, #300
003402 103003              BHIS     VECOKB
003404 104000      VECOKA:  TYPE
003406 017444              M1
003410 000670              BR       DMPAR
003412 023727 003356 000770  VECOKB:  CMP      VECTOR, #770
003420 101371              BHI     VECOKA
003422 032737 000007 003356  BIT      #7, VECTOR
003430 001365              BNE     VECOKA
003432 013737 003356 001260  MOV      VECTOR, CLKINT
003440 062737 000004 003356  ADD      #4, VECTOR
003446 013737 003356 001264  MOV      VECTOR, XMTINT

; UNIT NUMBER
003454 023737 000042 000046  DMPARB:  CMP      @#42, @#46
003462 001405              BEQ     UNIT+2
003464 104000              TYPE
003466 017352              WHICH
003470 004537 004124      JSR      5, RECD
003474 000000      UNIT:   0
003476 023727 003474 000017  CMP      UNIT, #17
003504 101403              BLOS    UNTOKA
003506 104000              TYPE
003510 017444              M1
003512 000760              BR       DMPARB
003514 006337 003474      UNTOKA:  ASL     UNIT
003520 006337 003474      ASL     UNIT
003524 006337 003474      ASL     UNIT
003530 012702 000004      MOV     #4, %2
003534 012701 001250      MOV     #CSR, %1
003540 042711 000370      UNTOKB:  BIC     #370, (1)
003544 063721 003474      ADD     UNIT, (1)+
003550 005302              DEC     %2
003552 001372              BNE     UNTOKB

003554 012702 000004      MOV     #4, %2
003560 012703 001250      MOV     #CSR, %3
003564 012701 001614      MOV     #CSR, %1
003570 012311      UNTOKC:  MOV     (3)+, (1)
; FORM ODD BYTE ADDRESSES
    
```



```

003572 005221          INC      (1)+
003574 005302          DEC      %2
003576 001374          BNE     UNTOKC
; CHARACTER LENGTH
003600 023737 000042 000046 DMPARC: CMP     @#42,@#46 ;ACT11?
003606 001405          BEQ     LENGTH+2 ;BR IF YES
003610 104000          TYPE
003612 017365          LEVEL
003614 004537 004124          JSR     5,RECD ;GET LENGTH AND PUT IT
003620 000000          LENGTH: 0 ;HERE
003622 005737 003620          TST     LENGTH
003626 001003          BNE     LENOKA
003630 012737 000010 003620          MOV     #8,LENGTH
003636 023727 003620 000005 LENOKA: CMP     LENGTH,#5 ;CHARACTER LENGTH SELECTED MUST
003644 103003          BHS     LENOKC ;BE BETWEEN 5-8
003646 104000          LENOKB: TYPE ;CARRIAGE RETURN SELECTS 8
003650 017444          MI
003652 000752          BR     DMPARC
003654 023727 003620 000010 LENOKC: CMP     LENGTH,#8
003662 101371          BHI     LENOKB
003664 162737 000005 003620          SUB     #5,LENGTH
003672 006337 003620          ASL     LENGTH
003676 013701 003620          MOV     LENGTH,%1
003702 016137 003714 001300          MOV     LENOKD(1),@#CARMSK ;SET CHARACTER LENGTH MASK
003710 000240          NOP
003712 000207          RTS     7 ;EXIT PARAMETERS ROUTINE

```

; THE BELOW TABLE REPRESENTS THE CHARACTER LENGTH MASK FOR 5, 6, 7, AND 8  
 ; BITS PER CHARACTER RESPECTIVELY.

```

003714 177740          LENOKD: 177740
003716 177700          177700
003720 177600          177600
003722 177400          177400

```

; CALCULATE MACHINE TIME TO TRANSMIT ONE CHARACTER

```

003724 005037 004074          TIMER: CLR     TIME1
003730 012737 177777 001146          MOV     #-1,WCT ;SET UP TO TRANSMIT
003736 012737 016564 001106          MOV     #OUTBUF,CAT ;1 CHARACTER ON LINE 1
003744 012777 004044 175306          MOV     #TIMEC,@CLKINT ;LOAD RECEIVER INTERRUPT
003752 012777 000340 175302          MOV     #PTY7,@CLKLVL ;AND PRIORITY
003760 012777 000001 175264          MOV     #LBIT0,@BAR ;START TRANSMITTING
003766 012777 000105 175254          MOV     #BIT6+BIT2+BIT0,@CSR ;SET IE BIT
003774 005037 177776          CLR     @PSW ;SET PROCESSER PRIORITY LEVEL = 0
004000 012737 000044 001604 TIMER: MOV     #4,COUNT
004006 062737 000001 004074          ADD     #1,TIME1 ;INCREMENT MACH. TIME TO TRANSMIT
004014 001007          BNE     TIMEB
004016 005077 175226          CLR     @CSR
004022 012737 000340 177776          MOV     #PTY7,PSW ;SET PROCESSER PRIORITY LEVEL = 7
004030 104001          ERROR ;TRANSMITTER FAILED TO INTERRUPT
004032 000734          BR     TIMER
004034 005337 001604          TIMEB: DEC     COUNT
004040 001375          BNE     -4
004042 000756          BR     TIMEA
004044 005077 175200          TIMEC: CLR     @CSR
004050 013737 004074 004076          MOV     TIME1,TIME14
004056 006037 004076          ROR     TIME14

```

```

004062 000241          CLC
004064 006037 004076  ROR    TIME14
004070 022626          POPSP2      ;RESTORE STACK POINTER
004072 000207          RTS     7        ;EXIT TIME CALCULATION ROUTINE

004074 000000          TIME1:  OPEN      ;CONTAINS MACHINE TIME TO XMIT 1 CHAR
004076 000000          TIME14: OPEN      ;CONTAIN TIME TO XMIT-1/4 CHAR

;SUBROUTINE TO PUT HALT, +2 IN VECTOR AREA (0300-1000)
004100 012702 000302  OVRLAY: MOV    #302,R2
004104 010262 177776 15:    MOV    R2,-2(R2)
004110 005022          CLR    (R2)+
004112 005722          TST   (R2)+
004114 022702 000776  CMP    #776,R2
004120 100371          BPL   15
004122 000207          RTS   7
  
```

;SUBROUTINE TO RECEIVE DATA  
 ;THIS SUBROUTINE RECEIVES DATA FROM THE KEYBOARD (UP TO SIX OCTAL  
 ;DIGITS AND PLACES THEM INTO THE ADDRESS FOLLOWING THE SUBROUTINE  
 ;CALL (JSR 5,RECD). NO REGISTER CONTENTS ARE DISTURBED.

;SUBROUTINE TO INPUT DATA FROM TTY

```

004124 010046          RECD:  MOV    R0,-(SP)
004126 005015          15:    CLR    (5)          ;CLEAR OLD DATA
004130 012737 000007 004302  MOV    #7,CNT      ;SET CHAR COUNT
004136 105777 175432 25:    TSTB  @TKCSR      ;WAIT FOR CHAR
004142 100375          BPL   25
004144 117700 175426  MOVB  @TKDBR,R0
004150 142700 000200  BICB  #200,R0      ;STRIP OFF PARITY
004154 110077 175422  MOVB  R0,@TPDBR    ;ECHO CHARACTER
004160 122700 000025  CMPB  #25,R0      ;IS IT A U
004164 001443          BEQ   55          ;BRANCH IF YES
004166 122700 000015  CMPB  #15,R0      ;IS IT A <CR>
004172 001415          BEQ   65          ;BRANCH IF YESS
004174 142700 000060  BICB  #60,R0
004200 132700 000110  BITB  #110,R0     ;CHECK FOR 0-7 (8)
004204 001031          BNE  75          ;BRANCH IF NOT
004206 006315          ASL   (5)
004210 006315          ASL   (5)
004212 006315          ASL   (5)          ;SHIFT DATA
004214 150015          BISB  R0,(5)      ;INSET NEW CHAR
004216 005337 004302  DEC   CNT
004222 001422          BEQ   73          ;ONLY 6 CHAR'S PLEASE
004224 000744          BR   25          ;NEXT CHARACTER
004226 105777 175346 65:    TSTB  @TPCSR
004232 100375          BPL   65          ;WAIT FOR READY
004234 012777 000012 175340  MOV    #12,@TPDBR ;TYPE <LF>
004242 105777 175332 85:    TSTB  @TPCSR
004246 100327          BPL   15          ;WAIT FOR READY
004250 005077 175326          CLR   @TPDBR    ;LOAD CHAR
004254 105777 175320 95:    TSTB  @TPCSR
004260 100375          BPL   95
  
```

```

004262 005725          TST      (R5)+          ;ADJUST R5
004264 012600          MOV      (SP)+,R0       ;RESTORE R0
004266 000205          RTS      R5
004270 104000          75:     TYPE
004272 017444          M1
004274 104000          55:     TYPE
004276 017350          SCTLU
004300 000712          BR      15             ;START OVER
004302 000000          CNT:    0
  
```

```

004304 012737 004314 000024 ;POWER FAIL ROUTINE
004312 000000 PFAIL:  MOV      #PWRUP,24
                          HALT
  
```

```

004314 000005          ;POWER UP SUBROUTINE
004316 012706 001100 PWRUP:  RESET          ;GIVE TELEPRINTER TIME TO START
004322 104001          MOV      #SPBOT,%6
004324 000137 002222  ERROR          ;TYPE POWER FAIL ERROR
                          JMP      @#RSTAT2      ;GO RESTART PROGRAM
  
```

;LINE TEST SUBROUTINE: THIS LINE TEST PROVIDES SEVERAL TESTS ON A DM11 LINE.  
 ;THE SUBROUTINE IS CALLED BY JSR 5, LNTST. THIS INSTRUCTION PROVIDES THE  
 ;LINE BIT AND LINE NUMBER. THE FOLLOWING LINE TESTS ARE PERFORMED:  
 ;WAITS UNTIL CHARACTER SHOULD HAVE BEEN TRANSMITTED, THEN TESTS

```

                          ;THAT BAR BIT CLEARED          ;DO NEXT TEST IF ERROR
                          ;READY SET                    ;DO NEXT TEST IF ERROR
                          ;WORD COUNT WENT TO 0        ;DO NEXT TEST IF ERROR
                          ;CURRENT ADDRESS DID NOT INCREMENT ;DO NEXT TEST IF ERROR
                          ; INTERRUPTS TO CORRECT VECTOR ;DO NEXT TEST IF ERROR (NO INTERRUPT)
                          ;READY BIT CAN BE CLEARED    ;END OF TEST
004330 012537 016412 XMTTST: MOV      (5)+,@#LINE ;GET LINE NUMBER
004334 004737 006552 JSR      7,GTLINB ;GO FORM LINE BIT (FOR BAR)
004340 005037 001276 CLR      XMTDAT
004344 004537 005754 15:     JSR      5,@#XMITD ;GO TO TRANSMIT SUBROUTINE
004350 177777          -1          ;TRANSMIT ONE CHARACTER
004352 012703 000010 MOV      #10,%3 ;WAIT IN
004356 005002          CLR      %2 ;THIS
004360 005302          25:     DEC      %2 ;LOOP
004362 001376          BNE     -2 ;UNTIL THE
004364 005303          DEC      %3 ;TRANSMITTER
004366 001374          BNE     25 ;IS FINISHED
004370 017737 174656 001274 MOV      @BAR,PCVDAT ;BAR SHOULD NOW BE CLEAR
004376 001401          BEQ     35 ;BRANCH IF IT IS
004400 104011          ERROR1 ;ERROR! BAR BIT FAILED TO CLEAR
004402 005777 174642 35:     TST      @CSR ;TEST READY BIT SHOULD BE SET
004406 100401          BMI     45 ;BRANCH IF SET
004410 104001          ERROR ;ERROR! READY NOT SET
004412 013701 016412 45:     MOV      @#LINE,%1 ;GET LINE NUMBER
004416 016137 001146 001274 MOV      WCT(1),RCVDAT ;WORD COUNT SHOULD BE 0
004424 001401          BEQ     55
004426 104011          ERROR1 ;ERROR! WORD COUNT NOT EQUAL TO 0
004430 012737 016564 001276 55:     MOV      #OUTBUF,XMTDAT ;
004436 016137 001106 001274 MOV      CAT(1),RCVDAT ;CURRENT ADDRESS SHOULD NOT HAVE INCREMENTED
004444 023737 001274 001276 CMP      RCVDAT,XMTDAT ;
004452 001401          BEQ     65
  
```

```
004454 104011          ERROR1          ;ERROR! CURRENT ADDR. DID NOT INCREMENT
004456 012777 004510 174600 65:  MOV      #7$, @XMTINT ;LOAD TRANSMITTER INTERRUPT VECTOR
004464 052777 010000 174556      BIS      #BIT12, @CSR ;ENABLE TRANSMITTER INTERRUPT
004472 005037 177776          CLR      @#PSW      ;SET PROCESSOR PRIORITY =0
004476 000240          NOP
004500 012737 000340 177776      MOV      #PRTY7, @#PSW ;LOCK OUT INTERRUPTS
004506 104001          ERROR          ;TRANSMITTER FAILED TO INTERRUPT OR
; INTERRUPTED TO WRONG LOCATION AND HALTED WITH ADDRESS +2 DISPLAYED.
004510 022626          75:  CMP      (6)+, (6)+ ;RESET STACK PTR
004512 012737 000340 177776      MOV      #PRTY7, @#PSW ;LOCK OUT INTERRUPTS
004520 042777 110000 174522      BIC      #BIT12+BIT15, @CSR ;CLEAR XMIT IE & READY BITS
004526 005777 174516          TST      @CSR      ;TEST THAT READY CLEARED
004532 100001          BPL      85        ;GO TO EXIT
004534 104001          ERROR          ;ERROR! READY FAILED TO CLEAR
004536 005726          85:  TST      (6)+      ;RESET STACK PTR
004540 104006          SCOPE          ;SCOPE
```

;RECEIVER LINE TESTS  
 ;THE RECEIVER LINE TEST SUBROUTINE IS ENTERED WITH  
 ;A JSR 5, RCVTST INSTRUCTION FOLLOWED BY THE  
 ;LINE BIT AND LINE NUMBER OF THE LINE TO BE  
 ;TESTED. THE SUBROUTINE PERFORMS THE FOLLOWING  
 ;TEST AS SHOWN BELOW. IN THE EVENT OF AN ERROR  
 ;THE REMAINING TESTS ARE ABORTED  
 ;TEST SEQUENCE AND ADDRESS TAG

;	CHARACTER DONE SETS	RTSTA
;	CHARACTER DONE CAUSES INTERRUPT	RTSTB
;	CHARACTER DONE CAN BE CLEARED	RTSTC
;	TUMBLE TABLE ENTRY IS CORRECT	RTSTD
;	NO ENTRY IN NEXT TABLE ADDRESS	RTSTE
;	HARDWARE TABLE POINTER INCREMENTED	RTSTF
;	NEXT ENTRY WAS CORRECT	RTSTG

;NOTES: IF THE HARDWARE PROVIDES AN INCORRECT VECTOR  
 ; ADDRESS THE PROGRAM WILL HALT AND DISPLAY  
 ; THE INCORRECT VECTOR+2 IN THE ADDRESS LIGHTS.

004542	012737	177777	016564	RCVTST:	MOV	#-1,OUTBUF	;LOAD ALL 1'S INTO OUTPUT BUFFER
004550	005037	001306			CLR	TUMTAB	;CLEAR THE FIRST
004554	005037	001310			CLR	TUMTAB+2	;TWO TUMBLE TABLE ADDRESSES
004560	012737	000340	177776		MOV	#PRTY7,@#PSW	;LOCK OUT INTERRUPTS
004566	012537	016412			MOV	(5)+,LINE	;GET LINE NUMBER
004572	004537	005754			JSR	5,@#XMITD	;TRANSMIT 1 CHARACTER (0'S)
004576	177777				-1		;ON LINE SPECIFIED BY JSR
004600	052777	000001	174442		BIS	#BIT0,@CSR	;SET GO BIT
004606	005777	174436			TST	@CSR	;WAIT FOR TRANSMITTER
004612	100375				BPL	-4	;TO TRANSMIT 1 CHAR.
004614	042777	100000	174426		BIC	#BIT15,@CSR	;CLEAR TRANSMITTER READY FLAG
004622	005046				CLP	-(SP)	;SET WATCH DOG TIMER
004624	105777	174420		15:	TSTB	@CSR	;TEST CHAR. DONE FLAG
004630	100404				BMI	25	;BRANCH IF SET
004632	005216				INC	(SP)	;WAIT FOR THE FLAG
004634	001373				BNE	15	
004636	104001				ERROR		;ERROP! CHAR. DONE FLAG FAILED TO SET
004640	000550				BR	85	;GO TO EXIT
004642	005726			25:	TST	(SP)+	;RESTORE STACK PTR
004644	012777	004700	174406		MOV	#35,@CLKINT	;LOAD RECEIVER INTERRUPT VEC. ADRS.
004652	052777	000100	174370		BIS	#BIT6,@CSR	;SET RECEIVER IE BIT
004660	005037	177776			CLR	@#PSW	;ENABLE INTERRUPTS
004664	000240				NOP		
004666	012737	000340	177776		MOV	#PRTY7,PSW	;LOCK OUT INTERRUPTS
004674	104001				ERROR		;RECEIVER FAILED TO INTERRUPT
004676	000531				BR	85	;GO TO EXIT
004700	012737	000340	177776	35:	MOV	#PRTY7,@#PSW	;LOCK OUT INTERRUPTS
004706	022626				CMP	(6)+,(6)+	
004710	042777	000300	174332		BIC	#BIT7+BIT6,@CSR	;CLEAR CHAR. DONE FLAG
004716	105777	174326			TSTB	@CSR	;TEST THAT CHAR DONE FLAG CLEARED
004722	106002				BPL	45	;BRANCH IF CHAR. DONE FLAG CLEARED
004724	104001				ERROR		;ERROR! CHAR. DONE FAILED TO CLEAR
004726	000515				BR	85	;GO TO EXIT

```

004730 013737 001306 001274 45:  MOV  TUMTAB,RCV DAT ;GET TUMBLE TABLE ENTRY
004736 042737 020000 001274      BIC  #BIT13,RCV DAT ;CLEAR PARITY INDICATOR
004744 012737 000377 001276      MOV  #377,XMT DAT ;LOAD XMT DAT WITH TRANSMITTED DATA
004752 043737 001300 001276      BIC  CARMSK,XMT DAT ;CLEAR NON TRANSMITTED BITS
004760 153737 016412 001277      BISB LINE,XMT DAT+1 ;LOAD XMT DAT WITH PROPER LINE #
004766 052737 100000 001276      BIS  #BIT15,XMT DAT ;SET VALID DATA ENTRY BIT IN XMT DAT
004774 023737 001274 001276      CMP  RCV DAT,XMT DAT ;COMPARE TUMBLE TABLE ENTRY (RCV DAT) &
                                ;CORRECT RESULT (XMT DAT)

005002 001402      BEQ  55
005004 104011      ERROR1
005006 000465      BR   85 ;ERROR! INCORRECT TUMBLE TABLE
                                ;ENTRY; GO TO EXIT
005010 005037 001306      55:  CLR  TUMTAB
005014 013737 001310 001274      MOV  TUMTAB+2,RCV DAT ;GET NEXT ENTRY
005022 001404      BEQ  65 ;BRANCH IF ALL 0'S
005024 005037 001276      CLR  XMT DAT
005030 104011      ERROR1
005032 000453      BR   85 ;ERROR! FALSE ENTRY IN NEXT
                                ;TUMBLE TABLE ADDRESS
005034 004537 005754      65:  JSR  5,@#XMITD ;TRANSMIT 1 CHARACTER (ALL 1'S)
                                ;ON LINE SPECIFIED BY JSR
005040 177777      -1
005042 005777 174202      TST  @CSR ;WAIT FOR TRANSMITTER
005046 100375      BPL  -4 ;READY FLAG
005050 105777 174174      TSTB @CSR ;TEST FOR THE DONE FLAG
005054 100375      BPL  -4
005056 042777 000200 174164      BIC  #BIT7,@CSR ;CLEAR CHAR. DONE FLAG
005064 013737 001306 001274      MOV  TUMTAB,RCV DAT ;TEST THAT HARDWARE TUMBLE
005072 001404      BEQ  75 ;TABLE POINTER INCREMENTED (+2)
005074 005037 001276      CLR  XMT DAT
005100 104011      ERROR1
005102 000427      BR   85 ;ERROR! TUMBLE TABLE POINTER DID
                                ;NOT INCREMENT; GO TO EXIT
005104 013737 001310 001274 75:  MOV  TUMTAB+2,RCV DAT ;GET TUMBLE TABLE ENTRY
005112 042737 020000 001274      BIC  #BIT13,RCV DAT ;CLEAR PARITY INDICATOR
005120 012737 000377 001276      MOV  #377,XMT DAT ;LOAD XMT DAT WITH TRANSMITTED DATA
005126 043737 001300 001276      BIC  CARMSK,XMT DAT ;CLEAR NON-TRANSMITTED BITS
005134 153737 016412 001277      BISB LINE,XMT DAT+1 ;LOAD LINE # INTO XMT DAT
005142 052737 100000 001276      BIS  #BIT15,XMT DAT ;SET VALID DATA ENTRY BIT INTO XMT DAT
005150 023737 001274 001276      CMP  RCV DAT,XMT DAT ;COMPARE TUMBLE TABLE ENTRY (RCV DAT) &
                                ;CORRECT RESULT (XMT DAT)

005156 001401      BEQ  85
005160 104011      ERROR1
005162 104006      85:  SCOPE ;ERROP! 2ND TUMBLE TABLE ENTRY
                                ;WAS INCORRECT; SCOPE
    
```

```

;SUBROUTINE TO TEST BREAK OPERATION
;THE TRANSMITTER WILL TRANSMIT THE BREAK FOR TWO CHARACTER
;TIMES AND THEN THE FOLLOWING TESTS WILL BE PERFORMED
;
; A VALID DATA ENTRY WAS MADE      BKTSTB
; BREAK BIT SET                      BKTSTC
; DATA WAS ALL 0'S                 BKTSTD
    
```

```

005164 012777 000001 174056 8RKTST: MOV  #1,@CSR ;SET THE GO BIT
005172 011577 174056      MOV  (5),@BKCSR ;SET THE BREAK BIT
005176 105777 174046      TSTB @CSR ;WAIT FOR THE RECEIVER TO
005202 100375      BPL  -4 ;RECEIVE BREAK
005204 042777 000200 174036      BIC  #BIT7,@CSR ;CLEAR FLAG
005212 105777 174032      TSTB @CSR ;WAIT FOR THE RECEIVER TO
005216 100375      BPL  -4 ;TO RECEIVE BREAK
    
```

```

005220 042777 000200 174022      BIC    #BIT7,@CSR      ;CLEAR FLAG
005226 005077 174022          CLR    @BKCSR         ;CLEAR BREAK BIT
005232 005737 001306      15:   TST    TUMTAB       ;TEST FOR VALID DATA ENTRY
005236 100402          BMI    25
005240 104001          ERROR          ;ERROR! NO VALID DATA ENTRY
005242 000421          BR     45            ;GO TO EXIT
005244 032737 040000 001306  25:   BIT    #BIT14,TUMTAB ;TEST THAT BREAK BIT IS SET
005252 001002          BNE    35            ;IN TUMBLE TABLE
005254 104001          ERROR          ;ERROR! BREAK BIT FAILED TO SET
005256 000413          BR     45            ;GO TO EXIT
005260 105737 001306      35:   TSTB  TUMTAB       ;TEST THAT DATA IS ALL 0'S
005264 001410          BEQ    45
005266 005037 001274          CLR    RCVDAT
005272 113737 001306 001274  MOVB  TUMTAB,RCVDAT  ;GET RECEIVED DATA
005300 005037 001276          CLR    XMTDAT
005304 104011          ERROR1
005306 104006      45:   SCOPE          ;ERROR! DATA WAS NOT ALL 0'S
                                ;SCOPE
  
```

; SUBROUTINE TO TRANSMIT & RECEIVE ON ALL LINES THE DELAY BETWEEN  
 ; TRANSMITTING ON A LINE IS SUPPLIED BY THE CALLING JSR INSTRUCTION.  
 ; NOTE NO DATA CHECKING IS PERFORMED BY THIS TEST

```

005310 012537 001604      DLYXMT: MOV    (5)+,@#COUNT ;GET CHARACTER DELAY COUNT
005314 005037 001276          CLR    XMTDAT
005320 004537 005732          JSR    5,BMOVE       ;LOAD OUTPUT BUFFER WITH DATA
005324 017657          MSG1          ;TO BE TRANSMITTED
005326 016564          OUTBUF
005330 000100          64.
005332 012737 000001 001272  MOV    #LBIT0,@#LINBIT
005340 005037 016412          CLR    @#LINE
005344 012777 000001 173676  15:   MOV    #BIT0,@CSR     ;SET THE GO BIT
005352 004537 005754      25:   JSR    5,@#XMITD     ;TRANSMIT 64. CHAR.
                                ;ON A LINE
005356 177700          -64.
005360 013737 004074 005374  MOV    @#TIME1,45
005366 013704 001604          MOV    @#COUNT,%4   ;GET CHARACTER DELAY COUNT
005372 104400      35:   DELAY
005374 000000      45:   0
005376 005304          DEC    %4
005400 001374          BNE    35
005402 062737 000002 016412  ADD    #2,LINE        ;FORM NEXT LINE NUMBER
005410 006337 001272          ASL    LINBIT        ;SHIFT LINE BIT
005414 103356          BCC    25            ;BRANCH IF ALL LINES NOT DONE
005416 012704 000100          MOV    #64,%4
005422 013737 004074 005432  MOV    TIME1,65
005430 104400      55:   DELAY
005432 000000      65:   0
005434 005304          DEC    %4
005436 001374          BNE    55
005440 017737 173606 001274  MOV    @BAR,RCVDAT   ;GET & TEST BAR DATA
005446 001402          BEQ    75            ;EXIT IF DONE
005450 104011          ERROR1          ;ERROR! BAR SHOULD'VE BEEN CLEAR
005452 000413          BR     85
005454 022777 100201 173566  75:   CMP    #100201,@CSR  ;TEST THAT ONLY DONE,GO,& READY BITS ARE SET
005462 001407          BEQ    85
005464 012737 100201 001276  MOV    #100201,XMTDAT
005472 017737 173552 001274  MOV    @CSR,RCVDAT  ;GET CSR CONTENTS
  
```

```

005500 104011          ERROR1          ; INCORRECT CSR CONTENTS
005502 005726          8$: POPSP          ; RESET THE STACK
005504 104006          SCOPE          ; SCOPE

; SUBROUTINE TO DELAY A SPECIFIED NUMBER OF MILLISECONDS
005506 011637 005556  DLY: MOV      (6), 3$      ; GET DELAY COUNT ADDRESS.
005512 062716 000002  ADD      #2, (6)      ; SET UP EXIT ADDRESS
005516 017737 000034 005556  MOV      @3$, 3$      ; GET DELAY COUNT
005524 001413          BEQ      2$          ; EXIT IF NO DELAY
005526 012737 000050 005560 1$: MOV      #50, 4$
005534 162737 000001 005556  SUB      #1, 3$
005542 001404          BEQ      2$
005544 005337 005560  DEC      4$
005550 001375          BNE      -4
005552 000765          BR       1$
005554 000002          2$: RTI          ; EXIT
005556 000000          3$: OPEN          ; CONTAINS DELAY COUNT
005560 000000          4$: OPEN          ; CONTAINS DELAY ROUTINE CONSTANT

; SUBROUTINE TO INITIALIZE BINARY COUNT PATTERNS
005562 012737 177777 005604  INBIN: MOV      #-1, RIND      ; SET ALL VARIABLES
005570 004537 005732  JSR      %5, BMOVE          ; TO MINUS 1.
005574 005604          RIND
005576 005605          RIND+1
005600 000005          5
005602 000207          RTS      %7          ; EXIT
005604 000000          RIND: OPEN
005606 000000          PTO: OPEN
005610 000000          PT1: OPEN

; SPECIAL BINARY COUNT PATTERN SUBROUTINE. EXITS WITH BIN CHAR IN R1
005612 013737 005606 005610  GTBIN: MOV      PTO, PT1      ; PREVIOUS BIN CHAR TO PT1
005620 005137 005610  COM      PT1
005624 005137 005604  COM      RIND
005630 001002          BNE      +6
005632 005237 005610  INC      PT1
005636 042737 177400 005610  BIC      #177400, PT1      ; MASK TO 8 BITS
005644 013737 005610 005606  MOV      PT1, PTO          ; SAVE BIN CHAR IN PTO
005652 013701 005610  MOV      PT1, %1          ; BIN CHAR TO R1
005656 000240          NOP
005660 000207          RTS      %7          ; EXIT.

; OCTAL TO ASCII CONVERT ROUTINE
005662 104007          OACNV: SAVREG          ; SAVE REGISTERS ON THE STACK
005664 013504          MOV      @(%5)+, %4      ; GET OCTAL VALUE.
005666 012501          MOV      (%5)+, %1      ; GET DESTINATION ADDR.
005670 012502          MOV      (%5)+, %2      ; GET CONVERT COUNT.
005672 060201          ADD      %2, %1          ; DEVELOP ADDR TO STORE 1ST CHAR
005674 010403          OACNVA: MOV      %4, %3
005676 042703 177770  BIC      #177770, %3      ; ISOLATE LEAST SIGNIFICANT DIGIT.
005702 062703 000060  ADD      #60, %3          ; CONVERT DIGIT TO ASCII.
005706 110341          MOV      %3, -(1)        ; STORE ASCII CHARACTER.
005710 042704 000007  BIC      #7, %4
005714 006004          ROR      %4
005716 006004          ROP      %4
005720 006004          ROR      %4
  
```



```

005722 005302      DEC      %2      ;DONE ALL DIGITS?
005724 001363      BNE      0ACNVA   ;BRANCH IF NOT DONE.
005726 104010      RSTREG                ;RESTORE THE REGISTERS
005730 000205      RTS      %5      ;DONE. EXIT.
  
```

```

; SUBROUTINE TO MOVE A VARIABLE NUMBER OF BYTES.
BMOVE: SAVREG                ;SAVE REGS.
      MOV      (5)+,%1      ;GET"FROM"ADDRESS
      MOV      (5)+,%2      ;GET"TO"ADDRESS
      MOV      (5)+,%3      ;GET COUNT
15:   MOVB     (1)+,(2)+    ;MOVE BYTE
      DEC      %3          ;DECREMENT COUNT
      BNE     15          ;BRANCH IF NOT DONE.
      RSTREG                ;RESTORE REGS.
      RTS     %5          ;DONE EXIT
  
```

```

; SUBROUTINE TO TRANSMIT DATA. SUBROUTINE CALLED BY
; JSR 5,XMITD
XMITD: MOV      %0,-(SP)    ;SAVE RO ON THE STACK
      MOV      @#LINE,%0   ;GET LINE
      JSR     7,@#GTLINB   ;FORM LINE BIT (FOR BAR)
      MOV      #CAT,@BASREG ;INITIALIZE BASE REGISTER
      MOV      #OUTBUF,CAT(0) ;LOAD FIRST CHAR ADDRESS IN CAT
      MOV      (5)+,WCT(0)  ;GET WORD COUNT
      BIS     @#LINBIT,@BAR ;LOAD LINE POSITION INTO BAR
      MOV      (SP)+,%0     ;RESTORE RO
      RTS     5            ;EXIT
  
```

```

; ROUTINE TO TEST A LINE.
; THE LINE TO BE TESTED IS PROVIDED BY THE JSR CALL TO THE ROUTINE.
; 100 CHARACTERS ARE TRANSMITTED, RECEIVED AND CHECKED BY THIS ROUTINE.
DAT1ST: MOV      (5)+,@#LINE ;GET LINE NUMBER
DAT1AA: MOV      #100,COUNT  ;GET CHARACTER COUNT
      MOV      #OUTBUF,%2    ;GET ADDRESS OF OUTPUT BUFFER
15:   JSR     7,@#GTBIN      ;GET DATA
      MOVB    %1,(2)+       ;LOAD OUTPUT BUFFER WITH DATA
      DEC     COUNT         ;GOT ALL DATA?
      BNE    15
      MOV     #TUMTAB,%1    ;LOAD TUMBLE TABLE POINTER
      CLR    TUMTAB
      JSR    5,BMOVE        ;CLEAR
      TUMTAB ;TUMBLE
      TUMTAB+1 ;TABLE
      177 ;64 WORDS
      MOV     #INBUF,%2     ;SETUP INPUT BUFFER POINTER
      BIS    #BIT0,@CSR    ;SET THE GO BIT
      JSR    5,@#XMITD     ;TRANSMIT
      -100 ;100 CHARACTERS
006020 012537 016412 001604 DAT1ST: MOV      (5)+,@#LINE ;GET LINE NUMBER
006024 012737 000144 001604 DAT1AA: MOV      #100,COUNT  ;GET CHARACTER COUNT
006032 012702 016564          MOV      #OUTBUF,%2    ;GET ADDRESS OF OUTPUT BUFFER
006036 004737 005612 15:   JSR     7,@#GTBIN      ;GET DATA
006042 110122          MOVB    %1,(2)+       ;LOAD OUTPUT BUFFER WITH DATA
006044 005337 001604          DEC     COUNT         ;GOT ALL DATA?
006050 001372          BNE    15
006052 012701 001306          MOV     #TUMTAB,%1    ;LOAD TUMBLE TABLE POINTER
006056 005037 001306          CLR    TUMTAB
006062 004537 005732          JSR    5,BMOVE        ;CLEAR
006066 001306          TUMTAB ;TUMBLE
006070 001307          TUMTAB+1 ;TABLE
006072 000177          177 ;64 WORDS
006074 012702 016730          MOV     #INBUF,%2     ;SETUP INPUT BUFFER POINTER
006100 052777 000001 173142  BIS    #BIT0,@CSR    ;SET THE GO BIT
006106 004537 005754          JSR    5,@#XMITD     ;TRANSMIT
006112 177634          -100 ;100 CHARACTERS
006114 032777 160000 173126 25:  BIT    #BIT15+BIT14+BIT13,@CSR ;TEST IF READY OR ANY ERROR
006122 001004          BNE    35            ;FLAGS ARE SET
006124 105777 173120          TSTB  @CSR          ;WAIT FOR THE RECEIVER
006130 100371          BPL   25            ;TO RECEIVE A CHARACTER
006132 000415          BR    55
006134 032777 060000 173106 35:  BIT    #BIT14+BIT13,@CSR ;TEST FOR ERROR FLAGS
006142 001403          BEQ   45            ;BRANCH NO ERROR
006144 104001          ERROR
  
```

```

006146 000137 006466      JMP      155      ;GO EXIT
006152 042777 100000 173070 45:  BIC      #BIT15,@CSR ;CLEAR TRANSMITTER READY FLAG
006160 105777 173064      TSTB     @CSR      ;TEST FOR CHARACTER READY
006164 100375      BPL      -4
006166 042777 000200 173054 55:  BIC      #BIT7,@CSR ;CLEAR CHAR DONE BIT
006174 005711      TST      (1)
006176 100401      BMI      +4      ;TEST FOR VALID ENTRY
006200 104001      ERROR
006202 111122      MOVB     (1),(2)+ ;MOVE CHAR FROM TUM. TAB. TO INPUT BUFFER
  
```

:ROUTINE TO STORE RECEIVED PARITY BIT IN PARITY BIT BUFFER

```

006204 012705 000001      MOV      #1,%5    ;GET ROTATE COUNT
006210 000261      SEC
006212 032711 020000      BIT      #BIT13,(1) ;TEST RECEIVED PARITY BIT
006216 001001      BNE      65      ;BRANCH IF RECEIVED PARITY WAS ODD
006220 000241      CLC
006222 004537 006472 65:  JSR      5,RORPARBUF ;ROTATE RECEIVED PARITY INTO PARITY BUFFER
  
```

:ROUTINE TO TEST THAT ENTRY IS FOR THE CORRECT LINE.

```

006226 011137 001270      MOV      (1),@#TTDAT ;GET TABLE ENTRY
006232 042737 160777 001270 BIC      #160777,TTDAT ;CLEAR ALL BUT LINE NUMBER
006240 123737 016412 001271 CMPB     LINE,TTDAT+1 ;COMPARE LINE NUMBERS
006246 001410      BEQ      75
006250 013737 016412 001276 MOV      LINE,XMTDAT ;GET CORRECT LINE # (X2)
006256 013737 001270 001274 MOV      TTDAT,RCVDAT ;GET LINE # (X2) THAT FALSE DATA CAME IN ON
006264 104011      ERROR1          ;ERROR! DATA CAME IN ON A LINE THAT
  
```

:PROGRAM WAS NOT TRANSMITTING ON.

```

006266 000477      BR       155      ;EXIT TEST
006270 020127 001504 75:  CMP      %1,#TUMTAB+176 ;IS POINTER AT THE END
006274 001002      BNE      85      ;OF THE TABLE
006276 012701 001304      MOV      #TUMTAB-2,%1
006302 005721 85:  TST      (1)+      ;INCREMENT POINTER
006304 010046      MOV      %0,-(6)  ;SAVE REGISTER ZERO
006306 013700 016412      MOV      LINE,%0  ;FETCH LINE NUMBER
006312 005760 001146      TST      WCT(0)   ;HAS THE LAST CHARACTER BEEN TRANSMITTED
006316 001402      BEQ      +6      ;LAST CHARACTER HAS BEEN TRANSMITTED
006320 012600      MOV      (6)+,%0 ;RESTORE REGISTER ZERO
006322 000674      BR       25
006324 012600      MOV      (6)+,%0 ;RESTORE REGISTER ZERO
006326 012701 016564 95:  MOV      #OUTBUF,%1
006332 012702 016730      MOV      #INBUF,%2
006336 012705 000014      MOV      #12,%5   ;ROTATE PARITY BUFFER
006342 004537 006472 105: JSR      5,RORPARBUF ;12 PLACES RIGHT
006346 005037 001274      CLR      RCVDAT
006352 005037 001276      CLR      XMTDAT
006356 020127 016727      CMP      %1,#OUTBUF+99 ;HAVE ALL CHARS. BEEN COMPARED
006362 001441      BEQ      155
006364 112137 001276      MOVB     (1)+,XMTDAT ;GET TRANSMITTED CHARACTER
006370 043737 001300 001276 BIC      CARMASK,XMTDAT ;CLEAR NON-TRANSMITTED BITS
006376 111237 001274      MOVB     (2),RCVDAT ;GET RECEIVED CHARACTER
006402 104002      DATCHK          ;COMPARE TRANS. & RCVD. CHARS.
  
```

:ROUTINE TO COMPUTE AND CHECK PARITY ON RECEIVED DATA

```

006404 012703 000010 115: MOV      #8,%3      ;GET BIT COUNTER
006410 005000      CLP      %0      ;CLEAR COMPUTED PARITY INDICATOR
006412 106037 001274 125: POPB     RCVDAT   ;LOOK AT RECEIVED BIT
  
```

```

006416 103001          BCC 135          ; BRANCH IF A 0
006420 005100          COM %0          ; COMPLEMENT RO IF A 1
006422 005303    135: DEC %3          ; DECREMENT BIT COUNTER
006424 001372          BNE 125          ; LOOK AT NEXT BIT IF NOT DONE
006426 000240          NOP          ; IF COMPUTED PARITY WAS ODD RO WILL
          ; CONTAIN ALL 1'S, IF EVEN RO = 0
006430 112237 001274  MOVB (2)+,RCVDAT ; GET RECEIVED CHARACTER
006434 012705 000001  MOV #1,%5      ; ROTATE PARITY BUFFER 1 PLACE
006440 004537 006472  JSR 5,RORPARBUF ; RIGHT LEAVING RECEIVED PARITY BIT IN CARRY
006444 103004          BCC 145          ; BRANCH IF RECEIVED PARITY WAS EVEN
006446 005700          TST %0          ; TEST FOR COMPUTED ODD PARITY
006450 001336          BNE 105          ; BRANCH IF COMPUTED & RECEIVED WAS ODD
006452 104001          ERROR          ; ERROR! COMPUTED =EVEN, RECEIVED = ODD
006454 000734          BR 105           ; CONTINUE TEST
006456 005700    145: TST %0          ; TEST FOR EVEN COMPUTED PARITY
006460 001732          BEQ 105           ; BRANCH IF COMPUTED PARITY WAS EVEN
006462 104001          ERROR          ; ERROR! COMPUTED =ODD, RECEIVED = EVEN
006464 000730          BR 105           ; CONTINUE TEST
006466 005726    155: POPSP          ; REPOSITION STACK POINTER
006470 104006          SCOPE          ; SCOPE
  
```

; ROUTINE TO ROTATE PARITY BUFFER.

```

006472 006037 006534  RORPARBUF: ROR PAR0
006476 006037 006536  ROR PAR1
006502 006037 006540  ROR PAR2
006506 006037 006542  ROR PAR3
006512 006037 006544  ROR PAR4
006516 006037 006546  ROR PAR5
006522 006037 006550  ROR PAR6
006526 005316          DEC (SP)          ; DECREMENT ROTATE COUNT
006530 001360          BNE RORPARBUF
006532 000205          RTS 5
  
```

; PARITY BUFFER

```

006534 000000  PAR0: OPEN
006536 000000  PAR1: OPEN
006540 000000  PAR2: OPEN
006542 000000  PAR3: OPEN
006544 000000  PAR4: OPEN
006546 000000  PAR5: OPEN
006550 000000  PAR6: OPEN
  
```

; SUBROUTINE TO FORM LINE BIT POSITION WITH THE LINE # IN LINE

```

006552 010046          GTLINB: MOV %0,-(SP) ; SAVE RO ON THE STACK
006554 005037 001272  CLP @#LINBIT ; CLEAR LINE BIT
006560 013700 016412  MOV @#LINE,%0 ; GET LINE
006564 000261          SEC          ; SET CARRY
006566 006137 001272  15: POL LINBIT ; SHIFT LINE BIT
006572 162700 000002  SUB #2,%0 ; SUBTRACT 2 FROM LINE NUMBER
006576 100373          BPL 15          ; BRANCH IF GREATER THAN 0
006600 012600          MOV (SP)+,%0 ; RESTORE RO
006602 000207          RTS 7          ; EXIT SUBROUTINE
  
```

```

006604 104000          PRGO:  TYPE
006606 017527          PRGOM
006610 012737 006644 001506  MOV    #RTO,KSTART ;GET ADDRESS OF FIRST TEST
006616 005037 001512          CLR    RTNNO      ;CLEAR ROUTINE #
006622 000137 002254          JMP    SRSET
006626 012737 006644 001506  PRGOR: MOV    #RTO,KSTART ;GET ADDRESS OF FIRST TEST
006634 005037 001512          CLR    RTNNO      ;CLEAR ROUTINE NUMBER
006640 000137 002306          JMP    GETRDY     ;GO AND START PROGRAM
;*****
006644 000000          RTO:   0           ;ROUTINE # 0 *
006646 006706          RT1    ;ADDR OF NEXT ROUTINE. *
006650 000144          100    ;ITERATION COUNT *
006652 006654          RTOA   ;SCOPE ENTRY POINT. *
000000          X=X+1
;*****
;TEST ABILITY TO REFERENCE CSR WITHOUT TRAPPING
006654 012737 006676 000004  RTOA:  MOV    #15,@#ERRVEC ;SET UP ERROR TRAP.
006662 005777 172362          TST    @CSR          ;REFERENCE CSR
006666 012737 000006 000004  MOV    #ERRVEC+2,@#ERRVEC ;RESET TIME OUT TRAP
006674 104006          SCOPE
006676 162716 000004  15:    SUB    #4,(6)      ;RESTORE PC TO WHERE THE ILLEGAL
;REFERENCE OCCURED
;ERROR! ILLEGAL REFERENCE OCCURED
006702 104001          ERROR
006704 000002          RT1    ;LOOP ILLEGAL REFERENCE INSTRUCTION
;*****
006706 000001          RT1:   1           ;ROUTINE # 1 *
006710 006760          RT2    ;ADDR OF NEXT ROUTINE. *
006712 000144          100    ;ITERATION COUNT *
006714 006716          RT1A   ;SCOPE ENTRY POINT. *
000001          X=X+1
;*****
;TEST THAT CSR BIT0 CAN BE SET AND CLEARED
006716 012777 000001 172324  RT1A:  MOV    #BIT0,@CSR   ;SET BIT0.
006724 022777 000001 172316  CMP    #BIT0,@CSR   ;TEST THAT BIT0 IS SET
006732 001402          BEQ    15           ;BRANCH IF SET
006734 104001          ERROR          ;CSR BIT0 FAILED TO SET
006736 000407          BR     25          ;OR AN ADDITIONAL BIT ALSO SET
006740 042777 000001 172302  15:    BIC    #BIT0,@CSR   ;CLEAR BIT0
006746 005777 172276          TST    @CSR          ;TEST THAT BIT0 IS CLEAR
006752 001401          BEQ    25          ;CSR BIT0 FAILED TO CLEAR
006754 104001          ERROR
006756 104006          25:    SCOPE
;*****
006760 000002          RT2:   2           ;ROUTINE # 2 *
006762 007032          RT3    ;ADDR OF NEXT ROUTINE. *
006764 000144          100    ;ITERATION COUNT *
006766 006770          RT2A   ;SCOPE ENTRY POINT. *
000002          X=X+1
;*****
;TEST THAT CSR BIT1 CAN BE SET AND CLEARED
006770 012777 000002 172252  RT2A:  MOV    #BIT1,@CSR   ;SET BIT1.
006776 022777 000002 172244  CMP    #BIT1,@CSR   ;TEST THAT BIT1 IS SET

```

```

007004 001402          BEQ      15          ; BRANCH IF SET
007006 104001          ERROR                     ; CSR BIT1 FAILED TO SET
007010 000407          BR       25          ; OR AN ADDITIONAL BIT ALSO SET
007012 042777 000002 172230 15:  BIC     #BIT1,@CSR ; CLEAR BIT1
007020 005777 172224          TST     @CSR      ; TEST THAT BIT1 IS CLEAR
007024 001401          BEQ      25          ; CSR BIT1 FAILED TO CLEAR
007026 104001          ERROR                     ; CSR BIT1 FAILED TO CLEAR
007030 104006          25:      SCOPE
; *****
007032 000003          RT3:     3          ; ROUTINE # 3 *
007034 007104          RT4     ; ADDR OF NEXT ROUTINE. *
007036 000144          100.     ; ITERATION COUNT *
007040 007042          RT3A    ; SCOPE ENTRY POINT. *
000003          X=X+1
; *****
; TEST THAT CSR BIT2 CAN BE SET AND CLEARED
007042 012777 000004 172200 RT3A:  MOV    #BIT2,@CSR ; SET BIT2.
007050 022777 000004 172172          CMP    #BIT2,@CSR ; TEST THAT BIT2 IS SET
007056 001402          BEQ     15          ; BRANCH IF SET
007060 104001          ERROR                     ; CSR BIT2 FAILED TO SET
007062 000407          BR      25          ; OR AN ADDITIONAL BIT ALSO SET
007064 042777 000004 172156 15:  BIC     #BIT2,@CSR ; CLEAR BIT2
007072 005777 172152          TST     @CSR      ; TEST THAT BIT2 IS CLEAR
007076 001401          BEQ     25          ; CSR BIT2 FAILED TO CLEAR
007100 104001          ERROR                     ; CSR BIT2 FAILED TO CLEAR
007102 104006          25:      SCOPE
; *****
007104 000004          RT4:     4          ; ROUTINE # 4 *
007106 007156          RT5     ; ADDR OF NEXT ROUTINE. *
007110 000144          100.     ; ITERATION COUNT *
007112 007114          RT4A    ; SCOPE ENTRY POINT. *
000004          X=X+1
; *****
; TEST THAT CSR BIT4 CAN BE SET AND CLEARED
007114 012777 000020 172126 RT4A:  MOV    #BIT4,@CSR ; SET BIT4.
007122 022777 000020 172120          CMP    #BIT4,@CSR ; TEST THAT BIT4 IS SET
007130 001402          BEQ     15          ; BRANCH IF SET
007132 104001          ERROR                     ; CSR BIT4 FAILED TO SET
007134 000407          BR      25          ; OR AN ADDITIONAL BIT ALSO SET
007136 042777 000020 172104 15:  BIC     #BIT4,@CSR ; CLEAR BIT4
007144 005777 172100          TST     @CSR      ; TEST THAT BIT4 IS CLEAR
007150 001401          BEQ     25          ; CSR BIT4 FAILED TO CLEAR
007152 104001          ERROR                     ; CSR BIT4 FAILED TO CLEAR
007154 104006          25:      SCOPE
; *****
007156 000005          RT5:     5          ; ROUTINE # 5 *
007160 007230          RT6     ; ADDR OF NEXT ROUTINE. *
007162 000144          100.     ; ITERATION COUNT *
007164 007166          RT5A    ; SCOPE ENTRY POINT. *
000005          X=X+1
; *****
; TEST THAT CSR BITS CAN BE SET AND CLEARED
007166 012777 000040 172054 RT5A:  MOV    #BIT5,@CSR ; SET BITS.
    
```

```
007174 022777 000040 172046      CMP      #BITS,@CSR      ;TEST THAT BITS IS SET
007202 001402                      BEQ      15              ;BRANCH IF SET
007204 104001                      ERROR                     ;CSR BITS FAILED TO SET
007206 000407                      BR       25              ;OR AN ADDITIONAL BIT ALSO SET
007210 042777 000040 172032 15:  BIC      #BITS,@CSR      ;CLEAR BITS
007216 005777 172026                TST      @CSR            ;TEST THAT BITS IS CLEAR
007222 001401                      BEQ      25              ;CSR BITS FAILED TO CLEAR
007224 104001                      ERROR                     ;CSR BITS FAILED TO CLEAR
007226 104006      25:  SCOPE
;*****
007230 000006      RT6:  6              ;ROUTINE # 6 *
007232 007302                      RT7              ;ADDR OF NEXT ROUTINE. *
007234 000144                      100.             ;ITERATION COUNT *
007236 007240      RT6A          ;SCOPE ENTRY POINT. *
000006      X=X+1
;*****
;TEST THAT CSR BIT6 CAN BE SET AND CLEARED
007240 012777 000100 172002  RT6A:  MOV      #BIT6,@CSR      ;SET BIT6.
007246 022777 000100 171774      CMP      #BIT6,@CSR      ;TEST THAT BIT6 IS SET
007254 001402                      BEQ      15              ;BRANCH IF SET
007256 104001                      ERROR                     ;CSR BIT6 FAILED TO SET
007260 000407                      BR       25              ;OR AN ADDITIONAL BIT ALSO SET
007262 042777 000100 171760 15:  BIC      #BIT6,@CSR      ;CLEAR BIT6
007270 005777 171754                TST      @CSR            ;TEST THAT BIT6 IS CLEAR
007274 001401                      BEQ      25              ;CSR BIT6 FAILED TO CLEAR
007276 104001                      ERROR                     ;CSR BIT6 FAILED TO CLEAR
007300 104006      25:  SCOPE
;*****
007302 000007      RT7:  7              ;ROUTINE # 7 *
007304 007354                      RT10             ;ADDR OF NEXT ROUTINE. *
007306 000144                      100.             ;ITERATION COUNT *
007310 007312      RT7A          ;SCOPE ENTRY POINT. *
000007      X=X+1
;*****
;TEST THAT CSR BIT12 CAN BE SET AND CLEARED
007312 012777 010000 171730  RT7A:  MOV      #BIT12,@CSR   ;SET BIT12.
007320 022777 010000 171722      CMP      #BIT12,@CSR     ;TEST THAT BIT12 IS SET
007326 001402                      BEQ      15              ;BRANCH IF SET
007330 104001                      ERROR                     ;CSR BIT12 FAILED TO SET
007332 000407                      BR       25              ;OR AN ADDITIONAL BIT ALSO SET
007334 042777 010000 171706 15:  BIC      #BIT12,@CSR     ;CLEAR BIT12
007342 005777 171702                TST      @CSR            ;TEST THAT BIT12 IS CLEAR
007346 001401                      BEQ      25              ;CSR BIT12 FAILED TO CLEAR
007350 104001                      ERROR                     ;CSR BIT12 FAILED TO CLEAR
007352 104006      25:  SCOPE
;*****
007354 000010      RT10: 10             ;ROUTINE # 10 *
007356 007426                      RT11             ;ADDR OF NEXT ROUTINE. *
007360 000144                      100.             ;ITERATION COUNT *
007362 007364      RT10A          ;SCOPE ENTRY POINT. *
000010      X=X+1
;*****
;TEST THAT CSR BIT13 CAN BE SET AND CLEARED
```

```
007364 012777 020000 171656 RT10A: MOV #BIT13,@CSR ;SET BIT13.
007372 022777 020000 171650 CMP #BIT13,@CSR ;TEST THAT BIT13 IS SET
007400 001402 BEQ 15 ;BRANCH IF SET
007402 104001 ERROR ;CSR BIT13 FAILED TO SET
007404 000407 BR 25 ;OR AN ADDITIONAL BIT ALSO SET
007406 042777 020000 171634 15: BIC #BIT13,@CSR ;CLEAR BIT13
007414 005777 171630 TST @CSR ;TEST THAT BIT13 IS CLEAR
007420 001401 BEQ 25
007422 104001 ERROR ;CSR BIT13 FAILED TO CLEAR
007424 104006 25: SCOPE
;*****
007426 000011 RT11: 11 ;ROUTINE # 11 *
007430 007516 RT12 ;ADDR OF NEXT ROUTINE. *
007432 000144 100. ;ITERATION COUNT *
007434 007436 RT11A ;SCOPE ENTRY POINT. *
. 000011 X=X+1
;*****
;TEST THAT RESET & CLEAR INSTRUCTION CLEAR ALL R/W BITS IN THE CONTROL
; STATUS REG. (CSR)
007436 012777 030167 171604 RT11A: MOV #30167,@CSR ;SET ALL R/W BITS IN THE CSR
007444 005037 001276 CLR XMTDAT
007450 104005 SRESET ;ISSUE RESET
007452 017737 171572 001274 MOV @CSR,RCVDAT ;GET CSR CONTENTS
007460 001402 BEQ 15 ;BRANCH IF RESET CLEARED ALL BITS
007462 104011 ERROR1 ;ERROR! RESET DID NOT CLEAR ALL BITS
007464 000764 BR RT11A ;LOOP ON ERROR
007466 012777 030167 171554 15: MOV #30167,@CSR ;SET ALL R/W BITS IN CSR
007474 005077 171550 CLR @CSR ;CLEAR THE CSR
007500 017737 171544 001274 MOV @CSR,RCVDAT ;GET & TEST CSR
007506 001402 BEQ 25 ;GO TO EXIT IF RESULT = 0
007510 104011 ERROR1 ;ERROR! CLEAR INST. DID NOT CLEAR ALL BITS
007512 000765 BR 15 ;LOOP ERROR
007514 104006 25: SCOPE ;SCOPE
;*****
007516 000012 RT12: 12 ;ROUTINE # 12 *
007520 007652 RT13 ;ADDR OF NEXT ROUTINE. *
007522 000012 10. ;ITERATION COUNT *
007524 007526 RT12A ;SCOPE ENTRY POINT. *
. 000012 X=X+1
;*****
;TEST THAT A BINARY COUNT CAN BE LOADED INTO A CLEAR BKCSR AND THAT
;A BINARY COUNT CAN BE CLEARED.
007526 005037 001276 RT12A: CLR XMTDAT
007532 013777 001276 171514 15: MOV XMTDAT,@BKCSR ;LOAD BINARY COUNT INTO BKCSR
007540 017737 171510 001274 MOV @BKCSR,RCVDAT ;GET BKCSR DATA
007546 023737 001276 001274 CMP XMTDAT,RCVDAT ;COMPARE DATA LOADED & DATA READ BACK
007554 001405 BEQ 25 ;BRANCH IF DATA COMPARES
007556 104011 ERROR1 ;ERROR! DATA DID NOT COMPARE
007560 032777 040000 171314 BIT #BIT14,@SWR ;SCOPE LOOP?
007566 001361 BNE 15 ;BRANCH IF SCOPE LOOP
007570 013701 001276 25: MOV XMTDAT,%1 ;SAVE BINARY COUNT
007574 005037 001276 CLR XMTDAT
007600 005077 171450 35: CLR @BKCSR ;CLEAR BKCSR AND TEST
007604 017737 171444 001274 MOV @BKCSR,RCVDAT ;BKCSR CAN BE CLEARED
```

```
007612 001405 BEQ 45 ; BRANCH IF BKCSR CLEARED
007614 104011 ERROR1 ; ERROR! BKCSR DID NOT CLEAR
007616 032777 040000 171256 BIT #BIT14,@SWR ; SCOPE LOOP?
007624 001365 BNE 35 ; BRANCH IF SCOPE LOOP
007626 010137 001276 45: MOV %1,XMTDAT ; GET BINARY COUNT
007632 023727 001276 177777 CMP XMTDAT,#-1 ; ALL NUMBERS BEEN LOADED
007640 001403 BEQ 55 ; GO TO EXIT
007642 005237 001276 INC XMTDAT ; INCREMENT BINARY COUNT
007646 000731 BR 15 ; REPEAT TEST
007650 104006 55: SCOPE ; SCOPE
; *****
007652 000013 RT13: 13 ; ROUTINE # 13 *
007654 007764 RT14 ; ADDR OF NEXT ROUTINE. *
007656 000144 100. ; ITERATION COUNT *
007660 007662 RT13A ; SCOPE ENTRY POINT. *
; *****
; TEST THAT RANDOM NUMBERS CAN BE LOADED INTO THE BKCSR
007662 012702 010000 RT13A: MOV #10000,%2 ; GET RANDOM #COUNTER
007666 017737 171362 001612 15: MOV @BKCSR,PRVCNT ; GET PREVIOUS CONTENTS
007674 004737 002756 JSR 7,@#RNGEN ; GO GET A RANDOM NUMBER
007700 010037 001276 MOV %0,XMTDAT ; GET RANDOM NUMBER
007704 013777 001276 171342 25: MOV XMTDAT,@BKCSR ; LOAD RANDOM NUMBER INTO BKCSR
007712 017737 171336 001274 MOV @BKCSR,RCVDAT ; GET BKCSR DATA
007720 023737 001276 001274 CMP XMTDAT,RCVDAT ; COMPARE DATA
007726 001401 BEQ 35 ; BRANCH IF SAME
007730 104011 ERROR1 ; ERROR! DATA NOT THE SAME
007732 032777 040000 171142 35: BIT #BIT14,@SWR ; SCOPE LOOP?
007740 001406 BEQ 45 ; BRANCH IF NO LOOP ON ERROR
007742 005077 171306 CLR @BKCSR ;
007746 013777 001612 171300 MOV PRVCNT,@BKCSR ; LOAD PREVIOUS CONTENTS
007754 000753 BR 25 ; REPEAT TEST
007756 005302 45: DEC %2 ;
007760 001342 BNE 15 ; BRANCH IF NOT
007762 104006 55: SCOPE ; SCOPE
; *****
007764 000014 RT14: 14 ; ROUTINE # 14 *
007766 010024 RT15 ; ADDR OF NEXT ROUTINE. *
007770 000012 10. ; ITERATION COUNT *
007772 007774 RT14A ; SCOPE ENTRY POINT. *
; *****
; TEST THAT RESET CLEARS ALL BREAK STATUS REGISTER BITS
007774 012777 177777 171252 RT14A: MOV #-1,@BKCSR
010002 005037 001276 CLR XMTDAT
010006 104005 SRESET
010010 017737 171240 001274 MOV @BKCSR,RCVDAT
010016 001401 BEQ 15
010020 104011 ERROR1
010022 104006 15: SCOPE
; *****
010024 000015 RT15: 15 ; ROUTINE # 15 *
010026 010160 RT16 ; ADDR OF NEXT ROUTINE *
```



010030 000012 10. ; ITERATION COUNT \*  
010032 010034 RT15A ; SCOPE ENTRY POINT. \*  
000015 X=X+1

\*\*\*\*\*

; TEST THAT A BINARY COUNT CAN BE LOADED INTO A CLEAR BASREG AND THAT  
; A BINARY COUNT CAN BE CLEARED.

010034 005037 001276 RT15A: CLR XMTDAT  
010040 013777 001276 171210 1\$: MOV XMTDAT, @BASREG ; LOAD BINARY COUNT INTO BASREG  
010046 017737 171204 001274 MOV @BASREG, RCVDAT ; GET BASREG DATA  
010054 023737 001276 001274 CMP XMTDAT, RCVDAT ; COMPARE DATA  
010062 001405 BEQ 2\$ ; BRANCH IF DATA COMPARES  
010064 104011 ERROR1 ; ERROR! DATA DID NOT COMPARE  
010066 032777 040000 171006 BIT #BIT14, @SWR ; SCOPE LOOP?  
010074 001361 BNE 1\$ ; BRANCH IF SCOPE LOOP  
010076 013701 001276 2\$: MOV XMTDAT, %1 ; SAVE BINARY COUNT  
010102 005037 001276 CLR XMTDAT  
010106 005077 171144 3\$: CLR @BASREG  
010112 017737 171140 001274 MOV @BASREG, RCVDAT  
010120 001405 BEQ 4\$ ; BRANCH IF BKCSR CLEARED  
010122 104011 ERROR1 ; ERROR! BKCSR DID NOT CLEAR  
010124 032777 040000 170750 BIT #BIT14, @SWR ; SCOPE LOOP?  
010132 001365 BNE 3\$ ; BRANCH IF SCOPE LOOP  
010134 010137 001276 4\$: MOV %1, XMTDAT ; GET BINARY COUNT  
010140 023727 001276 177000 CMP XMTDAT, #177000 ; ALL NUMBERS BEEN LOADED  
010146 001403 BEQ 5\$ ; GO TO EXIT  
010150 105237 001277 INCB XMTDAT+1 ; INCREMENT BINARY COUNT  
010154 000731 BR 1\$ ; REPEAT TEST  
010156 104006 5\$: SCOPE ; SCOPE

\*\*\*\*\*

010160 000016 RT16: 16 ; ROUTINE # 16 \*  
010162 010276 RT17 ; ADDR OF NEXT ROUTINE. \*  
010164 000144 100. ; ITERATION COUNT \*  
010166 010170 RT16A ; SCOPE ENTRY POINT. \*  
000016 X=X+1

\*\*\*\*\*

; TEST THAT RANDOM NUMBERS CAN BE LOADED INTO THE BASE REGISTER

010170 012702 010000 RT16A: MOV #10000, %2 ; GET RANDOM #COUNTER  
010174 017737 171056 001612 1\$: MOV @BASREG, PRVCNT ; GET PREVIOUS CONTENTS  
010202 004737 002756 JSR 7, @#RNGEN ; GO GET A RANDOM NUMBER  
010206 042700 000377 BIC #000377, %0 ; CLEAR UNUSED BITS  
010212 010037 001276 MOV %0, XMTDAT ; GET RANDOM NUMBER  
010216 013777 001276 171032 2\$: MOV XMTDAT, @BASREG ; LOAD RANDOM NUMBER INTO BASREG  
010224 017737 171026 001274 MOV @BASREG, RCVDAT ; GET BASREG DATA  
010232 023737 001276 001274 CMP XMTDAT, RCVDAT ; COMPARE DATA  
010240 001401 BEQ 3\$ ; BRANCH IF SAME  
010242 104011 ERROR1 ; ERROR! DATA NOT THE SAME  
010244 032777 040000 170630 3\$: BIT #BIT14, @SWR ; SCOPE LOOP?  
010252 001406 BEQ 4\$ ; BRANCH IF NO LOOP ON ERROR  
010254 005077 170776 CLR @BASREG  
010260 013777 001612 170770 MOV PRVCNT, @BASREG ; LOAD PREVIOUS CONTENTS  
010266 000753 BR 2\$ ; REPEAT TEST  
010270 005302 4\$: DEC %2 ; 10000 NUMBERS BEEN TESTED  
010272 001340 BNE 1\$ ; BRANCH IF NOT  
010274 104006 5\$: SCOPE ; SCOPE

```
010276 000017  
010300 010370  
010302 000144  
010304 010306  
000017  
; *****  
RT17: 17 ;ROUTINE # 17 *  
RT20 ;ADDR OF NEXT ROUTINE. *  
100. ;ITERATION COUNT *  
RT17A ;SCOPE ENTRY POINT. *  
X=X+1  
; *****
```

```
010306 013701 001252  
010312 012777 001106 170736  
010320 012700 000001  
010324 050011  
010326 020011  
010330 001006  
010332 040011  
010334 005711  
010336 001011  
010340 006300  
010342 103370  
010344 104006  
010346 010037 001276  
010352 011137 001274  
010356 104011  
010360 000771  
010362 005037 001276  
010366 000771  
; TEST THAT ALL BAR BITS CAN BE INDIVIDUALLY SET AND CLEARED  
RT17A: MOV BAR,%1 ;GET BAR ADDRESS  
MOV #CAT,@BASREG ;INITIALIZE BASE REGISTER  
MOV #1,%0 ;GET BIT TESTER  
15: BIS %0,(1) ;SET BAR BIT  
CMP %0,(1) ;TEST THAT ONLY THE PROPER BAR BIT SET  
BNE 35 ;BRANCH IF ERROR  
BIC %0,(1) ;CLEAR BAR BIT  
TST (1) ;TEST THAT BAR BIT CLEARED  
BNE 55 ;BRANCH IF BAR BIT FAILED TO CLEAR  
ASL %0 ;SHIFT BIT TESTER  
BCC 15  
25: SCOPE ;SCOPE  
35: MOV %0,XMTDAT ;GET WHAT DATA WAS SUPPOSED TO BE  
45: MOV (1),RCVDAT ;GET WHAT DATA WAS  
ERROR1 ;ERROR! IMPROPER BIT OPERATION  
BR 25 ;GO TO SCOPE  
55: CLR XMTDAT ;GET WHAT DATA WAS SUPPOSED TO BE  
BR 45
```

```
010370 000020  
010372 010456  
010374 000012  
010376 010400  
000020  
; *****  
RT20: 20 ;ROUTINE # 20 *  
RT21 ;ADDR OF NEXT ROUTINE. *  
10. ;ITERATION COUNT *  
RT20A ;SCOPE ENTRY POINT. *  
X=X+1  
; *****
```

```
010400 005037 001276  
010404 052777 177777 170640  
010412 104005  
010414 017737 170632 001274  
010422 001402  
010424 104011  
010426 000412  
010430 052777 177777 170614  
010436 005077 170610  
010442 017737 170604 001274  
010450 001401  
010452 104011  
010454 104006  
; TEST THAT RESET CLEARS ALL BAR BITS  
RT20A: CLR XMTDAT  
BIS #-1,@BAR ;SET ALL BAR BITS  
SRESET ;RESET  
MOV @BAR,RCVDAT ;GET BAR DATA  
BEQ 15 ;BRANCH IF ALL 0'S  
ERROR1 ;ERROR! RESET DID NOT CLEAR ALL BAR BITS  
BR 25 ;GO TO EXIT  
15: BIS #-1,@BAR ;SET ALL BIT IN THE BAR  
CLR @BAR ;CLEAR ALL BITS IN THE BAR  
MOV @BAR,RCVDAT ;GET & TEST RESULT OF CLEAR OPERATION  
BEQ 25 ;EXIT IF ALL BITS CLEARED  
ERROR1 ;ERROR! ALL BITS DID NOT CLEAR  
25: SCOPE ;SCOPE
```

```
010456 000021  
010460 010564  
010462 000144  
010464 010466  
000021  
; *****  
RT21: 21 ;ROUTINE # 21 *  
RT22 ;ADDR OF NEXT ROUTINE. *  
100. ;ITERATION COUNT *  
RT21A ;SCOPE ENTRY POINT. *  
X=X+1  
; *****
```

```
; TEST THAT CSR RESPONDS PROPERLY TO BYTE COMMANDS
010466 012777 010100 170554 RT21A: MOV #10100,@CSR ;LOAD TEST NUMBER IN CSR
010474 105077 170550 CLRB @CSR ;CLEAR EVEN BYTE
010500 022777 010000 170542 CMP #10000,@CSR ;TEST THAT ONLY EVEN BYTE CLEARED
010506 001410 BEQ 1$
010510 012737 010100 001276 MOV #10100,XMTDAT ;LOAD CORRECT RESULT
010516 017737 170526 001274 MOV @CSR,RCV DAT ;GET ACTUAL RESULT
010524 104011 ERROR1 ;ERROR! EVEN BYTE INSTRUCTION FAILED
010526 000415 BR 2$ ;GO TO SCOPE
010530 012777 010100 170512 1$: MOV #10100,@CSR ;LOAD TEST NUMBER IN CSR
010536 105077 171052 CLRB @.CSR ;TEST THAT ONLY ODD BYTE CLEARED
010542 001407 BEQ 2$
010544 012737 000100 001276 MOV #00100,XMTDAT ;LOAD CORRECT RESULT
010552 017737 170472 001274 MOV @CSR,RCV DAT ;LOAD ACTUAL RESULT
010560 104011 ERROR1 ;ERROR! ODD BYTE INSTRUCTION FAILED
010562 104006 2$: SCOPE ;SCOPE
;*****
010564 000022 RT22: 22 ;ROUTINE # 22 *
010566 010672 RT23 ;ADDR OF NEXT ROUTINE. *
010570 000144 100. ;ITERATION COUNT *
010572 010574 RT22A ;SCOPE ENTRY POINT. *
000022 X=X+1
;*****

; TEST THAT BAR RESPONDS PROPERLY TO BYTE COMMANDS
010574 012777 010100 170450 RT22A: MOV #10100,@BAR ;LOAD TEST NUMBER IN BAR
010602 105077 170444 CLRB @BAR ;CLEAR EVEN BYTE
010606 022777 010000 170436 CMP #10000,@BAR ;TEST THAT ONLY EVEN BYTE CLEARED
010614 001410 BEQ 1$
010616 012737 010100 001276 MOV #10100,XMTDAT ;LOAD CORRECT RESULT
010624 017737 170422 001274 MOV @BAR,RCV DAT ;GET ACTUAL RESULT
010632 104011 ERROR1 ;ERROR! EVEN BYTE INSTRUCTION FAILED
010634 000415 BR 2$ ;GO TO SCOPE
010636 012777 010100 170406 1$: MOV #10100,@BAR ;LOAD TEST NUMBER IN BAR
010644 105077 170746 CLRB @ BAR ;TEST THAT ONLY ODD BYTE CLEARED
010650 001407 BEQ 2$
010652 012737 000100 001276 MOV #00100,XMTDAT ;LOAD CORRECT RESULT
010660 017737 170364 001274 MOV @CSR,RCV DAT ;LOAD ACTUAL RESULT
010666 104011 ERROR1 ;ERROP! ODD BYTE INSTRUCTION FAILED
010670 104006 2$: SCOPE ;SCOPE
;*****
010672 000023 RT23: 23 ;ROUTINE # 23 *
010674 011000 RT24 ;ADDR OF NEXT ROUTINE. *
010676 000144 100. ;ITERATION COUNT *
010700 010702 RT23A ;SCOPE ENTRY POINT. *
000023 X=X+1
;*****

; TEST THAT BKCSR RESPONDS PROPERLY TO BYTE COMMANDS
010702 012777 010100 170344 RT23A: MOV #10100,@BKCSR ;LOAD TEST NUMBER IN BKCSR
010710 105077 170340 CLRB @BKCSR ;CLEAR EVEN BYTE
010714 022777 010000 170332 CMP #10000,@BKCSR ;TEST THAT ONLY EVEN BYTE CLEARED
010722 001410 BEQ 1$
010724 012737 010100 001276 MOV #10100,XMTDAT ;LOAD CORRECT RESULT
010732 017737 170316 001274 MOV @BKCSR,RCV DAT ;GET ACTUAL RESULT
```

```
010740 104011          ERROR1          ;ERROR! EVEN BYTE INSTRUCTION FAILED
010742 000415          BR            2$          ;GO TO SCOPE
010744 012777 010100 170302 1$:  MOV          #10100,@BKCSR ;LOAD TEST NUMBER IN BKCSR
010752 105077 170642          CLRB         @BKCSR      ;TEST THAT ONLY ODD BYTE CLEARED
010756 001407          BEQ          2$          ;
010760 012737 000100 001276  MOV          #00100,XMTDAT ;LOAD CORRECT RESULT
010766 017737 170256 001274  MOV          @CSR,RCVDAT  ;LOAD ACTUAL RESULT
010774 104011          ERROR1          ;ERROR! ODD BYTE INSTRUCTION FAILED
010776 104006          2$:  SCOPE          ;SCOPE
;*****
RT24:  24              ;ROUTINE # 24 *
        RT25          ;ADDR OF NEXT ROUTINE. *
        100           ;ITERATION COUNT *
        RT24A        ;SCOPE ENTRY POINT. *
        X=X+1
;*****
;TEST THAT OVER RUN BIT (CSR BIT13) CAUSES AN INTERRUPT WHEN SET
011010 012777 011042 170246 RT24A:  MOV          #1$,@XMTINT ;LOAD TRANSMITTER INTERRUPT VECTOR
011016 012777 010000 170224  MOV          #BIT12,@CSR  ;SET TRANSMITTER IE BIT
011024 052777 020000 170216  BIS          #BIT13,@CSR  ;SET OVER RUN BIT
011032 005037 177776          CLR          @#PSW       ;ENABLE INTERRUPTS
011036 000240          NOP
011040 104001          ERROR          ;ERROR!OVERRUN FAILED TO CAUSE AN
;INTERRUPT,OR INTERRUPTED TO INCOR-
;RECT ADDRESS
011042 104006          1$:  SCOPE          ;SCOPE
;*****
RT25:  25              ;ROUTINE # 25 *
        RT26          ;ADDR OF NEXT ROUTINE. *
        100           ;ITERATION COUNT *
        RT25A        ;SCOPE ENTRY POINT. *
        X=X+1
;*****
;TEST THAT THE DM11 INTERRUPTS AT THE CORRECT LEVEL
011054 012737 000340 177776 RT25A:  MOV          #PRTY7,@#PSW
011062 012777 011112 170174  MOV          #1$,@XMTINT ;LOAD TRANSMITTER INTERRUPT VECTOR
011070 012777 030000 170152  MOV          #30000,@CSR  ;SET OVER RUN & IE BITS
011076 012737 000200 177776  MOV          #PRTY4,@#PSW ;ALLOW INTERRUPTS ON LEVEL 5 & ABOVE
011104 000240          NOP
011106 104001          ERROR          ;ERROR!DM11 FAILED TO INTERRUPT
011110 000417          BR            3$          ;GO TO EXIT
011112 022626          1$:  CMP          (6)+,(6)+ ;RESET STACK POINTER
011114 013737 001266 177776  MOV          XMTLVL,@#PSW ;LOAD DM11 INTERRUPT LEVEL
011122 012777 011146 170134  MOV          #2$,@XMTINT ;LOAD TRANSMITTER INTERRUPT VECTOR
011130 005077 170114          CLR          @CSR
011134 012777 030000 170106  MOV          #30000,@CSR
011142 000240          NOP
011144 000401          BR            3$          ;GO TO EXIT
011146 104001          2$:  ERROR          ;ERROR! DM11 INTERRUPTED ON HIGHER
;PRIORITY LEVEL THAN SET FOR
011150 104006          3$:  SCOPE
;*****
RT26:  26              ;ROUTINE # 26 *
        RT27          ;ADDRESS OF NEXT TEST *
```

```
011156 000144          100.          ; ITERATION COUNT          *
011160 011162          LTST0          ; SCOPE ENTRY POINT        *
          000026          X=X+1
; *****
; TRANSMITTER LINE TEST LINE 0
011162 004537 004330  LTST0: JSR          5, XMTTST      ; GO TEST TRANSMITTER LINE 0
011166 000000          LINE0
          000001          Y=Y+1
; *****
011170 000027          RT27: 27          ; ROUTINE # 27              *
011172 011206          RT30          ; ADDRESS OF NEXT TEST.    *
011174 000144          100.          ; ITERATION COUNT          *
011176 011200          LTST1          ; SCOPE ENTRY POINT        *
          000027          X=X+1
; *****
; TRANSMITTER LINE TEST LINE 1
011200 004537 004330  LTST1: JSR          5, XMTTST      ; GO TEST TRANSMITTER LINE 1
011204 000002          LINE1
          000002          Y=Y+1
; *****
011206 000030          RT30: 30          ; ROUTINE # 30              *
011210 011224          RT31          ; ADDRESS OF NEXT TEST.    *
011212 000144          100.          ; ITERATION COUNT          *
011214 011216          LTST2          ; SCOPE ENTRY POINT        *
          000030          X=X+1
; *****
; TRANSMITTER LINE TEST LINE 2
011216 004537 004330  LTST2: JSR          5, XMTTST      ; GO TEST TRANSMITTER LINE 2
011222 000004          LINE2
          000003          Y=Y+1
; *****
011224 000031          RT31: 31          ; ROUTINE # 31              *
011226 011242          RT32          ; ADDRESS OF NEXT TEST.    *
011230 000144          100.          ; ITERATION COUNT          *
011232 011234          LTST3          ; SCOPE ENTRY POINT        *
          000031          X=X+1
; *****
; TRANSMITTER LINE TEST LINE 3
011234 004537 004330  LTST3: JSR          5, XMTTST      ; GO TEST TRANSMITTER LINE 3
011240 000006          LINE3
          000004          Y=Y+1
; *****
011242 000032          RT32: 32          ; ROUTINE # 32              *
011244 011260          RT33          ; ADDRESS OF NEXT TEST.    *
011246 000144          100.          ; ITERATION COUNT          *
011250 011252          LTST4          ; SCOPE ENTRY POINT        *
          000032          X=X+1
; *****
; TRANSMITTER LINE TEST LINE 4
011252 004537 004330  LTST4: JSR          5, XMTTST      ; GO TEST TRANSMITTER LINE 4
011256 000010          LINE4
          000005          Y=Y+1
; *****
011260 000033          RT33: 33          ; ROUTINE # 33              *
011262 011276          RT34          ; ADDRESS OF NEXT TEST.    *
011264 000144          100.          ; ITERATION COUNT          *
```

```
011266 011270          LTST5          ;SCOPE ENTRY POINT          *
          000033          X=X+1
;*****
;TRANSMITTER LINE TEST LINE 5
011270 004537 004330  LTST5: JSR          5,XMTTST          ;GO TEST TRANSMITTER LINE 5
011274 000012          LINE5
          000006          Y=Y+1
;*****
011276 000034          RT34:          34          ;ROUTINE # 34          *
011300 011314          RT35          ;ADDRESS OF NEXT TEST.          *
011302 000144          100.          ;ITERATION COUNT          *
011304 011306          LTST6          ;SCOPE ENTRY POINT          *
          000034          X=X+1
;*****
;TRANSMITTER LINE TEST LINE 6
011306 004537 004330  LTST6: JSR          5,XMTTST          ;GO TEST TRANSMITTER LINE 6
011312 000014          LINE6
          000007          Y=Y+1
;*****
011314 000035          RT35:          35          ;ROUTINE # 35          *
011316 011332          RT36          ;ADDRESS OF NEXT TEST.          *
011320 000144          100.          ;ITERATION COUNT          *
011322 011324          LTST7          ;SCOPE ENTRY POINT          *
          000035          X=X+1
;*****
;TRANSMITTER LINE TEST LINE 7
011324 004537 004330  LTST7: JSR          5,XMTTST          ;GO TEST TRANSMITTER LINE 7
011330 000016          LINE7
          000010          Y=Y+1
;*****
011332 000036          RT36:          36          ;ROUTINE # 36          *
011334 011350          RT37          ;ADDRESS OF NEXT TEST.          *
011336 000144          100.          ;ITERATION COUNT          *
011340 011342          LTST10         ;SCOPE ENTRY POINT          *
          000036          X=X+1
;*****
;TRANSMITTER LINE TEST LINE 10
011342 004537 004330  LTST10: JSR         5,XMTTST          ;GO TEST TRANSMITTER LINE 10
011346 000020          LINE10
          000011          Y=Y+1
;*****
011350 000037          RT37:          37          ;ROUTINE # 37          *
011352 011366          RT40          ;ADDRESS OF NEXT TEST.          *
011354 000144          100.          ;ITERATION COUNT          *
011356 011360          LTST11         ;SCOPE ENTRY POINT          *
          000037          X=X+1
;*****
;TRANSMITTER LINE TEST LINE 11
011360 004537 004330  LTST11: JSR         5,XMTTST          ;GO TEST TRANSMITTER LINE 11
011364 000022          LINE11
          000012          Y=Y+1
;*****
011366 000040          RT40:          40          ;ROUTINE # 40          *
011370 011404          RT41          ;ADDRESS OF NEXT TEST.          *
011372 000144          100.          ;ITERATION COUNT          *
011374 011376          LTST12         ;SCOPE ENTRY POINT          *
```

```
000040 X=X+1
;*****
;TRANSMITTER LINE TEST LINE 12
011376 004537 004330 LTST12: JSR 5,XMTTST ;GO TEST TRANSMITTER LINE 12
011402 000024 LINE12
000013 Y=Y+1
;*****
011404 000041 RT41: 41 ;ROUTINE # 41 *
011406 011422 RT42 ;ADDRESS OF NEXT TEST. *
011410 000144 100. ;ITERATION COUNT *
011412 011414 LTST13 ;SCOPE ENTRY POINT *
000041 X=X+1
;*****
;TRANSMITTER LINE TEST LINE 13
011414 004537 004330 LTST13: JSR 5,XMTTST ;GO TEST TRANSMITTER LINE 13
011420 000026 LINE13
000014 Y=Y+1
;*****
011422 000042 RT42: 42 ;ROUTINE # 42 *
011424 011440 RT43 ;ADDRESS OF NEXT TEST. *
011426 000144 100. ;ITERATION COUNT *
011430 011432 LTST14 ;SCOPE ENTRY POINT *
000042 X=X+1
;*****
;TRANSMITTER LINE TEST LINE 14
011432 004537 004330 LTST14: JSR 5,XMTTST ;GO TEST TRANSMITTER LINE 14
011436 000030 LINE14
000015 Y=Y+1
;*****
011440 000043 RT43: 43 ;ROUTINE # 43 *
011442 011456 RT44 ;ADDRESS OF NEXT TEST. *
011444 000144 100. ;ITERATION COUNT *
011446 011450 LTST15 ;SCOPE ENTRY POINT *
000043 X=X+1
;*****
;TRANSMITTER LINE TEST LINE 15
011450 004537 004330 LTST15: JSR 5,XMTTST ;GO TEST TRANSMITTER LINE 15
011454 000032 LINE15
000016 Y=Y+1
;*****
011456 000044 RT44: 44 ;ROUTINE # 44 *
011460 011474 RT45 ;ADDRESS OF NEXT TEST. *
011462 000144 100. ;ITERATION COUNT *
011464 011466 LTST16 ;SCOPE ENTRY POINT *
000044 X=X+1
;*****
;TRANSMITTER LINE TEST LINE 16
011466 004537 004330 LTST16: JSR 5,XMTTST ;GO TEST TRANSMITTER LINE 16
011472 000034 LINE16
000017 Y=Y+1
;*****
011474 000045 RT45: 45 ;ROUTINE # 45 *
011476 011512 RT46 ;ADDRESS OF NEXT TEST. *
011500 000144 100. ;ITERATION COUNT *
011502 011504 LTST17 ;SCOPE ENTRY POINT *
000045 X=X+1
```

```
*****  
; TRANSMITTER LINE TEST LINE 17  
011504 004537 004330 LTST17: JSR 5,XMTTST ; GO TEST TRANSMITTER LINE 17  
011510 000036 LINE17  
000020 Y=Y+1  
000000 Y=0  
000000 A=0  
*****  
; ROUTINE # 46 *  
011512 000046 RT46: 46 ; ADDRESS OF NEXT TEST *  
011514 011530 RT47 ; ITERATION COUNT *  
011516 000144 100. ; SCOPE ENTRY POINT *  
011520 011522 RCV0  
000046 X=X+1  
*****  
; RECEIVER LINE TEST LINE 0  
011522 004537 004542 RCV0: JSR 5,RCVTST ; GO TEST RECEIVER LINE 0  
011526 000000 LINE0  
000001 Y=Y+1  
*****  
; ROUTINE # 47 *  
011530 000047 RT47: 47 ; ADDRESS OF NEXT TEST *  
011532 011546 RT50 ; ITERATION COUNT *  
011534 000144 100. ; SCOPE ENTRY POINT *  
011536 011540 RCV1  
000047 X=X+1  
*****  
; RECEIVER LINE TEST LINE 1  
011540 004537 004542 RCV1: JSR 5,RCVTST ; GO TEST RECEIVER LINE 1  
011544 000002 LINE1  
000002 Y=Y+1  
*****  
; ROUTINE # 50 *  
011546 000050 RT50: 50 ; ADDRESS OF NEXT TEST *  
011550 011564 RT51 ; ITERATION COUNT *  
011552 000144 100. ; SCOPE ENTRY POINT *  
011554 011556 RCV2  
000050 X=X+1  
*****  
; RECEIVER LINE TEST LINE 2  
011556 004537 004542 RCV2: JSR 5,RCVTST ; GO TEST RECEIVER LINE 2  
011562 000004 LINE2  
000003 Y=Y+1  
*****  
; ROUTINE # 51 *  
011564 000051 RT51: 51 ; ADDRESS OF NEXT TEST *  
011566 011602 RT52 ; ITERATION COUNT *  
011570 000144 100. ; SCOPE ENTRY POINT *  
011572 011574 RCV3  
000051 X=X+1  
*****  
; RECEIVER LINE TEST LINE 3  
011574 004537 004542 RCV3: JSR 5,RCVTST ; GO TEST RECEIVER LINE 3  
011600 000006 LINE3  
000004 Y=Y+1  
*****  
; ROUTINE # 52 *  
011602 000052 RT52: 52 ; ADDRESS OF NEXT TEST *  
011604 011620 RT53 ; ITERATION COUNT *  
011606 000144 100. ; SCOPE ENTRY POINT *  
011610 011612 RCV4
```



```
000052 X=X+1
;*****
;RECEIVER LINE TEST LINE 4
011612 004537 004542 RCV4: JSR 5,RCVTST ;GO TEST RECEIVER LINE 4
011616 000010 LINE4
000005 Y=Y+1
;*****
011620 000053 RT53: 53 ;ROUTINE # 53 *
011622 011636 RT54 ;ADDRESS OF NEXT TEST *
011624 000144 100. ;ITERATION COUNT *
011626 011630 RCV5 ;SCOPE ENTRY POINT *
000053 X=X+1
;*****
;RECEIVER LINE TEST LINE 5
011630 004537 004542 RCV5: JSR 5,RCVTST ;GO TEST RECEIVER LINE 5
011634 000012 LINE5
000006 Y=Y+1
;*****
011636 000054 RT54: 54 ;ROUTINE # 54 *
011640 011654 RT55 ;ADDRESS OF NEXT TEST *
011642 000144 100. ;ITERATION COUNT *
011644 011646 RCV6 ;SCOPE ENTRY POINT *
000054 X=X+1
;*****
;RECEIVER LINE TEST LINE 6
011646 004537 004542 RCV6: JSR 5,RCVTST ;GO TEST RECEIVER LINE 6
011652 000014 LINE6
000007 Y=Y+1
;*****
011654 000055 RT55: 55 ;ROUTINE # 55 *
011656 011672 RT56 ;ADDRESS OF NEXT TEST *
011660 000144 100. ;ITERATION COUNT *
011662 011664 RCV7 ;SCOPE ENTRY POINT *
000055 X=X+1
;*****
;RECEIVER LINE TEST LINE 7
011664 004537 004542 RCV7: JSR 5,RCVTST ;GO TEST RECEIVER LINE 7
011670 000016 LINE7
000010 Y=Y+1
;*****
011672 000056 RT56: 56 ;ROUTINE # 56 *
011674 011710 RT57 ;ADDRESS OF NEXT TEST *
011676 000144 100. ;ITERATION COUNT *
011700 011702 RCV10 ;SCOPE ENTRY POINT *
000056 X=X+1
;*****
;RECEIVER LINE TEST LINE 10
011702 004537 004542 RCV10: JSR 5,RCVTST ;GO TEST RECEIVER LINE 10
011706 000020 LINE10
000011 Y=Y+1
;*****
011710 000057 RT57: 57 ;ROUTINE # 57 *
011712 011726 RT60 ;ADDRESS OF NEXT TEST *
011714 000144 100. ;ITERATION COUNT *
011716 011720 RCV11 ;SCOPE ENTRY POINT *
000057 X=X+1
```

```
*****  
;RECEIVER LINE TEST LINE 11  
011720 004537 004542 RCV11: JSR 5,RCVTST ;GO TEST RECEIVER LINE 11  
011724 000022 LINE11  
000012 Y=Y+1  
*****  
RT60: 60 ;ROUTINE # 60 *  
011726 000060 RT61 ;ADDRESS OF NEXT TEST *  
011730 011744 100. ;ITERATION COUNT *  
011732 000144 RCV12 ;SCOPE ENTRY POINT *  
011734 011736 X=X+1  
000060  
*****  
;RECEIVER LINE TEST LINE 12  
011736 004537 004542 RCV12: JSR 5,RCVTST ;GO TEST RECEIVER LINE 12  
011742 000024 LINE12  
000013 Y=Y+1  
*****  
RT61: 61 ;ROUTINE # 61 *  
011744 000061 RT62 ;ADDRESS OF NEXT TEST *  
011746 011762 100. ;ITERATION COUNT *  
011750 000144 RCV13 ;SCOPE ENTRY POINT *  
011752 011754 X=X+1  
000061  
*****  
;RECEIVER LINE TEST LINE 13  
011754 004537 004542 RCV13: JSR 5,RCVTST ;GO TEST RECEIVER LINE 13  
011760 000026 LINE13  
000014 Y=Y+1  
*****  
RT62: 62 ;ROUTINE # 62 *  
011762 000062 RT63 ;ADDRESS OF NEXT TEST *  
011764 012000 100. ;ITERATION COUNT *  
011766 000144 RCV14 ;SCOPE ENTRY POINT *  
011770 011772 X=X+1  
000062  
*****  
;RECEIVER LINE TEST LINE 14  
011772 004537 004542 RCV14: JSR 5,RCVTST ;GO TEST RECEIVER LINE 14  
011776 000030 LINE14  
000015 Y=Y+1  
*****  
RT63: 63 ;ROUTINE # 63 *  
012000 000063 RT64 ;ADDRESS OF NEXT TEST *  
012002 012016 100. ;ITERATION COUNT *  
012004 000144 RCV15 ;SCOPE ENTRY POINT *  
012006 012010 X=X+1  
000063  
*****  
;RECEIVER LINE TEST LINE 15  
012010 004537 004542 RCV15: JSR 5,RCVTST ;GO TEST RECEIVER LINE 15  
012014 000032 LINE15  
000016 Y=Y+1  
*****  
RT64: 64 ;ROUTINE # 64 *  
012016 000064 RT65 ;ADDRESS OF NEXT TEST *  
012020 012034 100. ;ITERATION COUNT *  
012022 000144 RCV16 ;SCOPE ENTRY POINT *  
012024 012026 X=X+1  
000064  
*****
```

```

;RECEIVER LINE TEST LINE 16
012026 004537 004542 RCV16: JSR 5,RCVTST ;GO TEST RECEIVER LINE 16
012032 000034 LINE16
000017 Y=Y+1
;*****
012034 000065 RT65: 65 ;ROUTINE # 65 *
012036 012054 RT66 ;ADDRESS OF NEXT TEST *
012040 000144 100 ;ITERATION COUNT *
012042 012044 RCV17 ;SCOPE ENTRY POINT *
000065 X=X+1
;*****
;RECEIVER LINE TEST LINE 17
012044 004537 004542 RCV17: JSR 5,RCVTST ;GO TEST RECEIVER LINE 17
012050 000036 LINE17
000020 Y=Y+1
012052 000240 NOP
;*****
012054 000066 RT66: 66 ;ROUTINE # 66 *
012056 012314 RT67 ;ADDR OF NEXT ROUTINE. *
012060 000012 10 ;ITERATION COUNT *
012062 012070 RT66A: ;SCOPE ENTRY POINT. *
000066 X=X+1
;*****

012064 000240 NOP
012066 000240 NOP

;TEST THAT NEX BIT (CSR BIT 14) SETS WHEN THE TRANSMITTER REFERENCES
;NON-EXISTANT MEMORY, THE CORRESPONDING BAR BIT CLEARS
;AND THAT AN INTERRUPT OCCURS. ALL LINES ARE USED FOR THE TEST.
012070 004737 003724 RT66A: JSR 7,TIMER ;GO CALCULATE MACHINE TIME TO TRANSMIT
;ONE CHARACTER
012074 012701 001106 MOV #CAT,%1 ;GET CAT ADDRESS
012100 012702 160000 MOV #160000,%2 ;GET A NON-EXISTANT ADDRESS
012104 012703 000020 MOV #16,%3 ;GET COUNTER
012110 010221 15: MOV %2,(1)+ ;LOAD THE CURRENT ADDRESS
012112 005303 DEC %3 ;TABLE WITH NON-EXISTANT
012114 001375 BNE 15 ;ADDRESSES
012116 012701 000001 MOV #LBIT0,%1 ;GET LINE BIT
012122 012777 012260 167134 MOV #65,@XMTINT ;LOAD TRANSMITTER INT. VECTOR
012130 052777 000060 167112 25: BIS #60,@CSR ;SET EXTENDED ADDRESS BITS
012136 050177 167110 BIS %1,@BAR ;START TRANSMITTER
012142 013737 004076 012152 MOV TIME14,%5 ;LOAD DELAY TIME TO
012150 104400 DELAY ;DELAY FOR 1/4TH OF A CHARACTER
012152 000000 35: OPEN ;TO RESPOND TO NEX
012154 017737 167072 001274 MOV @BAR,RCV DAT ;GET BAR DATA & TEST
012162 001406 BEQ 45 ;THAT IT IS CLEAR
012164 005037 001276 CLR XMT DAT
012170 104011 ERROR1 ;ERROR!BAR BIT DID NOT CLEAR
012172 005077 167054 CLR @BAR
012176 000440 BR 75 ;GO TO SCOPE
012200 032777 040000 167042 45: BIT #BIT14,@CSR ;TEST THAT NEX BIT IS SET
012206 001002 BNE 55 ;BRANCH IF SET
012210 104001 ERROR ;ERROR! NEX BIT FAILED TO SET
012212 000432 BR 75 ;GO TO SCOPE
012214 042777 100000 167026 55: BIC #BIT15,@CSR ;CLEAR TRANSMITTER READY FLAG
012222 052777 010000 167020 BIS #BIT12,@CSR ;SET TRANSMITTER IE BIT

```

```
012230 005037 177776 CLR @#PSW ;ALLOW INTERRUPTS
012234 000240 NOP
012236 012737 000340 177776 MOV #PRTY7,@#PSW ;LOCK OUT INTERRUPTS
012244 010137 001276 MOV %1,XMTDAT ;LOAD LINE THAT FAILED
012250 005037 001274 CLR RCVDAT
012254 104011 ERROR1 ;ERROR! NEX FAILED TO CAUSE INTERRUPT
;TYPEOUT SHOWS LINE # THAT FAILED
;GO TO SCOPE
012256 000410 BR 7%
012260 005077 166764 6%: CLR @CSR
012264 012737 000340 177776 MOV #PRTY7,@#PSW ;LOCK OUT INTERRUPTS
012272 022626 CMP (6)+,(6)+ ;ADJUST STACK PTR
012274 006301 ASL %1 ;SHIFT LINE BIT
012276 103314 BCC 2% ;DO NEXT LINE
012300 013737 004074 012310 7%: MOV TIME1,8% ;WAIT FOR TRANSMITTER TO RUN
012306 104400 DELAY ;TO COMPLETION BEFORE
012310 000000 8%: OPEN ;EXITING TEST
012312 104006 SCOPE ;SCOPE
;*****
012314 000067 RT67: 67 ;ROUTINE # 67 *
012316 012424 RT70 ;ADDR OF NEXT ROUTINE. *
012320 000012 100 ;ITERATION COUNT *
012322 012324 RT67A ;SCOPE ENTRY POINT. *
000067 X=X+1
;*****
;TEST THAT NEX BIT SETS IF THE DM11 TABLES ARE IN NON-EXISTANT CORE
012324 012777 160000 166724 RT67A: MOV #160000,@BASREG ;SET BASE REGISTER TO NON-EXISTANT ADRS.
012332 012737 012414 000004 MOV #4%,@#ERRVEC ;SET TIME OUT TRAP VECTOR
012340 005737 160000 TST @#160000 ;CHECK THAT ADDRESS TIMES OUT
012344 013737 004076 012362 MOV TIME14,1% ;GET TIME TO TRANSMIT 1/4 CHAR.
012352 052777 000001 166672 BIS #LBITO,@BAR ;START TO TRANSMIT ON LINE 0
012360 104400 DELAY ;DELAY 1/4TH OF A CHARACTER
012362 000000 1%: OPEN ;TIME
012364 005077 166662 CLR @BAR ;STOP TRANSMITTER
012370 022777 040060 166652 CMP #BIT14+60,@CSR ;TEST THAT ONLY NEX IS SET
012376 001401 BEQ 2%
012400 104001 ERROR ;ERROR! EITHER NEX FALED TO SET
;OR OTHER BITS SET
012402 013737 004074 012412 2%: MOV TIME1,3% ;DELAY 1 CHARACTER TIME TO ALLOW
012410 104400 DELAY ;TRANSMITTER TO RUN TO
012412 000000 3%: OPEN ;COMPLETION
012414 012737 000006 000004 4%: MOV #ERRVEC+2,@#ERRVEC ;RESTORE TIME OUT TRAP
012422 104006 SCOPE
;*****
012424 000070 RT70: 70 ;ROUTINE # 70 *
012426 012540 RT71 ;ADDR OF NEXT ROUTINE. *
012430 000144 100 ;ITERATION COUNT *
012432 012434 RT70A ;SCOPE ENTRY POINT. *
000070 X=X+1
;*****
;TEST THAT WHEN THE GO BIT IS CLEAR THAT THE RECEIVERS DO NOT RECEIVE
;DATA. EACH LINE IN TURN IS TRANSMITTED ON, AND WHEN TEN CHARACTERS
;HAVE BEEN TRANSMITTED THE RECEIVER DONE FLAG IS TESTED. IF IT IS SET
;AN ERROR IS INDICATED ON THE LINE DATA WAS RECEIVED ON.
012434 005037 016412 RT70A: CLR LINE ;SET UP TO TRANSMIT
```

```
012440 004537 005754 15: JSR 5, @XMITD ;10. CHARACTERS
012444 177766 -10. ;ON EACH LINE
012446 005777 166576 TST @CSR ;WAIT FOR 10. CHARACTERS
012452 100375 BPL -4 ;TO BE TRANSMITTED
012454 042777 100000 166566 BIC #100000, @CSR
012462 105777 166562 TSTB @CSR ;TEST RECEIVER DONE FLAG
012466 100010 BPL 25
012470 013737 001272 001274 MOV LINBIT, RCVDAT ;GET LINE BIT OF ACTIVE LINE
012476 013737 001272 001276 MOV LINBIT, XMTDAT ;THAT ERROR OCCURED ON
012504 104011 ERROR1 ;ERROR! DATA WAS RECEIVED ON LINE INDICATED
012506 000413 BR 45 ;GO TO SCOPE
012510 062737 000002 016412 25: ADD #2, LINE ;SET UP NEXT LINE NUMBER
012516 006337 001272 ASL LINBIT ;GET READY TO TRANSMIT ON NEXT LINE
012522 103346 BCC 15 ;GO TRANSMIT ON NEXT LINE
012524 013737 004074 012534 MOV TIME1, 35
012532 104400 DELAY ;DELAY 1 CHARACTER
012534 000000 35: 0 ;TIME BEFORE ENTERING NEXT TEST
012536 104006 45: SCOPE ;SCOPE
;*****
RT71: 71 ;ROUTINE # 71 *
RT72 ;ADDR OF NEXT ROUTINE. *
20. ;ITERATION COUNT *
RT71A ;SCOPE ENTRYPOINT. *
X=X+1
;*****
;TEST THAT CURRENT ADDRESS INCREMENTS PROPERLY WHEN A CHAR-
;ACTER IS TRANSMITTED. LINE 0 IS USED FOR THE TEST.
RT71A: CLR %0 ;R0=CURRENT ADRS AFTER TRANSMISSION
15: MOV %0, %1 ;R0=CURRENT ADDRESS BEFORE TRANSMISSION
INC %1 ;AND R1=CURRENT ADDRESS AFTER TRANSMISSION
MOV #35, @ERRVEC ;SET UP PROCESSOR
MOV #PRTY7, @ERRVEC+2 ;TIME OUT TRAP
MOV #CAT, @BASREG ;SET UP BASE REGISTER
MOV %0, CAT ;LOAD CURRENT ADDRESS TABLE (LINE 0)
TSTB (0) ;DOES MEMORY EXIST?
MOV #-2, WCT ;SET CHAR. COUNT TO TRANSMIT 1 CHAR.
MOV #5, @CSR ;SET MAINT & GO BITS
MOV #LBIT0, @BAR ;TRANSMIT ON LINE 0
TSTB @CSR ;WAIT FOR THE RECEIVER
BPL -4 ;TO RECEIVE FIRST CHARACTER
BIC #200, @CSR ;CLEAR RECEIVER DONE FLAG
TSTB @CSR ;WAIT FOR RECEIVER TO RECEIVE
BPL -4 ;THE SECOND CHARACTER
CMP CAT, %1 ;TEST THAT CURRENT ADRS
; INCREMENTED PROPERLY
012656 001413 BEQ 25
012660 010137 001276 MOV %1, XMTDAT ;GET COMPUTED RESULT
012664 013737 001106 001274 MOV CAT, RCVDAT ;GET ACTUAL RESULT
012672 104011 ERROR1 ;ERROR! CURRENT ADDRESS DID NOT
012674 032777 040000 166200 BIT #BIT14, @SWR ;INCREMENT PROPERLY
012702 001323 BNE 15 ;BRANCH IF SCOPE SWITCH IS SET
012704 000405 BR 35 ;GO TO EXIT
012706 005701 25: TST %1
012710 001403 BEQ 35
012712 000261 SEC
```

```

012714 006100          ROL    %0
012716 100715          BMI    1$
012720 012737 000006 000004 3$:  MOV    #ERRVEC+2,@#ERRVEC ;RESTORE TIME OUT TRAP
012726 005037 000006          CLR    @#ERRVEC+2
012732 104006          SCOPE          ;SCOPE
;*****
012734 000072          RT72:  72          ;ROUTINE # 72 *
012736 013300          RT73          ;ADDR OF NEXT ROUTINE. *
012740 000024          20.          ;ITERATION COUNT *
012742 012744          RT72A         ;SCOPE ENTRY POINT. *
000072          X=X+1
;*****

;TEST THAT DATA CAN BE TRANSMITTED FROM ALL AVAILABLE CORE.
;LINE 0 IS USED FOR THE TEST AND ONLY ONE WORD IS TRANSMITTED
;AT A TIME.
012744 005000          RT72A:  CLR    %0          ;CLEAR INDEX REGISTER
012746 000005          RESET
012750 016037 013234 001106 1$:  MOV    AREA(0),CAT ;LOAD CURRENT ADDRESS
012756 012777 000005 166264  MOV    #5,@CSR ;SET MAINT & GO BITS
012764 012737 177777 001146  MOV    #-1,WCT ;SET UP CHAR COUNT TO TRANSMIT 1 CHAR.
012772 012777 000001 166252  MOV    #LBITO,@BAR ;TRANSMIT CHAR ON LINE 0
013000 005777 166244          TST    @CSR ;WAIT FOR THE TRANSMITTER
013004 100375          BPL    -4 ;TO TRANSMIT THE CHARACTER
013006 105777 166236          TSTB   @CSP ;TEST FOR DONE
013012 100375          BPL    -4
013014 005077 166230          CLR    @CSR ;CLEAR ALL FLAGS
013020 005037 001274          CLR    RCVDAT
013024 113737 001306 001274  MOVB   TUMTAB,RCVDAT ;GET RECEIVED CHARACTER
013032 117037 013234 001276  MOVB   @AREA(0),XMTDAT ;GET TRANSMITTED CHARACTER
013040 043737 001300 001276  BIC    CARMSK,XMTDAT ;CLEAR NON-TRANSMITTED BITS
013046 123737 001274 001276  CMPB   RCVDAT,XMTDAT ;COMPARE CHARACTERS
013054 001402          BEQ    2$ ;BRANCH IF VALID COMPARISON
013056 104011          ERROR1 ;ERROR!DATA COMPARISON ERROR

; (CAT)-1 IS THE MEMORY LOCATION WHERE THE DATA WAS TRANSMITTED FROM
013060 000464          BR    6$ ;GO TO EXIT
013062 020027 000006          2$:  CMP    %0,#6 ;HAS FIRST 4K BEEN TESTED
013066 001402          BEQ    3$ ;BRANCH IF IT HAS
013070 005720          TST    (0)+ ;INCREMENT INDEX
013072 000726          BR    1$ ;GO REPEAT TEST

013074 012737 013222 000004 3$:  MOV    #5$,@#ERRVEC ;BEGIN TESTING ABOVE 4K
013074 012737 013222 000004  ;SET TIME OUT TRAP TO EXIT
;TEST IF MEMORY TIMES OUT
013102 005001          4$:  CLR    %1 ;SET UP DATA IDENTIFIER
013104 005201          INC    %1 ;INCREMENT DATA IDENTIFIER
013106 005720          TST    (0)+ ;INCREMENT INDEX
013110 110170 013234          MOVB   %1,@AREA(0) ;LOAD IDENTIFIER INTO MEMORY
013114 016037 013234 001106  MOV    AREA(0),CAT ;LOAD CURRENT ADDRESS
013122 012777 000005 166120  MOV    #5,@CSR ;SET MAINT & GO BITS
013130 012737 177777 001146  MOV    #-1,WCT ;SET UP CHAR COUNT TO TRANSMIT 1 CHAR.
013136 012777 000001 166106  MOV    #LBITO,@BAR ;TRANSMIT ON LINE 0
013144 005777 166100          TST    @CSR ;WAIT FOR THE TRANSMITTER TO
013150 100375          BPL    -4 ;TRANSMIT THE CHARACTER
013152 105777 166072          TSTB   @CSP ;TEST FOR CHARACTER DONE

```

```
013156 100375          BPL      -4
013160 005077 166064   CLR      @CSR
013164 113737 001306 001274  MOVB    TUMTAB,RCV DAT ;GET THE RECEIVED CHARACTER
013172 117037 013234 001276  MOVB    @AREA(0),XMTDAT ;GET THE TRANSMITTED CHARACTER
013200 043737 001300 001276  BIC     CARMSK,XMTDAT ;CLEAR NON-TRANSMITTED BITS
013206 123737 001274 001276  CMPB    RCV DAT,XMTDAT ;COMPARE CHARACTERS
013214 001733          BEQ     4$ ;BRANCH IF VALID COMPARISON
013216 104011          ERROR1 ;ERROR!DATA COMPARISON ERROR NUMBER
013220 000404          BR      6$ ;IN S/B GIVES MEMORY LOCATION (SEE TABLE)
013222 022626          5$: POPSP2 ;RESET THE STACK
013224 012737 000006 000004  MOV     #6,@#ERRVEC ;RESTORE TIME OUT TRAP
013232 104006          6$: SCOPE ;EXIT TEST
;MEMORY LOCATIONS TRANSMITTED FROM TABLE
013234 000000          AREA: 0 ;FOR DATA IN FIRST
013236 005252          5252 ;4K SEE THE LISTING
013240 012525          12525 ;CONTENTS OF THESE LOCATIONS (BYTE)
013242 017777          17777 ;IS THE DATA TRANSMITTED
013244 020000          A8K: 20000 ;CONTENTS =1 (IF AVAILABLE)
013246 026314          ; " 2 "
013250 031463          ; " 3 "
;
013252 037477          ; " 4 "
013254 040000          A12K: 40000 ; " 5 "
013256 057477          ; " 6 "
013260 060000          A16K: 60000 ; " 7 "
013262 077477          ; " 10 "
013264 100000          A20K: 100000 ; " 11 "
013266 117477          ; " 12 "
013270 120000          A24K: 120000 ; " 13 "
013272 137477          ; " 14 "
013274 140000          A28K: 140000 ; " 15 "
013276 173000          ; " 16 "
;*****
013300 000073          RT73: 73 ;ROUTINE # 73 *
013302 013514          RT74 ;ADDR OF NEXT ROUTINE *
013304 000012          10 ;ITERATION COUNT *
013306 013310          RT73A ;SCOPE ENTRY POINT. *
000073          X=X+1
;*****
;TEST THAT THE TRANSMITTER CAN TRANSMIT 100. CHARACTERS BEFORE SETTING
;THE READY BIT (CSR 15),AND CLEARING THE BAR BIT.
013310 005037 016412          RT73A: CLR     LINE
013314 012777 000001 165726  1$: MOV     #1,@CSR ;SET THE GO BIT
013322 005037 001274          CLR     RCV DAT
013326 013737 016412 001276  MOV     @#LINE,XMTDAT ;GET LINE NUMBER (X2)
013334 004537 005754          JSR     5,@#XMITD ;TRANSMIT 100. CHARACTERS
013340 177634          -100. ;ON LINE AS SPECIFIED BY LINE
013342 105777 165702          2$: TSTB   @CSR ;WAIT FOR THE RECEIVER
013346 100375          BPL     2$ ;TO RECEIVE ONE CHARACTER
013350 042777 000200 165672  BIC     #BIT7,@CSR ;CLEAR CHAR. DONE FLAG
013356 005237 001274          INC     RCV DAT ;INCREMENT CHAR. RCVD COUNT.
013362 023727 001274 000144  CMP     RCV DAT,#100. ;HAVE 100. CHARS. BEEN RCVD
013370 001416          BEQ     4$
013372 005777 165652          TST     @CSR ;TEST READY FLAG
013376 100002          BPL     3$ ;GO TEST BAR
```

```

013400 104011          ERROR1          ;ERROR!READY BIT SET TOO SOON
;TYPEOUT SHOWS HOW MANY CHARS WERE RECEIVED WHEN READY SET AND THE LINE # (X2)
013402 000443          BR          8$          ;GO TO EXIT

013404 023777 001272 165640 3$:  CMP      @#LINBIT,@BAR ;TEST THAT BAR BIT IS SET
013412 001753          BEQ      2$          ;BRANCH IF SET
013414 017737 165632 001276      MOV      @BAR,XMTDAT ;GET BAR CONTENTS
013422 104011          ERROR1          ;ERROR! BAR BIT CLEARED TOO SOON
;TYPEOUT SHOWS THE BAR CONTENTS AND HOW MANY CHARS WERE RECEIVED WHEN BAR FAILED
;LOCATION LINBIT HAS THE CORRECT BAR CONTENTS.
013424 000432          BR          8$          ;EXIT TEST
013426 013737 004076 013436 4$:  MOV      TIME14,5$ ;DELAY 1/4 CHARACTER TIME
013434 104400          DELAY          ;TO ALLOW TRANSMITTER TO FINISH
013436 000000          5$:  OPEN          ;
013440 005777 165604          TST      @CSR          ;TEST READY FLAG (SHOULD BE SET)
013444 100402          BMI      6$          ;GO TEST BAR
013446 104001          ERROR          ;ERROR! READY FLAG FAILED TO SET
013450 000420          BR          8$          ;GO TO EXIT
013452 005777 165574          6$:  TST      @BAR          ;TEST THAT BAR BIT IS CLEAR
013456 001407          BEQ      7$          ;GO TO 7$ IF CLEAR
013460 017737 165566 001274      MOV      @BAR,RCV DAT
013466 005037 001276          CLR      XMTDAT
013472 104011          ERROR1          ;ERROR! BAR BIT FAILED TO CLEAR
013474 000406          BR          8$          ;
013476 062737 000002 016412 7$:  ADD      #2,@#LINE
013504 006337 001272          ASL      @#LINBIT
013510 103301          BCC      1$          ;
013512 104006          8$:  SCOPE          ;
;*****
RT74:  74          ;ROUTINE # 74 *
013514 000074          RT75          ;ADDR OF NEXT ROUTINE. *
013516 013702          10.          ;ITERATION COUNT *
013520 000012          RT74A          ;SCOPE ENTRY POINT. *
013522 013524          X=X+1          ;
;*****
;TEST THAT THE TUMBLE TABLE POINTER INCREMENTS PROPERLY AND
;RETURNS TO THE BEGINNING AFTER 64. CHARACTERS HAVE BEEN RECEIVED
;LINE 0 IS USED FOR THE TEST.
013524 012701 001306      RT74A:  MOV      #TUMTAB,%1 ;CLEAP THE
013530 012702 000100      MOV      #64,%2 ;TUMBLE TABLE
013534 005021          1$:  CLR      (1)+
013536 005302          DEC      %2
013540 001375          BNE      1$
013542 012701 001306      MOV      #TUMTAB,%1
013546 012777 000004 165474      MOV      #BIT2,@CSR ;SET MAINT BIT & CLEAR GO BIT
013554 005037 001276      CLR      XMTDAT
013560 005037 001274      CLR      RCV DAT
013564 012737 177677 001146      MOV      #-65,%CT ;SET UP TO TRANSMIT 65. CHARACTERS
013572 052777 000001 165450      BIS      #BIT0,@CSR ;SET THE GO BIT
013600 012777 000001 165444      MOV      #LBIT0,@BAR ;TRANSMIT ON LINE 0
013606 105777 165436          2$:  TSTB     @CSR          ;WAIT FOR CHAR. DONE FLAG
013612 100375          BPL      2$
013614 042777 000200 165426      BIC      #BIT7,@CSR ;CLEAR CHAR. DONE FLAG
013622 005237 001274          INC      RCV DAT ;INCREMENT CHARACTERS
013626 005237 001276          INC      XMTDAT ;RECEIVED COUNT

```



```
013632 005711          TST      (1)          ;TEST TT ENTRY FOR VALID
013634 100402          BMI      35           ;DATA ENTRY
013636 104011          ERROR1          ;ERROR! NO VALID DATA ENTRY
                                ;TYPEOUT SHOWS # OF CHARS. RCVD WHEN ERROR OCCURED
013640 000417          BR       45           ;GO TO SCOPE
013642 005021          CLR      (1)+        ;CLEAR TT ENTRY
013644 023727 001276 000100 35:  CMP      XMTDAT,#64.  ;HAVE 64. CHARACTERS BEEN RECEIVED
013652 001355          BNE      25           ;
013654 005777 165370  TST      @CSR        ;WAIT FOR THE LAST CHARACTER
013660 100375          BPL      -4          ;TO BE TRANSMITTED
013662 105777 165362  TSTB    @CSR        ;TEST FOR DONE
013666 100375          BPL      -4          ;
013670 005737 001306  TST      TUMTAB      ;TEST FIRST TT ENTRY
013674 100401          BMI      45           ;FOR VALID DATA
013676 104001          ERROR          ;ERROR! POINTER DID NOT RETURN
013700 104006          45:  SCOPE          ;SCOPE
      000000
      000000
;*****
013702 000075          RT75:  75           ;ROUTINE # 75 *
013704 013720          RT76          ;ADDRESS OF NEXT TEST *
013706 000144          100.          ;ITERATION COUNT *
013710 013712          BRK0          ;SCOPE ENTRY POINT *
      000075
      X=X+1
;*****
;BREAK TEST ON LINE 0.
013712 004537 005164  BRK0:  JSR      5, BRKTST. ;GO DO BREAK TEST
013716 000001          LBIT0          ;ON LINE 0
      000001
      Y=Y+1
;*****
013720 000076          RT76:  76           ;ROUTINE # 76 *
013722 013736          RT77          ;ADDRESS OF NEXT TEST *
013724 000144          100.          ;ITERATION COUNT *
013726 013730          BRK1          ;SCOPE ENTRY POINT *
      000076
      X=X+1
;*****
;BREAK TEST ON LINE 1.
013730 004537 005164  BRK1:  JSR      5, BRKTST. ;GO DO BREAK TEST
013734 000002          LBIT1          ;ON LINE 1
      000002
      Y=Y+1
;*****
013736 000077          RT77:  77           ;ROUTINE # 77 *
013740 013754          RT100         ;ADDRESS OF NEXT TEST *
013742 000144          100.          ;ITERATION COUNT *
013744 013746          BRK2          ;SCOPE ENTRY POINT *
      000077
      X=X+1
;*****
;BREAK TEST ON LINE 2.
013746 004537 005164  BRK2:  JSR      5, BRKTST. ;GO DO BREAK TEST
013752 000004          LBIT2          ;ON LINE 2
      000003
      Y=Y+1
;*****
013754 000100          RT100:  100          ;ROUTINE # 100 *
013756 013772          RT101         ;ADDRESS OF NEXT TEST *
013760 000144          100.          ;ITERATION COUNT *
013762 013764          BRK3          ;SCOPE ENTRY POINT *
```

```
000100 X=X+1
;*****
;BREAK TEST ON LINE 3.
013764 004537 005164 BRK3: JSR 5, BRKTST ;GO DO BREAK TEST
013770 000010 ;ON LINE 3
000004 Y=Y+1
;*****
013772 000101 RT101: 101 ;ROUTINE # 101 *
013774 014010 RT102 ;ADDRESS OF NEXT TEST *
013776 000144 100 ;ITERATION COUNT *
014000 014002 BRK4 ;SCOPE ENTRY POINT *
000101 X=X+1
;*****
;BREAK TEST ON LINE 4.
014002 004537 005164 BRK4: JSR 5, BRKTST ;GO DO BREAK TEST
014006 000020 ;ON LINE 4
000005 Y=Y+1
;*****
014010 000102 RT102: 102 ;ROUTINE # 102 *
014012 014026 RT103 ;ADDRESS OF NEXT TEST *
014014 000144 100 ;ITERATION COUNT *
014016 014020 BRK5 ;SCOPE ENTRY POINT *
000102 X=X+1
;*****
;BREAK TEST ON LINE 5.
014020 004537 005164 BRK5: JSR 5, BRKTST ;GO DO BREAK TEST
014024 000040 ;ON LINE 5
000006 Y=Y+1
;*****
014026 000103 RT103: 103 ;ROUTINE # 103 *
014030 014044 RT104 ;ADDRESS OF NEXT TEST *
014032 000144 100 ;ITERATION COUNT *
014034 014036 BRK6 ;SCOPE ENTRY POINT *
000103 X=X+1
;*****
;BREAK TEST ON LINE 6.
014036 004537 005164 BRK6: JSR 5, BRKTST ;GO DO BREAK TEST
014042 000100 ;ON LINE 6
000007 Y=Y+1
;*****
014044 000104 RT104: 104 ;ROUTINE # 104 *
014046 014062 RT105 ;ADDRESS OF NEXT TEST *
014050 000144 100 ;ITERATION COUNT *
014052 014054 BRK7 ;SCOPE ENTRY POINT *
000104 X=X+1
;*****
;BREAK TEST ON LINE 7.
014054 004537 005164 BRK7: JSR 5, BRKTST ;GO DO BREAK TEST
014060 000200 ;ON LINE 7
000010 Y=Y+1
;*****
014062 000105 RT105: 105 ;ROUTINE # 105 *
014064 014100 RT106 ;ADDRESS OF NEXT TEST *
014066 000144 100 ;ITERATION COUNT *
014070 014072 BRK10 ;SCOPE ENTRY POINT *
000105 X=X+1
```

```
*****  
;BREAK TEST ON LINE 10.  
014072 004537 005164 BRK10: JSR 5, BRKTST ;GO DO BREAK TEST  
014076 000400 ;ON LINE 10  
000011 Y=Y+1  
*****  
RT106: 106 ;ROUTINE # 106 *  
014100 000106 ;ADDRESS OF NEXT TEST *  
014102 014116 ;ITERATION COUNT *  
014104 000144 ;SCOPE ENTRY POINT *  
014106 014110 X=X+1  
000106 BRK11  
*****  
;BREAK TEST ON LINE 11.  
014110 004537 005164 BRK11: JSR 5, BRKTST ;GO DO BREAK TEST  
014114 001000 ;ON LINE 11  
000012 Y=Y+1  
*****  
RT107: 107 ;ROUTINE # 107 *  
014116 000107 ;ADDRESS OF NEXT TEST *  
014120 014134 ;ITERATION COUNT *  
014122 000144 ;SCOPE ENTRY POINT *  
014124 014126 X=X+1  
000107 BRK12  
*****  
;BREAK TEST ON LINE 12.  
014126 004537 005164 BRK12: JSR 5, BRKTST ;GO DO BREAK TEST  
014132 002000 ;ON LINE 12  
000013 Y=Y+1  
*****  
RT110: 110 ;ROUTINE # 110 *  
014134 000110 ;ADDRESS OF NEXT TEST *  
014136 014152 ;ITERATION COUNT *  
014140 000144 ;SCOPE ENTRY POINT *  
014142 014144 X=X+1  
000110 BRK13  
*****  
;BREAK TEST ON LINE 13.  
014144 004537 005164 BRK13: JSR 5, BRKTST ;GO DO BREAK TEST  
014150 004000 ;ON LINE 13  
000014 Y=Y+1  
*****  
RT111: 111 ;ROUTINE # 111 *  
014152 000111 ;ADDRESS OF NEXT TEST *  
014154 014170 ;ITERATION COUNT *  
014156 000144 ;SCOPE ENTRY POINT *  
014160 014162 X=X+1  
000111 BRK14  
*****  
;BREAK TEST ON LINE 14.  
014162 004537 005164 BRK14: JSR 5, BRKTST ;GO DO BREAK TEST  
014166 010000 ;ON LINE 14  
000015 Y=Y+1  
*****  
RT112: 112 ;ROUTINE # 112 *  
014170 000112 ;ADDRESS OF NEXT TEST *  
014172 014206 ;ITERATION COUNT *  
014174 000144 ;SCOPE ENTRY POINT *  
014176 014200 X=X+1  
000112 BRK15  
*****
```

```
014200 004537 005164 ;BREAK TEST ON LINE 15.  
014204 020000 BRK15: JSR 5, BRKTST ;GO DO BREAK TEST  
000016 LBIT15 ;ON LINE 15  
Y=Y+1  
;*****  
014206 000113 RT113: 113 ;ROUTINE # 113 *  
014210 014224 RT114 ;ADDRESS OF NEXT TEST *  
014212 000144 100 ;ITERATION COUNT *  
014214 014216 BRK16 ;SCOPE ENTRY POINT *  
000113 X=X+1  
;*****  
014216 004537 005164 ;BREAK TEST ON LINE 16.  
014222 040000 BRK16: JSR 5, BRKTST ;GO DO BREAK TEST  
000017 LBIT16 ;ON LINE 16  
Y=Y+1  
;*****  
014224 000114 RT114: 114 ;ROUTINE # 114 *  
014226 014242 RT115 ;ADDRESS OF NEXT TEST *  
014230 000144 100 ;ITERATION COUNT *  
014232 014234 BRK17 ;SCOPE ENTRY POINT *  
000114 X=X+1  
;*****  
014234 004537 005164 ;BREAK TEST ON LINE 17.  
014240 100000 BRK17: JSR 5, BRKTST ;GO DO BREAK TEST  
000020 LBIT17 ;ON LINE 17  
000000 Y=Y+1  
000000 A=0  
000000 Y=0  
;*****  
014242 000115 RT115: 115 ;ROUTINE #115 *  
014244 014260 RT116 ;ADDRESS OF NEXT TEST *  
014246 000144 100 ;ITERATION COUNT *  
014250 014252 DAT0 ;SCOPE ENTRY POINT *  
000115 X=X+1  
;*****  
014252 004537 006020 ;DATA TEST 100 CHARACTERS LINE0  
014256 000000 DAT0: JSR 5, DATTST ;GO RUN DATA TEST  
000001 LINE0 ;ON LINE0  
Y=Y+1  
;*****  
014260 000116 RT116: 116 ;ROUTINE #116 *  
014262 014276 RT117 ;ADDRESS OF NEXT TEST *  
014264 000144 100 ;ITERATION COUNT *  
014266 014270 DAT1 ;SCOPE ENTRY POINT *  
000116 X=X+1  
;*****  
014270 004537 006020 ;DATA TEST 100 CHARACTERS LINE1  
014274 000002 DAT1: JSR 5, DATTST ;GO RUN DATA TEST  
000002 LINE1 ;ON LINE1  
Y=Y+1  
;*****  
014276 000117 RT117: 117 ;ROUTINE #117 *  
014300 014314 RT120 ;ADDRESS OF NEXT TEST *  
014302 000144 100 ;ITERATION COUNT *  
014304 014306 DAT2 ;SCOPE ENTRY POINT *  
000117 X=X+1
```

014306 004537 006020  
014312 000004  
000003

```
*****  
; DATA TEST 100 CHARACTERS LINE2  
DAT2: JSR 5, DATTST ; GO RUN DATA TEST  
LINE2 ; ON LINE2  
Y=Y+1
```

014314 000120  
014316 014332  
014320 000144  
014322 014324  
000120

```
*****  
RT120: 120 ; ROUTINE #120 *  
RT121 ; ADDRESS OF NEXT TEST *  
100. ; ITERATION COUNT *  
DAT3 ; SCOPE ENTRY POINT *  
X=X+1
```

014324 004537 006020  
014330 000006  
000004

```
*****  
; DATA TEST 100 CHARACTERS LINE3  
DAT3: JSR 5, DATTST ; GO RUN DATA TEST  
LINE3 ; ON LINE3  
Y=Y+1
```

014332 000121  
014334 014350  
014336 000144  
014340 014342  
000121

```
*****  
RT121: 121 ; ROUTINE #121 *  
RT122 ; ADDRESS OF NEXT TEST *  
100. ; ITERATION COUNT *  
DAT4 ; SCOPE ENTRY POINT *  
X=X+1
```

014342 004537 006020  
014346 000010  
000005

```
*****  
; DATA TEST 100 CHARACTERS LINE4  
DAT4: JSR 5, DATTST ; GO RUN DATA TEST  
LINE4 ; ON LINE4  
Y=Y+1
```

014350 000122  
014352 014366  
014354 000144  
014356 014360  
000122

```
*****  
RT122: 122 ; ROUTINE #122 *  
RT123 ; ADDRESS OF NEXT TEST *  
100. ; ITERATION COUNT *  
DAT5 ; SCOPE ENTRY POINT *  
X=X+1
```

014360 004537 006020  
014364 000012  
000006

```
*****  
; DATA TEST 100 CHARACTERS LINE5  
DAT5: JSR 5, DATTST ; GO RUN DATA TEST  
LINES ; ON LINES  
Y=Y+1
```

014366 000123  
014370 014404  
014372 000144  
014374 014376  
000123

```
*****  
RT123: 123 ; ROUTINE #123 *  
RT124 ; ADDRESS OF NEXT TEST *  
100. ; ITERATION COUNT *  
DAT6 ; SCOPE ENTRY POINT *  
X=X+1
```

014376 004537 006020  
014402 000014  
000007

```
*****  
; DATA TEST 100 CHARACTERS LINE6  
DAT6: JSR 5, DATTST ; GO RUN DATA TEST  
LINE6 ; ON LINE6  
Y=Y+1
```

014404 000124  
014406 014422  
014410 000144  
014412 014414  
000124

```
*****  
RT124: 124 ; ROUTINE #124 *  
RT125 ; ADDRESS OF NEXT TEST *  
100. ; ITERATION COUNT *  
DAT7 ; SCOPE ENTRY POINT *  
X=X+1  
*****
```

014414 004537 006020  
014420 000016  
000010

; DATA TEST 100 CHARACTERS LINE7  
DAT7: JSR 5,DATTST ; GO RUN DATA TEST  
LINE7 ; ON LINE7  
Y=Y+1

014422 000125  
014424 014440  
014426 000144  
014430 014432  
000125

\*\*\*\*\*  
RT125: 125 ; ROUTINE #125 \*  
RT126 ; ADDRESS OF NEXT TEST \*  
100. ; ITERATION COUNT \*  
DAT10 ; SCOPE ENTRY POINT \*  
X=X+1

014432 004537 006020  
014436 000020  
000011

\*\*\*\*\*  
; DATA TEST 100 CHARACTERS LINE10  
DAT10: JSR 5,DATTST ; GO RUN DATA TEST  
LINE10 ; ON LINE10  
Y=Y+1

014440 000126  
014442 014456  
014444 000144  
014446 014450  
000126

\*\*\*\*\*  
RT126: 126 ; ROUTINE #126 \*  
RT127 ; ADDRESS OF NEXT TEST \*  
100. ; ITERATION COUNT \*  
DAT11 ; SCOPE ENTRY POINT \*  
X=X+1

014450 004537 006020  
014454 000022  
000012

\*\*\*\*\*  
; DATA TEST 100 CHARACTERS LINE11  
DAT11: JSR 5,DATTST ; GO RUN DATA TEST  
LINE11 ; ON LINE11  
Y=Y+1

014456 000127  
014460 014474  
014462 000144  
014464 014466  
000127

\*\*\*\*\*  
RT127: 127 ; ROUTINE #127 \*  
RT130 ; ADDRESS OF NEXT TEST \*  
100. ; ITERATION COUNT \*  
DAT12 ; SCOPE ENTRY POINT \*  
X=X+1

014466 004537 006020  
014472 000024  
000013

\*\*\*\*\*  
; DATA TEST 100 CHARACTERS LINE12  
DAT12: JSR 5,DATTST ; GO RUN DATA TEST  
LINE12 ; ON LINE12  
Y=Y+1

014474 000130  
014476 014512  
014500 000144  
014502 014504  
000130

\*\*\*\*\*  
RT130: 130 ; ROUTINE #130 \*  
RT131 ; ADDRESS OF NEXT TEST \*  
100. ; ITERATION COUNT \*  
DAT13 ; SCOPE ENTRY POINT \*  
X=X+1

014504 004537 006020  
014510 000026  
000014

\*\*\*\*\*  
; DATA TEST 100 CHARACTERS LINE13  
DAT13: JSR 5,DATTST ; GO RUN DATA TEST  
LINE13 ; ON LINE13  
Y=Y+1

014512 000131  
014514 014530  
014516 000144  
014520 014522  
000131

\*\*\*\*\*  
RT131: 131 ; ROUTINE #131 \*  
RT132 ; ADDRESS OF NEXT TEST \*  
100. ; ITERATION COUNT \*  
DAT14 ; SCOPE ENTRY POINT \*  
X=X+1

\*\*\*\*\*  
; DATA TEST 100 CHARACTERS LINE14

014522 004537 006020  
014526 000030  
000015

DAT14: JSR 5,DATTST ;GO RUN DATA TEST  
LINE14 ;ON LINE14  
Y=Y+1

\*\*\*\*\*

014530 000132  
014532 014546  
014534 000144  
014536 014540  
000132

RT132: 132 ;ROUTINE #132 \*  
RT133 ;ADDRESS OF NEXT TEST \*  
100. ;ITERATION COUNT \*  
DAT15 ;SCOPE ENTRY POINT \*  
X=X+1

\*\*\*\*\*

;DATA TEST 100 CHARACTERS LINE15

014540 004537 006020  
014544 000032  
000016

DAT15: JSR 5,DATTST ;GO RUN DATA TEST  
LINE15 ;ON LINE15  
Y=Y+1

\*\*\*\*\*

014546 000133  
014550 014564  
014552 000144  
014554 014556  
000133

RT133: 133 ;ROUTINE #133 \*  
RT134 ;ADDRESS OF NEXT TEST \*  
100. ;ITERATION COUNT \*  
DAT16 ;SCOPE ENTRY POINT \*  
X=X+1

\*\*\*\*\*

;DATA TEST 100 CHARACTERS LINE16

014556 004537 006020  
014562 000034  
000017

DAT16: JSR 5,DATTST ;GO RUN DATA TEST  
LINE16 ;ON LINE16  
Y=Y+1

\*\*\*\*\*

014564 000134  
014566 014602  
014570 000144  
014572 014574  
000134

RT134: 134 ;ROUTINE #134 \*  
RT135 ;ADDRESS OF NEXT TEST \*  
100. ;ITERATION COUNT \*  
DAT17 ;SCOPE ENTRY POINT \*  
X=X+1

\*\*\*\*\*

;DATA TEST 100 CHARACTERS LINE17

014574 004537 006020  
014600 000036  
000020

DAT17: JSR 5,DATTST ;GO RUN DATA TEST  
LINE17 ;ON LINE17  
Y=Y+1

\*\*\*\*\*

014602 000135  
014604 015202  
014606 000144  
014610 014612  
000135

RT135: 135 ;ROUTINE # 135 \*  
RT136 ;ADDR OF NEXT ROUTINE. \*  
100. ;ITERATION COUNT \*  
RT135A ;SCOPE ENTRY POINT. \*  
X=X+1

\*\*\*\*\*

;TEST THAT DATA (ALL 1'S) CAN BE TRANSMITTED ON LINES SIMULTANEOUSLY.

;THE FOLLOWING TESTS ARE PERFORMED:

; THERE ARE 16. DATA ENTRIES  
; THERE ISN'T A 17TH ENTRY  
; DATA RECEIVED IS CORRECT  
; ONE DATA ENTRY PER LINE

014612 005037 001306  
014616 004537 005732  
014622 001306  
014624 001307  
014626 000177

RT135A: CLR TUMTAB ;CLEAR THE  
JSR 5,BMOVE ;TUMBLE  
TUMTAB ;TABLE  
TUMTAB+1 ;(200  
177 ;ENTRIES)

```

014630 012737 177777 016564      MOV    #-1,OUTBUF      ;LOAD CHAR INTO OUTPUT BUFFER
014636 005000                    CLR    %0              ;SET RO = LINE 0
014640 012737 000001 001272      MOV    #LBITO,LINBIT  ;GET LINE BIT
014646 012777 000001 164374      MOV    #BITO,@CSR    ;SET THE GO BIT
014654 010037 016412          15:   MOV    %0,LINE        ;GET LINE NUMBER
014660 004537 005754          JSR    5,XMITD        ;TRANSMIT 1 CHAR.
014664 177777                    -1      ;ON EACH LINE
014666 005720                    TST    (0)+           ;INCREMENT LINE NUMBER (+2)
014670 006337 001272          ASL    LINBIT         ;SHIFT LINE BIT TO NEXT LINE
014674 103367                    BCC    15            ;BRANCH IF ALL LINES NOT DONE
014676 013737 004074 014706      MOV    TIME1,25      ;PUT TIME TO TRANSMIT 1 CHAR
014704 104400                    DELAY 1              ;DELAY 1
014706 000000          25:   OPEN                    ;CHARACTER TIME
014710 017737 164336 001274      MOV    @BAR,RCVDAT   ;GET & TEST BAR CONTENTS
014716 001410                    BEQ    35            ;BRANCH IF 0
014720 005037 001276          CLR    XMTDAT
014724 104011                    ERROR1                ;ERROR! BAR NOT CLEAR AFTER ALL
014726 005077 164320          CLR    @BAR          ;LINES FINISHED
014732 005077 164312          CLR    @CSR
014736 000520                    BR     165          ;GO TO EXIT
014740 032777 020000 164302  35:   BIT    #BIT13,@CSR   ;TEST THAT OVER RUN DID NOT SET
014746 001404                    BEQ    45            ;BRANCH IF 0
014750 104001                    ERROR                ;ERROR! OVER RUN BIT SET
014752 005077 164272          CLR    @CSR
014756 000510                    BR     165          ;GO TO EXIT

;TEST THAT THERE ARE 16. VALID DATA ENTRIES
014760 005077 164264          45:   CLR    @CSR          ;CLEAR THE CSR
014764 012702 000020          MOV    #16,%2        ;GET TT SCAN COUNT
014770 012701 001306          MOV    #TUMTAB,%1    ;GET FIRST TT ADDRESS
014774 005302          55:   DEC    %2            ;DECREMENT SCAN COUNTER
014776 100404                    BMI    65            ;BRANCH IF 16. ENTRIES SCANNED
015000 005721                    TST    (1)+          ;TEST FOR VALID DATA ENTRY
015002 100774                    BMI    55            ;BRANCH IF FOUND
015004 104001                    ERROR                ;ERROR! MISSING DATA ENTRY
015006 000474                    BR     165          ;GO TO EXIT
015010 005721          65:   TST    (1)+          ;TEST 17TH ENTRY (SHOULD BE = TO 0)
015012 001402                    BEQ    75            ;BRANCH IF 0
015014 104001                    ERROR                ;ERROR! EXTRA DATA ENTRY
015016 000470                    BR     165          ;GO TO EXIT

;TEST THAT THE DATA IS CORRECT IN ALL 16. ENTRIES
015020 012701 001306          75:   MOV    #TUMTAB,%1    ;GET FIST TT ADDRESS
015024 012702 000020          MOV    #16,%2        ;GET SCAN COUNT
015030 005302          85:   DEC    %2            ;DECREMENT SCAN COUNT
015032 100421                    BMI    105          ;BRANCH IF 16. ENTRIES SCANNED
015034 013737 016564 001276      MOV    OUTBUF,XMTDAT ;GET TRANSMITTED DATA
015042 043737 001300 001276      BIC    CARMSK,XMTDAT ;CLEAR NON-TRANSMITTED BITS
015050 113737 001306 001274      MOVB  TUMTAB,RCVDAT  ;GET RECEIVED DATA
015056 123737 001276 001274      CMPB  XMTDAT,RCVDAT  ;COMPARE DATA
015064 001402                    BEQ    95            ;BRANCH IF 0
015066 104011                    ERROR1                ;ERROR INCORRECT DATA
015070 000443                    BR     165          ;GO TO EXIT
015072 005721          95:   TST    (1)+          ;INCREMENT TT ADDRESS
015074 000755                    BR     85            ;TEST NEXT ENTRY

```



```
; CLEAR ALL BUT LINE NUMBER IN TUMBLE TABLE ENTRY
015076 012701 001306 105: MOV #TUMTAB,%1 ; GET FIRST TT ADDRESS
015102 012702 000020 MOV #16,%2 ; GET SCAN COUNT
015106 005302 115: DEC %2 ; DECREMENT SCAN COUNT
015110 100403 BMI 125 ; BRANCH IF ALL LINES TESTED
015112 042721 160777 RIC #160777,(1)+ ; CLEAR ALL BUT LINE NUMBER IN TT
015116 000773 BR 115 ; DO NEXT TT ADDRESS

; TEST THAT THERE IS AN ENTRY FOR EACH OF THE 16. LINES
015120 005037 001276 125: CLR XMTDAT
015124 012701 000020 MOV #16,%1
015130 012702 000020 135: MOV #16,%2
015134 012700 001306 MOV #TUMTAB,%0
015140 023720 001276 145: CMP XMTDAT,(0)+ ; TEST FOR LINE ENTRY
015144 001406 BEQ 155 ; BRANCH IF FOUND
015146 005302 DEC %2 ; DECREMENT SCAN COUNT
015150 001373 BNE 145 ; LOOK AT NEXT ENTRY
015152 005037 001274 CLR RCVDAT
015156 104011 ERROR1 ; ERROR! NO ENTRY FOUND FOR THIS LINE
015160 000407 BR 165 ; GO TO EXIT
015162 005301 155: DEC %1 ; DECREMENT LINES FOUND COUNT
015164 005701 TST %1
015166 001404 BEQ 165 ; BRANCH IF ALL LINE TESTED
015170 062737 001000 001276 ADD #1000,XMTDAT ; INCREMENT LINE NUMBER
015176 000754 BR 135 ; GO DO NEXT LINE
015200 104006 165: SCOPE ; SCOPE
; *****
RT136: 136 ; ROUTINE # 136 *
137 ; ADDR OF NEXT ROUTINE. *
100 ; ITERATION COUNT *
RT136A ; SCOPE ENTRY POINT. *
X=X+1
; *****

; TEST THAT THE DM11 CAN TRANSMIT A BREAK ON ALL LINES SIMULTANEOUSLY
015212 013737 004074 015266 RT136A: MOV @#TIME1,15 ; GET TIME TO TRANSMIT ONE CHARACTER
015220 005037 001306 CLR TUMTAB ; CLEAR
015224 004537 005732 JSR 5,BMOVE ; THE
015230 001306 TUMTAB ; TUMBLE
015232 001307 TUMTAB+1 ; TABLE
015234 000177 177
015236 012777 000001 164004 MOV #BIT0,@CSR ; SET GO
015244 012777 177777 164002 MOV #-1,@BKCSR ; SET BREAK BIT FOR ALL LINES
015252 105777 163772 TSTB @CSR ; WAIT FOR THE RECEIVER
015256 100375 BPL -4 ; TO RECEIVE A BREAK
015260 005077 163770 CLR @BKCSR ; CLEAR ALL BREAK BITS
015264 104400 DELAY ; WAIT ONE CHARACTER
015266 000000 15: OPEN ; TIME
015270 022777 000201 163752 CMP #201,@CSR ; TEST THAT ONLY GO AND DONE ARE SET
015276 001410 BEQ 25
015300 017737 163744 001274 MOV @CSR,RCVDAT ; GET CSR ENTRY
015306 012737 000201 001276 MOV #201,XMTDAT ; GET CORRECT RESULT
015314 104011 ERROR1 ; ERROR! INCORRECT CSR DATA
015316 000476 BR 135 ; EXIT

; TEST THAT THERE IS 16. VALID DATA ENTRIES
```

015320 012701 001306  
015324 012702 000020  
015330 005721  
015332 100402  
015334 104001  
015336 000466  
015340 005302  
015342 001372

25: MOV #TUMTAB,%1 ;GET TUMBLE TABLE BASE ADDRESS  
MOV #16,%2 ;GET SCAN COUNT  
35: TST (1)+ ;TEST FOR VALID DATA ENTRY  
BMI 45 ;BRANCH IF VALID DATA ENTRY FOUND  
ERROR ;ERROR! MISSING VALID DATA ENTRY  
BR 135 ;EXIT  
45: DEC %2 ;DECREMENT SCAN COUNT  
BNE 35 ;BRANCH IF 16. ENTRIES NOT SCANNED

;TEST THAT THE BREAK BIT IS SET IN 16. TUMBLE TABLE ENTRIES

015344 012701 001306  
015350 012702 000020  
015354 032721 040000  
015360 001002  
015362 104001  
015364 000453  
015366 005302  
015370 001371

MOV #TUMTAB,%1  
MOV #16,%2  
55: BIT #BIT14,(1)+ ;BREAK BIT SET?  
BNE 65 ;BRANCH IF SET  
ERROR ;ERROR! MISSING BREAK BIT  
BR 135 ;EXIT  
65: DEC %2 ;DECREMENT SCAN COUNT  
BNE 55

;TEST THAT THE TUMBLE TABLE DATA BYTE IS ALL 0'S

015372 012701 001306  
015376 012702 000020  
015402 105721  
015404 001402  
015406 104001  
015410 000441  
015412 105721  
015414 005302  
015416 001371

MOV #TUMTAB,%1  
MOV #16,%2  
75: TSTB (1)+ ;TEST DATA BYTE  
BEQ 85 ;BRANCH IF 0'S  
ERROR ;ERROR! INCORRECT DATA  
BR 135 ;EXIT  
85: TSTB (1)+ ;STEP TABLE POINTER TO NEXT DATA BYTE  
DEC %2  
BNE 75

;CLEAR ALL BUT LINE NUMBER IN TUMBLE TABLE ENTRY

015420 012701 001306  
015424 012702 000020  
015430 042721 160777  
015434 005302  
015436 001374

MOV #TUMTAB,%1  
MOV #16,%2  
95: BIC #160777,(1)+ ;CLEAR ALL BUT LINE NUMBER  
DEC %2  
BNE 95

;TEST THAT THERE IS A TUMBLE TABLE ENTRY FOR EACH LINE

015440 005004  
015442 012703 000020  
015446 012702 000020  
015452 012701 001306  
015456 020421  
015460 001410  
015462 005302  
015464 001374  
015466 010437 001276  
015472 010437 001274  
015476 104011  
015500 000405  
015502 005303  
015504 001403  
015506 062704 001000  
015512 000755  
015514 104006

CLR %4 ;CLEAR LINE NUMBER  
MOV #16,%3  
105: MOV #16,%2  
MOV #TUMTAB,%1  
115: CMP %4,(1)+ ;TEST FOR LINE ENTRY FOR THIS LINE  
BEQ 125 ;BRANCH IF FOUND  
DEC %2  
BNE 115  
MOV %4,XMTDAT  
MOV %4,RCVDAT  
ERROR1 ;ERROR! NO LINE ENTRY FOUND FOR THIS LINE  
BR 135 ;EXIT  
125: DEC %3 ;ALL LINES BEEN FOUND  
BEQ 135 ;EXIT IF YES  
ADD #1000,%4 ;SEARCH FOR  
BR 105 ;NEXT LINE  
135: SCOPE ;SCOPE

\*\*\*\*\*

015516 000137  
015520 015534  
015522 000002  
015524 015526  
000137

RT137: 137 ;ROUTINE # 137 \*  
RT140 ;ADDR OF NEXT ROUTINE. \*  
2 ;ITERATION COUNT \*  
RT137A ;SCOPE ENTRY POINT \*  
X=X+1

\*\*\*\*\*

;TEST TO TRANSMIT ON EACH LINE WITH A DELAY BEFORE STATING THE  
;NEXT LINE.

015526 004537 005310  
015532 000040

RT137A: JSR 5,@#DLYXMT ;GO DO TEST. DELAY \*  
32. ;THIS MUCH BETWEEN LINES \*

\*\*\*\*\*

015534 000140  
015536 015552  
015540 000002  
015542 015544  
000140

RT140: 140 ;ROUTINE # 140 \*  
RT141 ;ADDR OF NEXT ROUTINE. \*  
2 ;ITERATION COUNT \*  
RT140A ;SCOPE ENTRY POINT. \*  
X=X+1

\*\*\*\*\*

;TEST TO TRANSMIT ON EACH LINE WITH A DELAY BEFORE STATING THE  
;NEXT LINE.

015544 004537 005310  
015550 000020

RT140A: JSR 5,@#DLYXMT ;GO DO TEST. DELAY \*  
16. ;THIS MUCH BETWEEN LINES \*

\*\*\*\*\*

015552 000141  
015554 015570  
015556 000002  
015560 015562  
000141

RT141: 141 ;ROUTINE # 141 \*  
RT142 ;ADDR OF NEXT ROUTINE. \*  
2 ;ITERATION COUNT \*  
RT141A ;SCOPE ENTRY POINT. \*  
X=X+1

\*\*\*\*\*

;TEST TO TRANSMIT ON EACH LINE WITH A DELAY BEFORE STATING THE  
;NEXT LINE.

015562 004537 005310  
015566 000010

RT141A: JSR 5,@#DLYXMT ;GO DO TEST. DELAY \*  
8. ;THIS MUCH BETWEEN LINES \*

\*\*\*\*\*

015570 000142  
015572 015606  
015574 000002  
015576 015600  
000142

RT142: 142 ;ROUTINE # 142 \*  
RT143 ;ADDR OF NEXT ROUTINE. \*  
2 ;ITERATION COUNT \*  
RT142A ;SCOPE ENTRY POINT. \*  
X=X+1

\*\*\*\*\*

;TEST TO TRANSMIT ON EACH LINE WITH A DELAY BEFORE STATING THE  
;NEXT LINE.

015600 004537 005310  
015604 000004

RT142A: JSR 5,@#DLYXMT ;GO DO TEST. DELAY \*  
4 ;THIS MUCH BETWEEN LINES \*

\*\*\*\*\*

015606 000143  
015610 015624  
015612 000002  
015614 015616  
000143

RT143: 143 ;ROUTINE # 143 \*  
RT144 ;ADDR OF NEXT ROUTINE. \*  
2 ;ITERATION COUNT \*  
RT143A ;SCOPE ENTRY POINT. \*  
X=X+1

\*\*\*\*\*

;TEST TO TRANSMIT ON EACH LINE WITH A DELAY BEFORE STATING THE

```

;NEXT LINE.
015616 004537 005310 RT143A: JSR      5, @#DLYXMT ;GO DO TEST. DELAY
015622 000002          2          ;THIS MUCH BETWEEN LINES
;*****
015624 000144 RT144:  144          ;ROUTINE # 144 *
015626 015642          RT145          ;ADDR OF NEXT ROUTINE. *
015630 000002          2          ;ITERATION COUNT *
015632 015634          RT144A         ;SCOPE ENTRY POINT. *
          000144          X=X+1
;*****

;TEST TO TRANSMIT ON EACH LINE WITH A DELAY BEFORE STATING THE
;NEXT LINE.
015634 004537 005310 RT144A: JSR      5, @#DLYXMT ;GO DO TEST. DELAY
015640 000001          1          ;THIS MUCH BETWEEN LINES
;*****
015642 000145 RT145:  145          ;ROUTINE # 145 *
015644 016010          RT146          ;ADDR OF NEXT ROUTINE. *
015646 000144          100          ;ITERATION COUNT *
015650 015652          RT145A         ;SCOPE ENTRY POINT. *
          000145          X=X+1
;*****

;TEST THAT THE DM11 WORKS PROPERLY WHEN THE HALF-DUPLEX BIT (CSR BIT 1)
;IS SET. THE TEST TRANSMITS DATA ON LINE 0, AND 'BREAKS' ON LINE 1. ONLY
;THE BREAK SHOULD BE RECEIVED ON LINE 0 IN THE TUMBLE TABLE.
015652 005037 001306 RT145A: CLR      TUMTAB ;CLEAR THE FIRST TWO
015656 005037 001310          CLR      TUMTAB+2 ;TUMBLE TABLE ENTRIES
015662 012737 016564 001106          MOV      #OUTBUF, CAT ;SET UP TO
015670 012737 177777 001146          MOV      #-1, WCT ;TRANSMIT 1 CHARACTER
015676 012777 000007 163344          MOV      #7, @CSR ;SET GO, HALF DUPLEX & MAINT BITS
015704 012777 000001 163340          MOV      #LBIT0, @BAR ;TRANSMIT 1 CHAP. ON LHNE 0
015712 012777 000002 163334          MOV      #LBIT1, @BKCSR ;SET BREAK ON LINE 1
015720 105777 163324          TSTB   @CSR ;WAIT FOR THE CHARACTER
015724 100375          BPL      -4 ;TO BE RECEIVED
015726 005077 163322          CLR      @BKCSR ;CLEAR THE BREAK BIT ON LINE 1

;TEST THAT ONLY THE BREAK WAS RECEIVED
015732 022737 141000 001306          CMP      #141000, TUMTAB ;TDST FOR BREAK ENTRY (LINE 1)
015740 001410          BEQ      15
015742 013737 001306 001274          MOV      TUMTAB, RCVDAT ;GET ACTUAL ENTRY
015750 012737 141000 001276          MOV      #141000, XMTDAT ;GET CORRECT ENTRY
015756 104011          ERROR1 ;ERROR! INCORRECT BREAK ENTRY
015760 000407          BR      25 ;GO TO EXIT
015762 013737 001310 001274 15: MOV      TUMTAB+2, RCVDAT ;TEST THAT NEXT ENTRY IS CLEAR
015770 001403          BEQ      25 ;EXIT IF CORRECT
015772 005037 001276          CLR      XMTDAT
015776 104011          ERROR1 ;ERROR! SECOND ENTRY WAS NOT CLEAR
016000 005777 163244 25: TST      @CSR ;WAIT FOR THE TRANSMITTER
016004 100375          BPL      -4 ;TO FINISH
016006 104006          SCOPE ;SCOPE
;*****
016010 000146 RT146:  146          ;ROUTINE # 146 *
016012 177777          RTLAST         ;ADDR OF NEXT ROUTINE. *
016014 000144          100          ;ITERATION COUNT *
016016 016020          RT146A         ;SCOPE ENTRY POINT. *
```

000146

X=X+1

\*\*\*\*\*

;TEST THAT THE DM11 RESPONDS CORRECTLY TO A RESET

016020	012737	016564	001106	RT146A:	MOV	#OUTBUF,CAT	;SET UP TO TRANSMIT 10
016026	012737	177770	001146		MOV	#-10,WCT	;CHARACTERS ON LINE 0
016034	013737	004074	016104		MOV	TIME1,2%	;GET TIME TO TRANSMIT 2 CHARACTERS
016042	013737	004074	016150		MOV	TIME1,6%	
016050	005037	001276			CLR	XMTDAT	
016054	012777	000007	163166		MOV	#7,@CSR	;SET MAINT., HALF DUPLEX & GO BITS
016062	012777	000001	163162		MOV	#LBIT0,@BAR	;START TO TRANSMIT ON LINE 0
016070	012777	000002	163156		MOV	#LBIT1,@BKCSR	;BREAK ON LINE 1
016076	012704	000002			MOV	#2,%4	
016102	104400			1%	DELAY		;WAIT 2 CHARACTER
016104	000000			2%	OPEN		;TIMES
016106	005304				DEC	%4	
016110	001374				BNE	1%	
016112	104005				SRESET		;RESET
016114	017737	163130	001274		MOV	@CSR,RCV DAT	;GET CSR CONTENTS
016122	001401				BEQ	3%	;BRANCH IF 0
016124	104011				ERROR1		;ERROR! CSR DID NOT CLEAR
016126	017737	163120	001274	3%	MOV	@BAR,RCV DAT	;GET BAR CONTENTS
016134	001402				BEQ	4%	;BRANCH IF 0
016136	104011				ERROR1		;ERROR! BAR IS NOT CLEAR
016140	000427				BR	9%	;EXIT
016142	012704	000010		4%	MOV	#8,%4	
016146	104400			5%	DELAY		;WAIT 8 MORE CHARACTER TIMES
016150	000000			6%	OPEN		
016152	005304				DEC	%4	
016154	001374				BNE	5%	
016156	017737	163066	001274		MOV	@CSR,RCV DAT	;TEST THAT CSR IS CLEAR
016164	001402				BEQ	7%	
016166	104011				ERROR1		;ERROR! CSR WAS NOT CLEAR
016170	000413				BR	9%	;GO TO EXIT
016172	017737	163054	001274	7%	MOV	@BAR,RCV DAT	;TES THAT BAR IS CLEAR
016200	001402				BEQ	8%	
016202	104011				ERROR1		;ERROR! BAR DID NOT CLEAR
016204	000405				BR	9%	
016206	017737	163042	001274	8%	MOV	@BKCSR,RCV DAT	;TEST THAT BKCSR IS CLEAR
016214	001401				BEQ	9%	
016216	104011				ERROR1		;ERROR! BKCSR DID NOT CLEAR
016220	104006			9%	SCOPE		;SCOPE

```
016222                                     ;PRG1- TRANSMITTER SCOPE LOOP
016222 104000                               PRG1:                                     ;BEGIN
016224 017543                               TYPE                                     ;TYPE PROGRAM TITLE
016226 004737 016402                       PRG1M
016232 004737 016454                       JSR 7,PARAM                               ;GO GET USER PARAMETERS
016236 005777 163010                       PRG1R: JSR 7,LOOP                          ;GO LOOP TRANSMITTER
016242 001375                               PRG1B: TST @BAR                            ;WAIT FOR ALL LINES TO FINISH
016244 005077 163000                       BNE PRG1B                                ;BRANCH IF NOT DONE
016250 005037 016264                       PRG1C: CLR @CSR                            ;CLEAR THE CSR
016254 117737 162623 016264               CLR PRG1D                                ;CLEAR DELAY TIME
016262 104400                               MOV8 @SWR+1,PRG1D                         ;GET DELAY
016264 000000                               DELAY                                     ;DELAY AS SPECIFIED
016266 000761                               PRG1D: OPEN                               ;BY USER
                                                BP PRG1R                                ;LOOP BACK
```

```

; PRG2- RECEIVER SCOPE LOOP
PRG2:
016270                                ; BEGIN
016270 104000                          ; TYPE PROGRAM
016272 017561                          ; TITLE
016274 004737 016402                    ; GO GET USER PARAMETERS
016300 004737 016454                    ; GO START TRANSMITTER
016304 012777 000001 162736            PRG2R: JSR 7, PARAM
016312 005777 162734                    PRG2A: MOV #BIT0, @CSR
016316 001415                          PRG2AA: TST @BAR
016320 105777 162724                    BEQ PRG2B
016324 100372                          TSTB @CSR
016326 042777 000200 162714            BPL PRG2AA
016334 020127 001504                    BIC #BIT7, @CSR
016340 001002                          CMP %1, #TUMTAB+176
016342 012701 001304                    BNE .+6
016346 005721                          MOV #TUMTAB-2, %1
016350 000755                          TST (1)+
016352 005077 162672                    PRG2B: BR PRG2A
016356 005077 162670                    CLR @CSR
016362 005037 016376                    CLR @BAR
016366 117737 162511 016376            PRG2C: CLR PRG2C
016374 104400                          MOV @SWR+1, PRG2C
016376 000000                          DELAY
016400 000737                          OPEN
BR PRG2R                                ; REPEAT LOOP
  
```

```

; SUBROUTINE TO GET USER PARAMETERS (FOR PRG1 & 2)
PARAM: TYPE
016402 104000                          ; ASK USER WHICH LINE
016404 017600                          ; TO TEST
016406 004537 004124                    JSR 5, RECD
016412 000000                          ; GET LINE AND PUT IT
016414 104000                          ; HERE
016416 017623                          PARAMA: TYPE
016420 004537 004124                    JSR 5, RECD
016424 000000                          ; ASK USER HOW MANY
016426 023727 016424 000310            CHARS: CMP CHARS, #200
016434 101403                          ; CHARACTERS TO TRANSMIT
016436 104000                          ; GET CHARS AND PUT IT
016440 017444                          ; HERE
016442 000764                          BR PARAMA
016444 104000                          PARAMB: TYPE
016446 017467                          ; RE-REQUEST PARAMETER
016450 104015                          ; TYPE INSTRUCTIONS
016452 000207                          M4
CNTLU
RTS 7
; GO GET VALUE
; EXIT
  
```

```

; SUBROUTINE TO TRANSMIT DATA FROM THE SR
LOOP: MOV @SWR, OUTBUF
016454 117737 162422 016564            JSR 5, BMOVE
016462 004537 005732                    OUTBUF
016466 016564                          ; WITH
016470 016565                          ; DATA TO BE
016472 000307                          ; TRANSMITTED
016474 012777 001106 162554            MOV #CAT, @BASREG
016502 012737 016564 001106            MOV #OUTBUF, CAT
016510 004537 005732                    JSR 5, BMOVE
; INITIALIZE BASE REGISTER
; LOAD CURRENT
; ADDRESS TABLE
  
```

```

016514 001106          CAT          ;WITH ADDRESS
016516 001110          CAT+2        ;OF OUTPUT BUFFER
016520 000040          32.
016522 013737 016424 001146  MOV      CHARS,WCT    ;LOAD WORD COUNT
016530 005437 001146          NEG      WCT          ;FORM TWO'S COMPLEMENT
016534 004537 005732          JSR      5,BMOVE      ;TABLE WITH
016540 001146          WCT          ;NUMBER OF
016542 001150          WCT+2        ;CHARACTERS TO BE
016544 000040          32.          ;TRANSMITTED
016546 013737 016412 001272  MOV      LINE,LINBIT  ;SAVE LINES TO BE TRANSMITTED ON
016554 013777 016412 162470  MOV      LINE,@BAR    ;START TRANSMITTING ON SELECTED LINES
016562 000207          RTS      7          ;EXIT
  
```

```

016564 000000          OUTBUF: 0          ;FIRST ADDRESS OF 100.
016730 016730          . =OUTBUF+100. ;CHARACTER OUTPUT BUFFER
016730 000000          INBUF: 0          ;FIRST ADDRESS OF 100.
017074 017074          . =INBUF+100. ;CHARACTER INPUT BUFFER (WHERE RECEIVED
                                ;DATA IS STORED)
  
```

```

017074 013746 000006          SUSWRP: MOV      @#6,-(SP)    ;SAVE VECTORS
017100 013746 000004          MOV      @#4,-(SP)
017104 012737 017124 000004  MOV      #1$,@#4      ;SET UP FOR TIMEOUT
017112 022777 177777 161762  CMP      #-1,@SWR     ;REFERENCE HARDWARE SWITCH REGISTER
017120 001402          BEQ      2$
017122 000407          BR      3$
017124 022626          1$: CMP      (SP)+,(SP)+ ;ADJUST STACK
017126 012737 000176 001102  2$: MOV      #SWREG,SWR ;POINT TO SOFTWARE SWITCH REG
017134 012737 000174 001104  MOV      #DISPREG,DISPLAY ;POINT TO SOFT DISPLAY REG
017142 012637 000004          3$: MOV      (SP)+,@#4    ;RESTORE VECTORS
017146 012637 000006          MOV      (SP)+,@#6
017152 000002          RTI
  
```

;ROUTINE TO CHECK FOR G BEING TYPED

```

017154 022737 000176 001102  KBDINTT: CMP      #SWREG,SWR
017162 001021          BNE      1$
017164 023737 000042 000046  CMP      @#42,@#46    ;ACT11?
017172 001415          BEQ      1$          ;BR IF YES
017174 005037 017266          CLR      TMP1        ;CLEAR TEMP APEA
017200 117737 162372 017266  MOV      @TKDBR,TMP1  ;FETCH THE BUFFER
017206 142737 000200 017266  BICB    #200,TMP1     ;STRIP OFF PARITY
017214 122737 000007 017266  CMP      #7,TMP1     ;WAS IT G
017222 001001          BNE      1$          ;NOP
017224 104015          CNTLU   ;GO CHANGE IT
017226 000002          1$: RTI          ;EXIT
  
```

;ROUTINE TO CHANGE CONTENTS OF SWREG(LOC 176)

```

017230 022737 000176 001102  CNTLUU: CMP      #SWREG,SWR
017236 001023          BNE      FJX
017240 104000          TYPE
017242 017323          $SWREG
017244 004537 005662          JSR      R5,@ACNV    ;CONVERT TO ASCII
  
```



017250	000176				SWREG				
017252	017331				SVALUE				
017254	000006				6				
017256	104000				TYPE				
017260	017331				SVALUE				
017262	004537	004124			JSR	5, RECD			; GET TMP1 AND PUT IT
017266	000000			TMP1:	0				; HERE
017270	022737	000007	004302		CMP	#7, CNT			
017276	001403				BEQ	FJX			
017300	013777	017266	161574		MOV	TMP1, @SWR			; CHANGE CONTENTS OF SWREG
017306	000002			FJX:	RTI				

```

; MESSAGES
017310 053045 041505 020124 WHERE: . ASCII '%VECT ADR?@'
017316 042101 037522 100
017323 045 053523 036522 $$WREG. . ASCII '%SWR=@'
017330 100
017331 040 020040 020040 $VALUE: . ASCII ' NEW=@'
017336 020040 020040 047040
017344 053505 040075
017350 036445
017352 052445 044516 021524 $CTLU: . ASCII '%='
017360 034050 037451 100 WHICH: . ASCII '%UNIT#(8)?@'
017365 045 044103 051101 LEVEL: . ASCII '%CHAR LNGTH@'
017372 046040 043516 044124
017400 100
017401 045 051105 020122 ERDAT: . ASCII '%ERR S/B. '
017406 027523 035102 040
017413 040 020040 020040 AASB: . ASCII ' WAS: '
017420 020040 040527 035123
017426 040
017427 040 020040 020040 AWAS: . ASCII ' @'
017434 040040
017436 050045 043522 040043 M0: . ASCII '%PRG#@'
017444 037445 100 M1: . ASCII '%?@'
017447 045 047105 040104 M2: . ASCII '%END@'
017454 051445 036522 037460 M3: . ASCII '%SP=0? GO. @'
017462 043440 027117 100
017467 045 042114 041440 M4: . ASCII '%LD CHAR IN SPO-7; DLY IN SPS-15@'
017474 040510 020122 047111
017502 051440 030122 033455
017510 042073 054514 044440
017516 020116 051123 026470
017524 032461 100
017527 045 047514 044507 PRG0M: . ASCII '%LOGIC TSTS@'
017534 020103 051524 051524
017542 100
017543 045 046530 052111 PRG1M: . ASCII '%XMITTER LOOP@'
017550 042524 020122 047514
017556 050117 100
017561 045 046530 052111 PRG2M: . ASCII '%XMIT/REC LOOP@'
017566 051057 041505 046040
017574 047517 040120
017600 052045 050131 046040 LINPAR: . ASCII '%TYP LINES TO TST @'
017606 047111 051505 052040
017614 020117 051524 020124
017622 100
017623 045 047443 020106 HOWMAN: . ASCII '%#OF CHARS?@'
017630 044103 051101 037523
017636 100
017637 045 020122
017642 020040 050040 036503 EMO: . ASCII '%R '
017650 020040 020040 020040 ATNUMB: . ASCII ' PC='
017656 100 APC: . ASCII ' @'
017657 015 012 MSG1: . BYTE 15,12
017661 124 042510 050440 . ASCII 'THE QUICK BROWN FOX JUMPED OVER THE LAZY DOGS BACK 1234567890'
017666 044525 045503 041040
017674 047522 047127 043040
  
```

017702	054117	045040	046525
017710	042520	020104	053117
017716	051105	052040	042510
017724	046040	055101	020131
017732	047504	051507	041040
017740	041501	020113	031061
017746	032063	033065	034067
017754	030071		
	000001		

END

A = 000000	CAT = 001106	ERROR = 104001	LINE12= 000024	PAR5 = 006546
AASB 017413	CC = 177776	ERROR1= 104011	LINE13= 000026	PAR6 = 006550
APC 017650	CHALT = 104003	ERRVEC= 000004	LINE14= 000030	PASS = 001624
APCADD 001610	CHARS 016424	ERR1 = 001672	LINE15= 000032	PCADD = 001606
AREA 013234	CLKINT 001260	ESCOPE 002406	LINE16= 000034	PFAIL = 004304
ATNUMB 017642	CLKLVL 001262	FJX 017306	LINE17= 000036	POPSP = 005726
AWAS 017427	CNT 004302	FORWD 002554	LINE2 = 000004	POPSP2= 022626
A12K 013254	CNTLU = 104015	FORWDA 002600	LINE3 = 000006	PRGL IM 001522
A16K 013260	CNTLUU 017230	GETRDY 002306	LINE4 = 000010	PRGNUM 002162
A20K 013264	CNV DAT 002040	GTBIN 005612	LINE5 = 000012	PRGTAB 001524
A24K 013270	COUNT 001604	GTLINB 006552	LINE6 = 000014	PRGO = 006604
A28K 013274	CSR 001250	GTRDYA 002324	LINE7 = 000016	PRGOM 017527
ASK 013244	CURTST 001510	GTRDYC 002346	LINPAR 017600	PRGOR 006626
BAR 001252	DATCHK= 104002	GTRDYD 002370	LOOP 016454	PRG1 = 016222
BASREG 001256	DATTST 006020	GTRDYX 002314	LTST0 011162	PRG18 016236
BAT 001206	DATO 014252	HOWMAN 017623	LTST1 011200	PRG1C 016244
BELL = 000007	DAT1 014270	ICTR 001516	LTST10 011342	PRG1D 016264
BIT0 = 000001	DAT1AA 006024	INBIN 005562	LTST11 011360	PRG1M 017543
BIT1 = 000002	DAT10 014432	INBUF 016730	LTST12 011376	PRG1R 016232
BIT10 = 002000	DAT11 014450	INCR TN 001626	LTST13 011414	PRG2 = 016270
BIT11 = 004000	DAT12 014466	INIT 003142	LTST14 011432	PRG2A 016304
BIT12 = 010000	DAT13 014504	INITIA= 104012	LTST15 011450	PRG2AA 016312
BIT13 = 020000	DAT14 014522	KBDIN = 104014	LTST16 011466	PRG2B 016352
BIT14 = 040000	DAT15 014540	KBDINT 017154	LTST17 011504	PRG2C 016376
BIT15 = 100000	DAT16 014556	KSTART 001506	LTST2 011216	PRG2M 017561
BIT2 = 000004	DAT17 014574	LBIT0 = 000001	LTST3 011234	PRG2R 016300
BIT3 = 000010	DAT2 014306	LBIT1 = 000002	LTST4 011252	PRTY0 = 000000
BIT4 = 000020	DAT3 014324	LCIT10= 000400	LTST5 011270	PRTY1 = 000040
BIT5 = 000040	DAT4 014342	LBIT11= 001000	LTST6 011306	PRTY2 = 000100
BIT6 = 000100	DAT5 014360	LBIT12= 002000	LTST7 011324	PRTY3 = 000140
BIT7 = 000200	DAT6 014376	LBIT13= 004000	MACHER 000004	PRTY4 = 000200
BIT8 = 000400	DAT7 014414	LBIT14= 010000	MANUAL= 100000	PRTY5 = 000240
BIT9 = 001000	DELAY = 104400	LBIT15= 020000	MSG1 017657	PRTY6 = 000300
BKCSR 001254	DISPLA 001104	LBIT16= 040000	M0 017436	PRTY7 = 000340
BMOVE 005732	DISPRE 000174	LBIT17= 100000	M1 017444	PRVCNT 001612
BRKTST 005164	DLY 005506	LBIT2 = 000004	M2 017447	PSW = 177776
BRK0 013712	DLYXMT 005310	LBIT3 = 000010	M3 017454	PTO = 005606
BRK1 013730	DMPAR 003172	LBIT4 = 000020	M4 017467	PT1 = 005610
BRK10 014072	DMPARB 003454	LBIT5 = 000040	NOF = 000240	PWRUP = 004314
BRK11 014110	DMPARC 003600	LBIT6 = 000100	NXTST 001514	RCVDAT 001274
BRK12 014126	DTCHK 001634	LBIT7 = 000200	OACNV 005662	RCVTST 004542
BRK13 014144	EHALT = 104004	LENGTH 003620	OACNVA 005674	RCVO = 011522
BRK14 014162	EMO 017637	LENOKA 003636	OPEN = 000000	RCV1 = 011540
BRK15 014200	EMTINT 002606	LENOKB 003646	OUTBUF 016564	RCV10 011702
BRK16 014216	EMTTAB 001540	LENOKC 003654	OVLAY 004100	RCV11 011720
BRK17 014234	ERDAT 017401	LENOKD 003714	PARAM 016402	RCV12 011736
BRK2 013746	ERR 001654	LEVEL 017365	PARAMA 016414	RCV13 011754
BRK3 013764	ERRA 001712	LINBIT 001272	PARAMB 016444	RCV14 011772
BRK4 014002	ERRB 001754	LINE 016412	PARO 006534	RCV15 012010
BRK5 014020	ERRC 001762	LINE0 = 000000	PAR1 006536	RCV16 012026
BRK6 014036	ERRD 002014	LINE1 = 000002	PAR2 006540	RCV17 012044
BRK7 014054	ERREX 002034	LINE10= 000020	PAP3 006542	RCV2 = 011556
CARMSK 001300	ERRHLT 002032	LINE11= 000022	PAP4 006544	RCV3 = 011574

RCV4	011612	RT121	014332	RT21	010456	RT62	011762	TIME14	004076
RCV5	011630	RT122	014350	RT21A	010466	RT63	012000	TKCSR	001574
RCV6	011646	RT123	014366	RT22	010564	RT64	012016	TKDBR	001576
RCV7	011664	RT124	014404	RT22A	010574	RT65	012034	TMP1	017266
RECD	004124	RT125	014422	RT23	010672	RT66	012054	TPCSR	001600
RIND	005604	RT126	014440	RT23A	010702	RT66A	012070	TPDBR	001602
RNGEN	002756	RT127	014456	RT24	011000	RT67	012314	TTDAT	001270
RORPAR	006472	RT13	007652	RT24A	011010	RT67A	012324	TUMTAB	001306
RP1	003024	RT13A	007662	RT25	011044	RT7	007302	TYP	003030
RP2	003026	RT130	014474	RT25A	011054	RT7A	007312	TYPA	003040
RSTART	001532	RT131	014512	RT26	011152	RT70	012424	TYPC	003056
RSTAT1	002134	RT132	014530	RT27	011170	RT70A	012434	TYPD	003074
RSTAT2	002222	RT133	014546	RT3	007032	RT71	012540	TYPDAT	003140
RSTPC	002734	RT134	014564	RT3A	007042	RT71A	012550	TYPE =	104000
RSTPSW	002736	RT135	014602	RT30	011206	RT72	012734	TYPF	003112
RSTREG=	104010	RT135A	014612	RT31	011224	RT72A	012744	TYPG	003124
RSTRG	002700	RT136	015202	RT32	011242	RT73	013300	UNIT	003474
RTLAST=	177777	RT136A	015212	RT33	011260	RT73A	013310	UNTOKA	003514
RTNNO	001512	RT137	015516	RT34	011276	RT74	013514	UNTOKB	003540
RT0	006644	RT137A	015526	RT35	011314	RT74A	013524	UNTOKC	003570
RT0A	006654	RT14	007764	RT36	011332	RT75	013702	VAC	001246
RT1	006706	RT14A	007774	RT37	011350	RT76	013720	VECOK	003374
RT1A	006716	RT140	015534	RT4	007104	RT77	013736	VECOKA	003404
RT10	007354	RT140A	015544	RT4A	007114	SAVREG=	104007	VECOKB	003412
RT10A	007364	RT141	015552	RT40	011366	SAVRG	002640	VECTOR	003356
RT100	013754	RT141A	015562	RT41	011404	SCOPE =	104006	WCT	001146
RT101	013772	RT142	015570	RT42	011422	SCOPEA	002444	WHERE	017310
RT102	014010	RT142A	015600	RT43	011440	SCOPEB	002450	WHICH	017352
RT103	014026	RT143	015606	RT44	011456	SCOPEP	002504	X =	000146
RT104	014044	RT143A	015616	RT45	011474	SCOPTR	001520	XMITD	005754
RT105	014062	RT144	015624	RT46	011512	SPBOT	001100	XMTDAT	001276
RT106	014100	RT144A	015634	RT47	011530	SRESET=	104005	XMTINT	001264
RT107	014116	RT145	015642	RT5	007156	SRSET	002254	XMTLVL	001266
RT11	007426	RT145A	015652	RT5A	007166	SRSETT	002740	XMTTST	004330
RT11A	007436	RT146	016010	RT50	011546	START	002066	Y =	000020
RT110	014134	RT146A	016020	RT51	011564	SUSWR =	104013	\$CTLU	017350
RT111	014152	RT15	010024	RT52	011602	SUSWRR	017074	\$ENDAD	002544
RT112	014170	RT15A	010034	RT53	011620	SVRPC	002674	\$SWREG	017323
RT113	014206	RT16	010160	RT54	011636	SVRPSW	002676	\$VALUE	017331
RT114	014224	RT16A	010170	RT55	011654	SWR	001102	=	017756
RT115	014242	RT17	010276	RT56	011672	SWREG	000176	BAR	001616
RT116	014260	RT17A	010306	RT57	011710	TIMEA	004000	BASRE	001622
RT117	014276	RT2	006760	RT6	007230	TIMEB	004034	BKCSR	001620
RT12	007516	RT2A	006770	RT6A	007240	TIMEC	004044	CSR	001614
RT12A	007526	RT20	010370	RT60	011726	TIMER	003724		
RT120	014314	RT20A	010400	RT61	011744	TIME1	004074		

ABS. 017756 000

ERRORS DETECTED: 0

.DZDMAC.SEQ/NL: SEQ=DZDMAC.P11  
RUN-TIME: 59.5 SECONDS  
RUN-TIME RATIO: 426/15=27.5  
CORE USED: 13K (25 PAGES)

MAINDEC-11-DZDMA-C DM11 LOGIC TESTS  
DZDMAC.P11 22-NOV-77 17:57

M 6  
MACY11 30(1046) 30-NOV-77 09:34 PAGE 22-2  
SYMBOL TABLE

SEQ 0077