

MAY 13 1966

PROGRAM TYPES OUT
' NO PROGRAM INTERRUPT '
DUE TO AN ECO

Run w/sw 5 up.

RIM loader

- 1. IDENTIFICATION
- 1.1 Digital-7-92-M, Maindec 722
- 1.2 PDP-7 EAE Part 1
- 1.3 March 24, 1966

2. ABSTRACT

Part 1 of the PDP-4/7 EAE Diagnostic verifies correct operation of all EAE operations, except multiplies and divides. Part 1 is written in three logical sections. Part 1 Section 1 is the EAE Set-Up Test and verifies that all set-up operations except LACS operate correctly. Part 1 Section 2 is the Shift Counter (LACS is verified) and Basic Shift Test and verification that the AC and MQ will each shift left 1 and shift right 1 all combinations of 18 bits. Part 1 Section 3 is the Random Data, Normalize, and Interrupt Test verifying that random data will shift left and right 0 to 44₈ places, that normalize will "stop shift" on negative and positive data, and that the teleprinter flag will cause a break after an EAE operation. Hardware malfunctions detected by the program result in an error on the teleprinter.

3. REQUIREMENTS

3.1 Storage

CAL subroutine	00020-00027
AC contents initial	00030
MQ contents initial	00031
Link initial	00032
SC of shift instructions	00033
AC contents as result	00034
MQ contents as result	00035
Link as result	00036
SC of LACS instruction	00037
Halt and/or Scope Loop subroutine	00040-00057
Halt and/or Repeat Sequence subroutine	00060-00077
Set-Up Test	00100-01000 (approx.)
Error Timeout subroutine	
Error texts and program constants	01035-02100 (approx.)
SC and Basic Shift Test	02200-04600 (approx.)
Random Data and Normalize	05000-06400 (approx.)

3.2 Subprograms and/or Subroutines
PDP-4/7 Teletype Output Package
(ASCII tape 2A of this test)

3.3 Equipment

Minimum configuration PDP-4/7 with EAE option installed.

4. USAGE

4.1 Loading

Normal binary loading procedures are to be used. *RIM LOADER*

4.2 Calling Sequence

Part 1 Section 1 must run in its entirety and at all margins before running Part 1 Section 2.

Part 1 Section 2 must run in its entirety and at all margins before running Part 1 Section 3.

4.3 Switch Settings

4.3.1 AC switches = 0 or down. With all AC Switches down the program results in the following:

(1) All hardware malfunctions detected by the program result in an error typeout on the teleprinter.

(2) At the completion of an error typeout the processor halts.

(3) The program repeats whichever section of the test it was started in and sequences from each sub-test of that section to the next without halting.

4.3.2 AC switches = 1 or up

SW #	Operation	Description
0	Delete error typeouts	The program will not type out error messages and will not error halt (see also SW0 and 7, Ring Bell on Error).
1	Halt after EAE operation Processor halts at address 0046 (AC) = S.A. to set up last operation	The processor halts after each EAE operation is initiated and its results are verified. (Note: Press CONTINUE to proceed.)
2	Repeat EAE operation (Scope Loop)	The program repeats the last EAE operation. If SW2 is set during an error typeout or halt, the program repeats the operation that caused the error (Note: SW1 is tested before SW2.)
3	Halt after EAE sequence Processor halts at address 0066 (AC) = S.A. of last sequence	The processor halts after each sequence of testing an EAE operation; i.e., after testing that the MQ will complement all patterns, the processor halts.

SW #	Operation	Description
4	Repeat EAE sequence	The program repeats the last sequence of testing an EAE operation; i.e., the program repeats the LEFT SHIFT ALL COMBINATIONS and does not proceed to RIGHT SHIFT ALL COMBINATIONS. (Note: The program tests SW3 before SW4) In the Random Data Left and Random Data Right routines SW4 causes the program to repeatedly shift a single pair of random numbers 0 to 44g places.
5	Cycle all sections	At the completion of 1 pass through the Set-Up Test the program proceeds to the SC and Basic Shift Test. At the completion of 1 pass through the SC and Basic Shift Test the program proceeds to the Random Data and Normalize Test. At the completion of 1 pass through Random Data and Normalize Test the program repeats the Set-Up Test.
6	Type end of section	At completion of 1 pass through each of the sections a character is typed on the teleprinter as follows: <p style="text-align: right;">Set-Up Test / SC and Basic Shift Test , Random Data and Normalize *</p>
7	Delete error halt	The processor will not halt after error typeouts.
0 & 7	Ring bell on error	SW0 and SW7 both up. Error typeouts and halts are deleted and the "bell" on the teleprinter is rung (to be used to determine marginal voltage limits, eliminates waiting for long typeouts).

- 4.4 Start Up and/or Entry
- 4.4.1 Start Up, Set-Up Test
- Set AC switches = 000000
- Set ADDRESS = 0100
- Press START

Processor halts at 0101 with MQ = 777777

Set ADDRESS = 0102

Press START

Program reads C(MQ) into the AC and tests for 0, then proceeds to rest of test.

NOTE: This section of Part 1 must run at all margins before running Section 2.

4.4.2 Start Up, SC and Basic Shift Test

Set AC switches = 000000

Set ADDRESS = 2200

Press START

Processor halts at 2204 AC = 200000
SC = 77

Set ADDRESS = 2205

Press START

Program reads C(SC) into the AC and tests for 0, then proceeds to rest of test.

NOTE: This section must run at all margins before running Section 3.

4.4.3 Start Up Random Data and Normalize Test

Set AC switches = 000000

Set ADDRESS = 5000

Press START

NOTE: This section must run at all margins before running EAE Part 2.

4.5 Errors in Usage

Hardware malfunctions detected by the program will result in an error typeout on the teleprinter and a processor halt (see section 4.3.2, SW0 and SW7).

4.5.1 Error Typeout Format

All error typeouts are in standard formats and include the following information:

(1) An address that may be used to determine which test the program was in at the time the error was detected

(2) A mnemonic describing the operation being tested

- (3) The initial condition of registers pertinent to the failure
- (4) The expected results of the operation being tested if they are not easily determined from the initial conditions and operation
- (5) The resultant register contents that are pertinent to the failure

A common typeout routine called ERROR generates all error typeouts. The first line of every error typeout is the contents of memory register ERROR or the address + 1 of the JMS ERROR instruction.

The second line of every typeout is the mnemonic describing the operation being tested (see paragraph 4.5.2 for definitions of mnemonics used).

The third line of a typeout may be another address. In this case the second address typed should be used to determine which test failed. (Operations such as LRS or LLSS each have common error routines.)

The next information typed is a header to format the typeouts of the contents of pertinent registers. One of five headers may be used for any typeout.

The abbreviations used by the headers are as follows:

<u>Abbr.</u>	<u>Meaning</u>
L	The information under this column is the contents of the link.
C(AC)	The information under this column is the contents of the accumulator.
C(MQ)	The information under this column is the contents of the MQ register.
SC	The information under this column is the contents of the shift counter or the SC portion of shift instructions.
START	The information in this line is the initial condition of pertinent registers.

The five headers are as follows:

START	C(AC)			
START	C(AC)	C(MQ)		
START	L	C(AC)	C(MQ)	
START	SC	C(AC)		
L	C(AC)	C(MQ)		

4.5.2 Error Typeout Mnemonics

<u>Mnemonic</u>	<u>Description</u>
EAENOP	EAE instruction with no other operation specified.
EAECLA	EAE. Clear the accumulator.
CLQ	Clear the MQ register.
CMQ	Complement the MQ register.
ORMQAC	Inclusive OR the MQ to the AC and place the results in the AC.
AC0TOL	Set AC bit 0 into the link.
ORACMQ	Inclusive OR the AC to the MQ and place the results in the MQ (and in test ACORMQ clear the AC).
LACQ	Clear the AC, then MQ 1's to the AC.
LLS	Long left shift.
LLSS	Long left shift signed.
LRS	Long right shift.
LRSS	Long right shift signed.
LMQ	Clear the MQ, then AC 1's to the MQ.
ABS	Complement the AC if it is negative.
CLR Δ SC	Clear the step counter (START).
LACS	Clear the AC and step counter; 1's to the AC.
NORM	Normalize the AC and MQ.
NORMS	Normalize signed.
ALS	Accumulator left shift.
PAT	Pattern being tested.
COR	Results expected from the operation being tested.
INCO	Erroneous results of the operation.

Error Typeout Examples

The following are examples of error typeouts. The addresses indicated by these typeouts should not necessarily be taken as true representations:

Example 1: Complement the MQ Failure

	<u>Example</u>		<u>Explanation</u>
000226			JMS ERROR is at 00225
CMQ			Operation is complement the MQ
	C(AC)	C(MQ)	Header
START	000000	000000	Initial conditions
CMQ	000000	767777	Contents of the AC and MQ after CMQ was executed.

Note: Examine the MQ indicators to be sure they agree with the typeout. If the MQ as indicated does not agree with a typeout, an error was present in MQ 1's to the AC. This is true of all error typeouts that include the MQ as an end condition.

Example 2: EAE NOP AC Failure

	<u>Example</u>		<u>Explanation</u>
000135			JMS ERROR is at 00134
EAENOP			Operation is NOP 640000
	C(AC)		Header
START	777777		Initial condition of the AC
EAENOP	000000		Contents of the AC after the NOP was executed

Example 3: AC Sign to Link Failure

	<u>Example</u>			<u>Explanation</u>
000455				JMS ERROR is at 00454
AC0TOL				Operation is AC bit 0 to link
	L	C(AC)	C(MQ)	Header
START	1	400000		Initial conditions MQ not pertinent
AC0TOL	0	400000		State of the LINK and AC after the operation was executed

Example 4: AC to MQ to AC Failures

				<u>Example</u>	<u>Explanation</u>
000526					JMS ERROR is at 00525
ORACMQ					Operation is AC 1's to MQ
	C(AC)	C(AC)			Header
START	000000	000000			Initial register states
ORACMQ	000000	000000	COR		Expected results
LACQ	000000	040000	INCO		The contents of the AC after ORACMQ and the contents of the MQ as indicated by a LACQ instruction
000526					
ORACMQ					
	C(AC)	C(MQ)			
START	005000	000000			
ORACMQ	000000	005000	COR		
LACQ	000000	004000	INCO		

Note: Again, the contents of the MQ as indicated by the MQ indicators may not necessarily agree with the MQ contents as typed.

Example 5: Step Counter Error

				<u>Example</u>	<u>Explanation</u>
002530					JMS ERROR is at 02527
SC ERROR					One of the SC tests failed
002262					JMS SCERR is at 02261
	SC	C(AC)			Header
START	00	200000			Initial register status
NORM	01				Instruction used to set the SC
SET SC	76				NORM 01 should set the SC to 76
SC + 1	77	COR			SC should increment to 77
LACS	67	INCO	200000		Contents of the SC as read to the AC by a LACS instruction and the contents of the AC after the NORM instruction

Example 6: ALS (Accumulator Left Shift) Failure

<u>Example</u>		<u>Explanation</u>
003123		JMS ERROR is at 03122
ALS	05	ALS instruction 5 places
003076		JMS ALSERR is at 03075
L	C(AC) C(MQ)	Header
1	777776 PAT	Pattern being tested
1	777777 RESULT	Results in AC after the shift
LACS	00	Shift counter read back to the AC

Example 7: Long Left Shift

<u>Example</u>		<u>Explanation</u>
003673		JMS ERROR is at 03672
LLS	01	Long left shift 1 place
003507		JMS LLSERR is at 03506
L	C(AC) C(MQ)	Header
1	777777 777737 PAT	Initial register states
1	777777 777377 RESULT	Registers at completion of shift
LACS	00	SC as read back to the AC

Example 8: Long Left Shift Signed

<u>Example</u>		<u>Explanation</u>
003716		JMS ERROR is at 03715
LLSS	03	Long left shift signed 3 places
005075		JMS LRSSER is at 05074
L	C(AC) C(MQ)	Header
0	456701 234567 PAT	Pattern being tested
	567012 345677 COR	Expected results
1	567012 347677 INCO	L, AC, and MQ after the shift
LACS	00	SC as read back to the AC

Example 9: Long Right Shift

<u>Example</u>				<u>Explanation</u>
004600				JMS ERROR is at 004577
LRS	01			Long right shift 1 place
004537				JMS LRSER 1 is at 004536
L	C(AC)	C(MQ)		Header
1	402101	402101	PAT	Pattern being tested
	601200	601200	COR	Expected results
1	601200	601000	INCO	AC and MQ after completion of the shift
LACS	00			SC as read to the AC after completion of the shift

Example 10: Random Data Sequenced

<u>Example</u>				<u>Explanation</u>
005501				JMS ERROR is at 005500
	RANDOM DATA SEQUENCED 02			Random sequence 2
005301				JMS SEQCOM is at 005300
L	C(AC)	C(MQ)		Header
0	045670	123450	START	Pattern sequenced
0	045630	123450	RESULT	L, AC, and MQ after shift sequence
LACS	00			SC after shift sequence

Note: Sequence 2 is: LLSS 03
LRS 06
LLSS 06
LRS 03

The AC and MQ results should equal the AC and MQ at START. This is true of all of the Random Data Sequences.

Example 11: Normalize

<u>Example</u>			<u>Explanation</u>
006217			JMS ERROR
NORM	01		Normalize SC = 1
005766			JMS NORMER is at 05765
L	C(AC)	C(MQ)	Header
0	200000	000000	PAT Pattern being tested
0	400000	000000	RESULT L, AC, and MQ after NORM
LACS	77	COR	SC expected after the NORM
LACS	00	RESULT	SC read back to the AC

Example 12: Interrupt Failure

<u>Example</u>		<u>Explanation</u>
006310		JMS ERROR is at 06307
NO PROGRAM INTERRUPT		Error is no interrupt
EAE NOP		Instruction tested
006305		Address of NOP instruction

4.6 Recovery From Such Errors

4.6.1 General

At the completion of an error timeout the processor halts. One of the following operations may be necessary if more information about the failure is required to repair the malfunction:

1. Repeat the exact operation that detected the failure (possibly for a scope loop).
2. Continue normally in the test to generate more information about the failure.
3. Repeat the sequence of operations or data patterns that detected the error.

AC switch control is built into the program to allow for any of these operations. Assuming the processor has halted after an error timeout, the operations may be accomplished as follows:

1. Repeat same operation

Set AC switch 2 up or to a 1
Press CONTINUE

Note that AC SW0 allows deletion of error timeouts for a scope loop.

2. Continue normally

Press CONTINUE

3. Repeat Sequence

Set AC switch 4 up or to a 1

Press CONTINUE

In the Random Data Tests, switch 4 a 1 causes the same pair of random numbers to be repeatedly shifted 0 to 44g places. This is useful in determining which shift the random data first fails.

4.6.2 To Determine Area in Program that Failed

4.6.2.1 From Error Typeouts

Each error typeout includes an address typeout that may be used to determine the exact test routine that detected the error. Some of the typeouts include an address that points at a common error routine for that type of error and a second address that points at the test routine. (Section 4.5.3, example 3 has only one octal typeout before the header and example 5 has two. The second octal typeout in example 5 (002262) determines which SC test failed.) Determine which address to use, go to the numerically sorted program labels (section 10.4.1) and find the program labels with addresses lower and higher than the one typed. The last program label with an address lower than the one typed is in the test routine that failed.

4.6.2.2 From CAL Routine

This test program includes a halt at address 00026 that indicates a CAL instruction was executed. Pressing CONTINUE at this point causes the processor to CAL at address 00027. At the time of the first HALT the contents of the AC indicate the contents of address 00020 after the CAL or the address + 1 of the CAL. The approximate area of the test program that was being executed may be determined by examining the following memory addresses.

<u>Address</u>	<u>Contents Indicate</u>
00040	Address +1 or +2 of last JMS SWITCH
00057	Starting address of last SCOPE LOOP
00060	Address +1 or +2 of last JMS SWITCH
00077	Starting address of last TEST SEQUENCE

By comparing the contents of these memory locations with the numerically sorted symbol list, the test routine (at the time of a CAL, hang up, or program wipeout) that was being executed may be determined.

5. RESTRICTIONS (Not Applicable)

6. DESCRIPTION

6.1 Discussion

6.1.1 General

The PDP-4/7 EAE Diagnostic Part 1 verifies correct operation of all EAE operations except multiplies and divides. Part 1 itself is written in three logical sections as follows:

Section 1: Set-Up Test

Verifies correct operation of all EAE set-up operations except LACS.

Section 2: SC and Basic Shift Test

Verifies correct operation of the SC and LACS instruction and verifies that the AC and MQ will shift left and right 1 place all combinations of 18 bits.

Section 3: Random Data and Normalize Test

This section of Part 1 verifies that the AC and MQ will shift random data left and right 0 to 448 places, that the NORM and NORMS instructions operate correctly, and that the processor interrupts after an EAE operation.

The above sections are to be used incrementally. That is, Section 1 must operate at all margins before Section 2 is run. Section 2 must run at all margins before Section 3 is run.

6.1.2 Test Descriptions

6.1.2.1 Set-Up Test

The Set-Up Test incrementally verifies correct operation of all of the EAE set-up instructions except LACS.

The sequence of testing is as follows:

<u>Test Mnemonic</u>	<u>Operation(s) Tested</u>
SETUP	Does CMQ set MQ = 0's to 1's Do all MQ indicators light (visual)
EAERMQ	Does START clear the MQ Does MQ = 0's to AC = 0's
NOPAC	Does EAE NOP not clear the AC
EAECAC	Do EAE and bit 8 clear the AC
EAELQ	Does bit 5 clear the MQ
MQITAC	Does bit 16 with MQ = 1's set AC to 1's
NOPACI	Does EAE NOP with MQ = 1's alter the AC
NOPMQ	Does EAE NOP with MQ = 1's alter the MQ
NOPMQI	Does EAE NOP with AC = 1's alter the MQ
NOPLNK	Does EAE NOP alter the link

<u>Test Mnemonic</u>	<u>Operation(s) Tested</u>
QONEAC	Does MQ = 1's inclusive OR to AC = 1's
EAESLK	Do EAE and bit 4 get AC sign to link
NOPLKI	Does EAE NOP alter the MQ with link = 1
ACORMQ	Does AC inclusive OR all patterns to MQ = 0's and MQ to AC all patterns
ACLMQ	Does the LMQ instruction operate as specified
COMPMQ	Will the MQ complement all patterns
ACONEQ	Will the AC = 1's inclusive OR to MQ = 1's
EAEABS	Does the ABS instruction operate as specified

6.1.2.2 SC and Basic Shift Test

The SC and Basic Shift Test incrementally verifies correct operation of the SC (including the LACS instruction) and the left and right shifts. The SC Test assumes that a NORM instruction with the AC = 200000 generates a stop shift.

The sequence of testing is as follows:

<u>Test Mnemonic</u>	<u>Operation(s) Tested</u>
SCTSTI	(1) Does NORM "stop shift" with AC = 200000 (visual) SC is set to 77 (2) Does START clear the SC (3) Does LACS get SC = 0's to the AC
NOPSC	Does EAE NOP alter the SC = 0's
SCTO76	(1) Will the SC set to 76 and + 1 to 77 (2) Will LACS read SC = 77 to the AC
SCTO74	Will the SC set to 74 and + 1 to 75
SCTO70	Will the SC set to 70 and + 1 to 71
SCTO60	Will the SC set to 60 and + 1 to 61
SCTO40	Will the SC set to 40 and + 1 to 41
SCTO00	Will the SC set to 00 and + 1 to 01
SCTO01	Will the SC set to 01 and + 1 to 02
SCTO03	Will the SC set to 03 and + 1 to 04
SCTO07	Will the SC set to 07 and + 1 to 10 (Is "high count" generated?)

<u>Test Mnemonic</u>	<u>Operation(s) Tested</u>
SCTO17	Will the SC set to 17 and + 1 to 20
SCTO37	Will the SC set to 37 and + 1 to 40
SCTO77	Will the SC set to 77 and + 1 to 00
NOPSC1	Does EAE NOP alter SC = 77
ALSZER	Does ALS with SC = 00 "stop shift"
ALS01	Does ALS 1 place shift AC = 0's
ALSLNK	Does link get to AC17 on an ALS 1 place
LNKALS	Does bit 0 of the AC not go to the link on an ALS 1 place
ALSMQT	Does ALS alter the MQ Does MQ0 not go to AC17
HSALS	Will ALS shift the AC 1 to 18 places bit and no-bit
LLSTS1	Will the AC/MQ shift 0's place left
LLSTS2	Does link go to MQ17 on an LLS
LLSACT	(1) Does link not go to AC17 on an LLS (2) Does MQ0 go to AC17 on an LLS
LLSTS3	Does each bit of the MQ = 1 shift left 1 place (1 bit at a time = 1)
LLSTS4	Does each bit of the MQ = 0 shift left 1 place (1 bit at a time = 0)
LLSTS5	Will MQ/AC shift a 1 bit 1 to 44 _g places left
LLSTS6	Will MQ/AC shift a 0 bit 1 to 44 _g places left
LRSTS1	Will AC/MQ shift right 1 all 0's
LRSTS2	Does link go to AC0 on an LRS
LRSTS3	Does AC17 go to MQ0 on an LRS
LRSTS4	Does AC17 not go to link on an LRS
LRSTS5	Will AC/MQ shift a 1 bit from each position right 1 place (1 bit at a time)
LRSTS6	Will AC/MQ shift a 0 bit right 1 place (1 bit at a time)
LRSTS7	Will AC/MQ shift 1 bit (AC0) right 1 to 44 _g places
LRSTS8	Will AC/MQ shift a 0 bit (AC0) right 1 to 44 _g places

<u>Test Mnemonic</u>	<u>Operation(s) Tested</u>
LLSSEQ	Will the AC and MQ each shift left 1 place every combination of 18 bits
LRSSSEQ	Will the AC and MQ each shift right 1 place every combination of 18 bits

6.1.2.3 Random Data and Normalize Test

The Random Data and Normalize Test verifies that the AC/MQ will shift left and right random data 0 to 44₈ places, that the NORM and NORMS instructions operate as specified, and that the processor interrupts after an EAE instruction.

The sequence of testing is as follows:

<u>Test Mnemonic</u>	<u>Operation(s) Tested</u>
RANSHF	Generates 4096 pairs of random numbers, 1 for the AC and 1 for the MQ. Each pair of random numbers is shifted left signed (LLSS) 0 to 44 ₈ places, and the results are tested against a table generated by 44 left shift 1 place.
RANRIT	Generates 4096 pairs of random numbers 1 for the AC and 1 for the MQ. Each pair of random numbers is shifted right (LRS) 0 to 44 ₈ places, and the results are tested against a table generated by 44 shift right 1 place.
RANSEQ	Generates 4096 pairs of random numbers 1 for the AC and 1 for the MQ. Each pair of random numbers is used by RANSQ0 to RANSQ8. After each sequence the AC and MQ should equal their starting patterns.
RANSQ0	Bit 0 of AC = bit 17 of MQ. Random numbers are sequenced 1 left signed, 2 right, 2 left signed, 1 right.
RANSQ1	Bit 0 and 1 of AC = bit 16 and 17 of MQ. Sequence is: 2 right signed 4 left signed 4 right 2 left signed
RANSQ2	Bits 0 to 2 of AC = bits 15 to 17 of MQ. Sequence is: 3 left signed 6 right 6 left signed 3 right

<u>Test Mnemonic</u>	<u>Operation(s) Tested</u>
RANSQ3	Bits 0 to 3 of AC = bits 14 to 17 of MQ. Sequence is: 4 right signed 8 left signed 8 right 4 left signed
RANSQ4	Bits 0 to 4 of AC = bits 13 to 17 of MQ. Sequence is: Left 5 signed Right 10 Left 10 signed Right 5
RANSQ5	Bits 0 to 5 of AC = bits 12 to 17 of MQ. Sequence is: Right 6 signed Left 12 signed Right 12 Left 6 signed
RANSQ6	Bits 0 to 6 of AC = bits 11 to 17 of MQ. Sequence is: Left 7 signed Right 14 Left 14 signed Right 7
RANSQ7	Bits 0 to 7 of AC = bits 10 to 17 of MQ. Sequence is: Right 8 signed Left 16 signed Right 16 Left 8 signed
RANSQ8	Bits 0 to 8 of AC = bits 9 to 17 of MQ. Sequence is: Left 9 signed Right 18 Left 18 signed Right 9
NRMLZE	Does NORMS get AC sign = 0 to link
NRMLZ1	Does NORMS get AC sign = 1 to link
NRMLZ2	Will NORM "stop shift" with $AC0 \neq AC1$, $AC0 = 1$, $AC1 = 0$, or $AC0 = 0$, $AC1 = 0$

<u>Test Mnemonic</u>	<u>Operation(s) Tested</u>
NRMLZ3	Does NORM NOT "stop shift" with AC0 = AC1, AC1 = 0, or AC0 = 0, AC1 = 0 or until SC = 77
NRMLZ4	Will NORMS normalize the alternate pattern of 1 and 0 bits for each bit position of the AC and MQ.
NRMLZ5	Will complement bit patterns normalize
INTEST	(1) Will the teleprinter flag cause an interrupt after an EAE NOP (2) Will the teleprinter flag cause an interrupt after an LLS 43 _g places (3) Does the interrupt not occur until the LLS is complete
7. METHODS (Not Applicable)	
8. FORMAT (Not Applicable)	
9. EXECUTION TIME (Not Applicable)	
10. PROGRAM	
10.1 Core Map (None)	
10.2 Dimension List (None)	
10.3 Macro, Parameter, and Variable Lists (None)	

10.4 Program Listing

10.4.1 Symbol Table

ACCOMK	4700	EAERMO	102	LLSTS2	3342
ACEND	34	EAESLK	436	LLSTS3	3465
ACLMQ	607	ENDSHF	3267	LLSTS4	3526
ACLMQE	635	ERCONT	1070	LLSTS5	3575
ACONEQ	722	ERLOOP	1034	LLSTS6	3656
ACORMQ	544	ERROR	1043	LNKALS	2721
ACSTRT	30	EXT	520000	LRANEX	5057
ALSERR	3175	FIVE6	1225	LRANLP	5046
ALSLNK	2663	FOUR1	1212	LRSERR	4630
ALSMOT	2760	FOUR3	1223	LRSER1	4652
ALSZER	2604	FOUR4	1222	LRSEQ	4566
ALS01	2626	FOUR5	1221	LRSTS1	4016
BIT0	1227	HDR1	1274	LRSTS2	4050
BIT1	1230	HDR2	1305	LRSTS3	4122
BIT10	1241	HDR3	1321	LRSTS4	4175
BIT11	1242	HDR4	1336	LRSTS5	4236
BIT12	1243	HDR5	1355	LRSTS6	4313
BIT13	1244	HSALS	3114	LRSTS7	4373
BIT14	1245	HSALSE	3134	LRSTS8	4450
BIT15	1246	HSALSL	3127	LRST5L	4254
BIT16	1247	INDAT	1410	LRST6L	4333
BIT17	1250	INTEST	6272	LRST7E	4417
BIT2	1231	INTS1	6314	LRST7L	4405
BIT3	1232	INTS2	6346	LRST8E	4476
BIT4	1233	INTS2E	6403	LRST8L	4463
BIT5	1234	INTS2L	6330	MIN5	1166
BIT6	1235	KALL7	1273	MIN6	1167
BIT7	1236	KALS01	2732	MQCOMK	4701
BIT8	1237	KLLSS	5537	MQEND	35
BIT9	1240	KLLSS1	3767	MQSTRT	31
CHARK	1170	KLRS	5536	MQ1TAC	213
COMBIT	1226	K18	1206	MTATAB	2061
COMMA	3307	LKEND	36	NBIT0	1251
COMPMQ	666	LKSTRT	32	NBIT1	1252
CRCODE	1373	LLSACT	3406	NBIT10	1263
CRLF	1145	LLSERR	3745	NBIT11	1264
DCPCNT	6436	LLSSEQ	4527	NBIT12	1265
DCPDIG	6435	LLSSER	3770	NBIT13	1266
DCPNUM	6437	LLSSEX	3617	NBIT14	1267
DCPTAB	1635	LLSSL1	3606	NBIT15	1270
EAEABS	754	LLSSL2	3671	NBIT16	1271
EAECAC	146	LLSSX2	3703	NBIT17	1272
EAECLO	164	LLSTS1	3310	NBIT2	1253

NBIT3	1254	RRANLP	5151	TEM	6432
NBIT4	1255	RRSTAY	5212	TEMY	6434
NBIT5	1256	SALSRP	3244	TEMY1	6431
NBIT6	1257	SAVERR	1205	THREE	1220
NBIT7	1260	SCCOMK	4702	THREE4	1224
NBIT8	1261	SCEND	37	THREE7	1213
NBIT9	1262	SCERR	2520	TIN	101746
NCT	2033	SCSTRT	33	TSP	101720
NDSETU	1021	SCT000	2355	TSR	101773
NIOT	2043	SCT001	2373	TSR1	2003
NOPAC	125	SCT003	2411	TTAB	10
NOPAC1	241	SCT007	2427	TWORD	101627
NOPLK1	502	SCT017	2445	TWORDZ	101572
NOPLNK	343	SCT037	2463	TW040	1201
NOPM0	266	SCT040	2337	TW060	1202
NOPM01	314	SCT060	2321	TYABS	1525
NOPSC	2225	SCT070	2303	TYALS	1566
NOPSC1	2555	SCT074	2265	TYALSQ	1561
NORMER	6216	SCT076	2246	TYCLA	1426
NORMSE	6244	SCT077	2501	TYCLQ	1432
NRMLZE	5652	SCTST1	2200	TYCMQ	1436
NRMLZ1	5704	SEQCOM	5452	TYCOR	1367
NRMLZ2	5737	SETLLS	5025	TYCSC	1531
NRMLZ3	6000	SETLRS	5131	TYDELE	1151
NRMLZ4	6054	SETUP	100	TYINCO	1371
NRMLZ5	6143	SEVEN	1200	TYINTE	1400
NSNERR	3077	SEVFIV	1217	TYLACQ	1456
NXTSEQ	5417	SEVNTY	1210	TYLACS	1535
OCL	2046	SEVN4	1211	TYLLS	1462
OCS	2047	SEVONE	1216	TYLLSS	1466
OCU	2045	SEVSEV	1203	TYLMQ	1521
ONESEV	1215	SEVSIX	1204	TYLRS	1472
OTY	2007	SGNSHF	3042	TYLRSS	1502
PASSK	5535	SHFBUF	5540	TYNOP	1422
QONEAC	410	SIMALS	3216	TYNORM	1545
RANGEN	5522	SIXONE	1214	TYNRMS	1374
RANNO	5533	SIXTY	1207	TYPATR	1365
RANRIT	5112	SPACE3	1350	TYPECN	1074
RANSEQ	5214	SPACE4	1352	TYPLS1	1555
RANSHF	5002	SVCHAR	1172	TYRDSQ	1506
RANSQ0	5237	SVMASK	5450	TYRES	1516
RANSQ1	5253	SVSIGN	5446	TYRMQ	1442
RANSQ2	5267	SWITCH	40	TYSCER	1541
RANSQ3	5303	SWTCHS	60	TYSIMR	1476
RANSQ4	5317	TAB	101730	TYSLK	1446
RANSQ5	5333	TBC	6433	TYSMQ	1452
RANSQ6	5347	TCALL	1122	TYSSC	1551
RANSQ7	5363	TCR	101703	TYSTRT	1420
RANSQ8	5377	TCRRRR	1715	TYT	101730
RESULT	3245	TCRSSS	1712	TY1	101643
RLSTAY	5110	TCTWO	1131	TY1A	1647
RL6	2026	TDIGIT	101755	TY1BBB	2050
RRANEX	5161	TDIGT1	1771	TY2	1671
				TY3	101673

TTAB 10
ACSTRT 30
MQSTRT 31
LKSTRT 32
SCSTRT 33
ACEND 34
MQEND 35
LKEND 36
SCEND 37
SWITCH 40
SWCHS 60
SETUP 100
EAERMQ 102
NOPAC 125
EAECAC 146
EAELQ 164
MQ1TAC 213
NOPAC1 241
NOPMQ 266
NOPMQ1 314
NOPLNK 343
QONEAC 410
EAESLK 436
NOPLK1 502
ACORMQ 544
ACLMQ 607
ACLMQE 635
COMPMQ 666
ACONEQ 722
EAEABS 754
NDSETU 1021
ERROR 1043
ERLOOP 1054
ERCONT 1070
TYPECN 1074
TCALL 1122
TCTWO 1131
CRLF 1145
TYDELE 1151
MIN5 1166
MIN6 1167
CHARK 1170
SVCHAR 1172
SEVEN 1200
TWO40 1201
TWO60 1202
SEVSEV 1203
SEVSIX 1204
SAVERR 1205
K18 1206
SIXTY 1207
SEVNTY 1210
SEVN4 1211
FOUR1 1212
THREE7 1213
SIXONE 1214

ONESEV 1215
SEVONE 1216
SEVFIV 1217
THREE 1220
FOUR5 1221
FOUR4 1222
FOUR3 1223
THREE4 1224
FIVE6 1225
COMBIT 1226
BIT0 1227
BIT1 1230
BIT2 1231
BIT3 1232
BIT4 1233
BIT5 1234
BIT6 1235
BIT7 1236
BIT8 1237
BIT9 1240
BIT10 1241
BIT11 1242
BIT12 1243
BIT13 1244
BIT14 1245
BIT15 1246
BIT16 1247
BIT17 1250
NBIT0 1251
NBIT1 1252
NBIT2 1253
NBIT3 1254
NBIT4 1255
NBIT5 1256
NBIT6 1257
NBIT7 1260
NBIT8 1261
NBIT9 1262
NBIT10 1263
NBIT11 1264
NBIT12 1265
NBIT13 1266
NBIT14 1267
NBIT15 1270
NBIT16 1271
NBIT17 1272
KALL7 1273
HDR1 1274
HDR2 1305
HDR3 1321
HDR4 1336
SPACE3 1350
SPACE4 1352
HDR5 1355
TYPATR 1365
TYCOR 1367

TYINCO 1371
CROODE 1373
TYNRMS 1374
TYINTE 1400
INDAT 1410
TYSTRT 1420
TYNOP 1422
TYCLA 1426
TYCLQ 1432
TYCMQ 1436
TYRMQ 1442
TYSLK 1446
TYSMO 1452
TYLACQ 1456
TYLLS 1462
TYLLSS 1466
TYLRS 1472
TYSIMR 1476
TYLRSS 1502
TYRDSQ 1506
TYRES 1516
TYLMQ 1521
TYABS 1525
TYCSC 1531
TYLACS 1535
TYSCER 1541
TYNORM 1545
TYSSC 1551
TYPLS1 1555
TYALSQ 1561
TYALS 1566
DCPTAB 1635
TY1A 1647
TY2 1671
TCRSSS 1712
TCRRRR 1715
TDIGT1 1771
TSR1 2003
OTY 2007
RL6 2026
NCT 2033
NJOT 2043
OCU 2045
OCL 2046
OCS 2047
TY1888 2050
MTATAB 2061
SCTST1 2200
NOPSC 2225
SCT076 2246
SCT074 2265
SCT070 2303
SCT060 2321
SCT040 2337
SCT000 2355
SCT001 2373

SCT003	2411	LRSTS3	4122	SVSIGN	5446
SCT007	2427	LRSTS4	4175	SVMASK	5450
SCT017	2445	LRSTS5	4236	SEQCOM	5452
SCT037	2463	LRST5L	4254	RANGEN	5522
SCT077	2501	LRSTS6	4313	RANNO	5533
SCERR	2520	LRST6L	4333	PASSK	5535
NOPSC1	2555	LRSTS7	4373	KLRS	5536
ALS±ER	2604	LRST7L	4405	KLLSS	5537
ALS01	2626	LRST7E	4417	SHFBUF	5540
ALSLNK	2663	LRSTS8	4450	NRMLZE	5652
LNKALS	2721	LRST8L	4463	NRMLZ1	5704
KALS01	2732	LRST8E	4476	NRMLZ2	5737
ALSMQT	2760	LLSSEQ	4527	NRMLZ3	6000
SGNSHF	3042	LRSEEQ	4566	NRMLZ4	6054
NSNERR	3077	LRSERR	4630	NRMLZ5	6143
HSALS	3114	LRSER1	4652	NORMER	6216
HSALSL	3127	ACCOMK	4700	NORMSE	6244
HSALSE	3134	MQCOMK	4701	INTEST	6272
ALSERR	3175	SCCOMK	4702	INTS1	6314
SIMALS	3216	RANSHF	5002	INTS2L	6330
SALSRP	3244	SETLLS	5025	INTS2	6346
RESULT	3245	LRANLP	5046	INTS2E	6403
ENDSHF	3267	LRANEX	5057	TEMY1	6431
COMMA	3307	RLSTAY	5110	TEM	6432
LLSTS1	3310	RANRIT	5112	TBC	6433
LLSTS2	3342	SETLRS	5131	TEMY	6434
LLSACT	3406	RRANLP	5151	DCPDIG	6435
LLSTS3	3465	RRANEX	5161	DCPCNT	6436
LLSTS4	3526	RRSTAY	5212	DCPNUM	6437
LLSTS5	3575	RANSEQ	5214	TWORDZ	101572
LLSSL1	3606	RANSQ0	5237	TWORD	101627
LLSSEX	3617	RANSQ1	5253	TY1	101643
LLSTS6	3656	RANSQ2	5267	TY3	101673
LLSSL2	3671	RANSQ3	5303	TCR	101703
LLSSX2	3703	RANSQ4	5317	TSP	101720
LLSERR	3745	RANSQ5	5333	TAB	101730
KLLSS1	3767	RANSQ6	5347	TYT	101730
LLSSER	3770	RANSQ7	5363	TIN	101746
LRSTS1	4016	RANSQ8	5377	TDIGIT	101755
LRSTS2	4050	NXTSEQ	5417	TSR	101773
				EXT	520000

10.4.2 EAE Set-Up Test

/EAE SET UP DIAGNOSTIC

/START AT 100
/PROCESSOR HALTS AT 101 WITH MQP1'S
/OR START AT 102

/SW0 = DELETE ERROR TYPEOUTS
/SW1 = HALT AFTER EACH EAE OPERATION
/SW2 = REPEAT LAST EAE OPERATION
/SW3 = HALT AFTER EACH EAE SEQUENCE
/SW4 = REPEAT EACH EAE SEQUENCE
/SW5 = 0-REPEAT SET UP TEST OR SCA AND SHIFT TESTS
/SW5 = 1-CYCLE SET UP AND SC AND SHIPT TEST

/CAL SUBROUTINE

20/

20	/20 IN CASE CAL 1
LAC 20	/GET ADDRESS
DAC 0	/SAVE
LAC .+4	/RSTORE 20
DAC 20	
LAC 0	
HLT	/HIT DISPLAY
20	/WILL CAL IF CONTINUE

/AC, MQ, LINK AND SC FOR TYPEOUTS

30/

ACSTRT,	0
MQSTRT,	0
LKSTRT,	0
SCSTRT,	0
ACEND,	0
MQEND,	0
LKEND,	0
SCEND,	0

/ROUTINES THAT TEST REPEAT AND STOP
/STOP AFTER MINOR LOOP (SW1) AND REPEAT MINOR LOOP (SW2)

```
SWITCH,      JMP .
              LAS
              AND BIT1
              SNA                /MINOR LOOP HALT?
              JMP .+3            /NO
              LAC I SWITCH
              HLT
              LAC I SWITCH
              DAC .+7
              ISZ SWITCH
              LAS
              AND BIT2
              SZA                /REPEAT LOOP?
              JMP I .+2           /YES
              JMP I SWITCH       /CONTINUE IN SEQUENCE
              Ø
```

/STOP AFTER MAJOR LOOP (SW3) AND REPEAT MAJOR LOOP (SW4)
SWTCHS.

```
JMP .
LAS
AND BIT3
SNA                /MAJOR LOOP HALT?
JMP .+3            /NO
LAC I SWTCHS
HLT
LAC I SWTCHS
DAC SWTCHS-1
ISZ SWTCHS
LAS
AND BIT4
SNA                /REPEAT MAJOR LOOP?
JMP I SWTCHS       /CONTINUE
JMP I SWTCHS-1     /REPEAT LOOP
```

/DOES EAE - OR THE MQ TO AC READ 0'S
/MQ SHOULD BE ZERO FROM START KEY
100/

SETUP, CMQ
 HLT

EAERMQ, CLA 4000 /CLEAR LINK
 DAC MQSTRT
 DAC ACSTRT
 EAF 2 /OR MQ 1'S TO AC
 DAC ACEND
 SNA
 JMP .+11
 JMS ERROR
 TYRMQ
 HDR2
 ACSTRT 600000
 MQSTRT 600000
 TYRMQ
 ACEND 600000
 0
 JMS SWITCH
 FAERMQ
 LAC NBIT16
 DAC CHARK /SET END TEST K

/DOES EAE NOP CLEAR THE AC?

NOPAC, CLC 4000 /CLEAR LINK
 DAC ACSTRT /AC AT START
 AND KALL7 /MAKE MB=1+S BEFORE
 EAE
 DAC ACEND /AC AT END
 CMA
 SNA /AC ALTERED
 JMP .+10 /NO
 JMS ERROR
 TYNOP
 HDR1
 ACSTRT 600000 /TYPE CONTENTS OF
 TYNOP /TYPE TEXT
 ACEND 600000 /TYPE CONTENTS OF
 0
 JMS SWITCH /REPEAT SET
 NOPAC /LOOP TO HERE

/DOES EAE AND CLR AC BIT CLR THE AQ?

```
EAECAC,      CLC 4000          /CLEAR LINK
              EAE 1000          /SHOULD CLEAR AC
              DAC ACEND
              SNA
              JMP .+10
              JMS ERROR
              TYCLA
              HDR1
              ACSTRT 600000
              TYCLA
              ACEND 600000
              0
              JMS SWITCH
              EAECAC
```

/DOES CLQ CLFAR THE MQ

```
EAECLO,      CLC 4000
              DAC MQSTRT
              EAE 4              /SET MQ TO 1'S
              CLA
              DAC ACSTRT
              CLQ              /CLEAR THE MQ
              DAC ACEND
              CLA
              EAE 2            /OR MQ 1'S TO AC
              DAC MQEND
              SNA              /READ 0'S BACK?
              JMP .+12
              JMS ERROR
              TYCLO
              HDR2
              ACSTRT 600000
              MQSTRT 600000
              TYCLO
              ACEND 600000
              MQEND 600000
              0
              JMS SWITCH
              EAECLO          /REPEAT SET
                               /START OVER
```

/DOES MQ COMPLIMENT FROM 0'S TO 1'S
/AND MQ 1'S TO AC

```
MQ1TAC,      CLA 4000
              DAC ACSTRT
              DAC MQSTRT
              CLQ 4           /CLEAR THE MQ AND COMPLIMENT
              DAC ACEND
              CLA
              EAE 2          /OR THE MQ TO AC
              DAC MQEND
              CMA
              SNA
              JMP .+12
              JMS ERROR
              TYCMQ
              HDR2
              ACSTRT 600000
              MQSTRT 600000
              TYCMQ
              ACEND 600000
              MQEND 600000
              0
              JMS SWITCH
              MQ1TAC
```

/DOES EAE-NOP WITH MQ=1'S ALTER THE AC

```
NOPAC1,      CLA 4000
              DAC ACSTRT
              CLC
              DAC MQSTRT
              CLQ 4           /SPT MQ TO ONES
              AND KALL7       /MAKE MB TO 1-S
              EAE             /NOP
              DAC ACEND
              CMA
              SNA             /ONES FROM MQ TO AC?
              JMP .+11
              JMS ERROR
              TYNOP
              HDR2
              ACSTRT 600000
              MQSTRT 600000
              TYNOP
              ACEND 600000
              0
              JMS SWITCH
              NOPAC1
```

/DOES EAE NOP WITH MQ=1'S ALTER THE MQ

```

NOPMQ,      CLA 4000
             CLO 4           /SPT MQ TO 1'S
             AND KALL7       /MAKE MB TO 1+S BEFORE
             EAE             /NOP
             DAC ACEND
             CLA
             EAE 2
             DAC MQEND
             CMA
             SNA             /MQ STILL 1'S?
             JMP .+12
             JMS ERROR
             TYNOP
             HDR2
             ACSTRT 600000
             MQSTRT 600000
             TYNOP
             ACEND 600000
             MQEND 600000
             0
             JMS SWITCH
             NOPMQ
    
```

/DOES NOP WITH AC=1'S ALTER MQ

```

NOPMQ1,     CLA 4000
             DAC MQSTRT
             CLO
             CLC
             DAC ACSTRT
             AND KALL7       /MAKE MB TO 1S BEFORE
             EAE             /NOP
             DAC ACEND
             LACQ            /GET MQ TO AC
             DAC MQEND
             SNA             /ANY 1'S IN MQ
             JMP .+12
             JMS ERROR
             TYNOP
             HDR2
             ACSTRT 600000
             MQSTRT 600000
             TYNOP
             ACEND 600000
             MQEND 600000
             0
             JMS SWITCH
             NOPMQ1
    
```

/DOES NOP ALTER THE LINK
/AC 0'S MQ 0'S, AC 1'S MQ 1'S

```

NOPLNK,      CLQ
              DZM ACSTRT
              DZM MQSTRT
              DZM LKSTRT
              LAC LKSTRT
              RAR
              LAC ACSTRT      /SET LINK FOR TEST
              AND KALL7      /MAKE MB TO ONES BEFORE
              EAE            /NOP
              GLK
              DAC LKEND
              SAD LKSTRT      /LINK ALTERED;
              JMP .+12
              JMS ERROR
              TYNOP
              HDR3
              LKSTRT 700000    /ZPRO SUPPRESS CONTENTS
              ACSTRT 600000
              MQSTRT 600000
              TYNOP
              LKEND 700000
              0
              JMS SWITCH
              NOPLNK+4
              LAC LKSTRT
              ISZ LKSTRT
              SNA            /CHECKED L=0 AND I=1?
              JMP NOPLNK+4
              LAC ACSTRT
              SZA            /CHECKED FOR AC=1'S
              JMP EAESLK      /Y=S
              CLQ 4          /SET MQ TO 1'S
              CLC
              DAC ACSTRT      /AC START =1'S
              DAC MQSTRT
              DZM LKSTRT      /LINK START=0
              JMP NOPLNK+4

```

/DOES MQ TO AC ALL 1'S WITH AC=1'S
GONEAC,

```

CLC
DAC ACSTRT
DAC MQSTRT
CLQ 4          /SET MQ TO 1'S
              /MQ 1'S TO AC 1'S
OMQ
DAC ACEND
CMA
SNA            /AC STAY 1'S
JMP .+11
JMS ERROR
TYRMQ
HDR2
ACSTRT
600000
600000
MQSTRT

```

TYRMO
ACEND
600000
Ø
JMS SWITCH
QONEAC

/LINK SET TO 1 AND TO ZERO?

```
EAESLK,      DZM LKSTRT      /START LINK 0 TO 1
              DZM MQSTRT      /MO 0'S
              CLQ
              LAC BIT0        /400000
              DAC ACSTRT
              LAC LKSTRT      /SPT LINK INITIAL
              RAR
              LAC ACSTRT
              EAF 20000
              DAC ACEND
              GLK
              DAC LKEND
              RTR
              SAD ACSTRT      /LINK SAME AS START?
              SKP
              JMP .+4          /ERROR
              LAC ACEND
              SAD ACSTRT      /AC ALTERED?
              JMP .+13
              JMS ERROR
              TYSLK
              HDR3
              LKSTRT 700000
              ACSTRT 600000
              MQSTRT
              TYSLK
              LKEND 700000
              ACEND 600000
              0
              JMS SWITCH      /LOOP SET?
              EAESLK+5
              IS7 LKSTRT      /NEXT PASS LINK 1 TO ZERO
              LAC ACSTRT
              DZM ACSTRT
              SZA
              JMP EAESLK+5
```


/DOES NOP ALTER MQ=0'S WITH L=1

NOPLK1,	DZM ACSTRT	/START AC 0'S
	DZM MQSTRT	/MQ 0'S
	CLD	
	LAC RIT17	/I=LINK
	DAC LKSTRT	
	RAR 4000	/CLR LINK, SET LINK
	AND KALL7	/MAKE MR TO ONES BEFORE
	EAF	/NOP
	DAC ACEND	
	GLK	
	DAC LKEND	
	LACQ	
	DAC MQEND	
	SNA	/MQ STILL ZERO'S
	LAC ACEND	
	SNA CLA-OPR	/AC STILL ZERO'S
	LAC LKEND	
	SZA	/LINK STILL 1
	JMP .+14	
	JMS ERROR	
	TYNOP	
	HDR3	
	LKSTRT 700000	
	ACSTRT 600000	
	MQSTRT 600000	
	TYNOP	
	LKEND 700000	
	ACEND 600000	
	MQEND 600000	
	0	
	JMS SWITCH	/CHECK MINOR LOOP SW
	NOPLK1	
	JMS SWTCHS	/MAJOR LOOP SET?
	NOPAC	/START NOP THE AC

/WILL AC TO MQ TO AC ALL PATTERNS
 /WITH MQ = LAST PATTERN AND LINK = 1

```

ACLMQ,      DZM ACSTRT      /START AC 0'S
            DZM MQSTRT      /MQ 0'S
            CLQ
            LAC BIT17        /LINK 1
            DAC LKSTRT
            STL              /SET LINK
            LAC ACSTRT      /GET NEXT CONSTANT
            LMQ              /MQ TO 0'S, AC 1'S TO MQ
            DAC ACEND        /SAVE AC RESULT
            GLK
            DAC LKEND        /SAVE LINK RESULT
            LACQ            /GET MQ
            DAC MQEND
            SAD ACSTRT      /MQ = AC AT START?
            SKP
            JMP ACLMQE      /MQ ERROR
            LAC LKEND
            SNA              /LINK=1 AT END?
            JMP ACLMQE      /LINK ERROR
            LAC ACEND
            SAD ACSTRT      /AC END = AC START?
            JMP .+22
ACLMOE,     JMS ERROR
            TYLMQ
            HDR3
            LKSTRT 700000
            ACSTRT 600000
            MQSTRT 600000
            TYLMQ
            LKSTRT 700000
            ACSTRT 600000
            ACSTRT 600000
            TYCOR
            TYLACQ
            LKEND 700000
            ACEND 600000
            MQEND 600000
            TYINCO
            0
            LAC MQEND
            DAC MQSTRT      /NEW MQ START
            JMS SWITCH      /REPEAT SET?
            ACLMQ+5
            ISZ ACSTRT      /TO 777777?
            JMP ACLMQ+5
            JMS SWCHS
            ACLMQ
  
```

/DOES THE MQ COMPLIMENT ALL PATTERNS

```

COMPMQ,      DZM ACSTRT
              LAC ACSTRT          /GET NEXT PATTERN
              DAC MQSTRT
              LMQ 20000          /AC TO MQ, AC0 TO L
              CMQ                /-MQ
              DAC ACEND          /SAVE AC RESULT
              LACQ              /GET MQ
              DAC MQEND
              CMA                //MQ
              SNA ACSTRT        /-MQ = AC START?
              LAC ACEND
              SNA ACSTRT        /ACEND = AC START?
              JMP .+12
              JMS ERROR
              TYCMQ
              HDR2
              ACSTRT 600000
              MQSTRT 600000
              TYCMQ
              ACEND 600000
              MQEND 600000
              0
              JMS SWITCH
              COMPMQ+1
              ISZ ACSTRT
              JMP COMPMQ+1
              JMS SWITCHS
              COMPMQ

```

/DOES AC TO MQ ALL 1'S WITH MQ=1'S

```

ACONEQ,      CLC
              DAC MQSTRT
              DAC ACSTRT
              CLO 4             /SET MQ=1'S
              FAF 2000         /AC 1'S TO MQ 1'S
              DAC ACEND
              LACQ
              DAC MQEND
              CMA
              SNA              /MQ STAY 1'S
              JMP .+16
              JMS ERROR
              TYSMQ
              HDR2
              ACSTRT
              600000
              600000
              MQSTRT
              TYSMQ
              ACEND
              600000
              600000
              MQEND
              0
              JMS SWITCH
              ACONEQ

```

/DOES ABS GET ABSOLUTE AC
/AND NOT DISTURB LINK=1 OR 0

```
DZM ACSTRT          /START AC 0'S
LAC BIT17           /LINK 1
DAC LKSTRT
LAC LKSTRT
RAR                /SPT LINK
LAC ACSTRT          /GET AC START
ABS                /ABSOLUTE AC
DAC ACEND           /SAVE RESULT
GLK
DAC LKEND
SAD LKSTRT          /LINK SAME?
SKP                /YES
JMP .+6             /ERROR, LINK CHANGED
LAC ACSTRT
SPA                /Am POSITIVE AT START?
CMA                /No, SHOULD BE POS. ABS
SAD ACEND           /RESULT AC OK?
JMP .+12            /YES
JMS ERROR           /ABS ERROR LINK OR AC
TYABS
HDR3
LKSTRT 700000
ACSTRT 600000
TYABS
LKEND 700000
ACEND 600000
0
JMS SWITCH
EAEABS+3
ISZ ACSTRT
SKP
JMP NDSETU
LAC LKSTRT
CMA
AND BIT17
DAC LKSTRT
JMP EAEABS+3
```

NDSETU.

```
JMS SWITCHS        /TEST REPEAT MAJOR
EAEABS
LAS
AND BIT6
SNA
JMP .+6
LAW 57
TY1
ISZ CHARK
JMP .+4
JMS CRLF
LAC NBIT16
DAC CHARK
LAS
AND BIT5
SNA
JMP NOPAC          /REPEAT ALL SET?
                   /CYCLE SET UP TEST
```

JMP SGT076

/CYCLE SET UP AND SHIFT

START

/EAE ERROR TYPEOUT ROUTINE
/GENERAL PURPOSE
/LINKS TYPTX AND ALL TYPE CONTENTS

/AC=0 IS END OF TYPEOUT
/AC NOT = 0 AND POSITIVE IS TYPETEXT
/AC - AND BIT 1=0 IS CR, LF TYPE CONTENTS
/AC - AND BIT 1=1 IS TYPE CONTENTS
/AC - AND BIT 2=0 IS NO ZERO SUPPRESS
/AC - AND BIT 2=1 IS ZERO SUPPRESS
/AC - AND BIT 3=0 IS ZERO SUPPRESS5
/AC - AND BIT 3=1 IS ZERO SUPPRESS4

```
ERR0R,      JMP
             LAS
             SPA
             JMP TYDELE
             JMS CRLF
             LAC .+3
             DAC SAVERR
             JMP TYPECN          /CR LF TYPE CONTENTS ERR0R
ERR0R
ERLOOP,     LAC I ERROR          /GET NEXT TYPE CONSTANT
             DAC SAVERR        /FOR INDIRECTS
             ISZ ERROR
             SZA                /END OF MESSAGE?
             JMP ERCONT        /NO
             LAS                /GET SWITCHES
             AND BIT7
             SNA                /DELETE HALT?
             HLT                /ERROR HALT
             TSF
             JMP .-1          /WAIT FLAG
ERCONT,     JMP I ERROR          /EXIT ERROR ROUT.
             SPA                /TYPE TEXT INDICATED?
             JMP TYPECN        /NO, TYE CONTENTS
             TSR
             JMP ERLOOP
```

/TYPE CONTENTS ROUTINES

TYPECN,	AND BIT1	/CARRIAGE RETURN INDICATED
	SNA	/YES
	JMS CRLF	
	LAC SAVERR	
	AND BIT2	
	SNA	/SUPPRESS ZERO SET?
	JMP TCALL	/NO, TYPE ALL
	LAC SAVERR	
	AND BIT3	
	SZA	/SUPPRESS 4 0'S SET?
	JMP TCTWO	/YES
	LAC I SAVERR	
	AND .+11	
	SZA	/UPPER 5 CHAR = 0
	JMP TCALL	/NO, TYPE ALL
	LAC I SAVERR	
	CLL RTR-OPR	
	RTR	
	TWORD	
	1	
	JMP TCALL+3	/SPACE 3
	777770	


```
TCALL,      LAC I SAVERR
             TWORD                /TYPE 6 OCTAL
             6
             LAW SPACE3
             TSR                    /OUTPUT 3 SPACES
             JMP ERLOOP
             777700
TCTWO,      LAC I SAVERR
             AND .-2
             SZA                    /FIRST 4 CHARACTERS 0
             JMP TCALL              /NO, TYPE WHOLE WORD
             LAC I SAVERR
             CLL RTR-OPR           /POSITION LS 2
             RTR                    /TO UPPER 2
             RTR                    /FOR TYPEOUT ROUT
             RAR
             TWORD                /TYPE UPPER 2 CHAR
             2
             JMP TCALL+3           /SPACE 3

CRLF,       JMP .
             LAW CRCODE
             TSR
             JMP I CRLF
```

```
TYDELE,      LAC I ERROR
              ISZ ERROR
              SZA                /REACHED END OF MESS.
              JMP TYDELE         /NO
              LAS
              AND BIT7
              SNA                /RING BELL SET?
              JMP I ERROR       /NO, EXIT
              LAW 51
              TY1
              TSF                /WAIT FLAG
              JMP .-1
              JMP I ERROR
```

```
MIN5,        777773
MIN6,        777772
CHARK,       0
             0
SVCHAR,      0
             0
             0
             0
             0
             0
             0
SEVEN,       7
TWO40,      240
TWO60,      260
SEVSEV,     77
SEVSIX,     76
SAVERR,     0
K18,        777756
SIXTY,      60
SEVENTY,    70
SEVN4,      74
FOUR1,      41
THREE7,     37
SIXONE,     61
ONESEV,     17
SEVONE,     71
SEVFIV,     75
THREE,      3
FOUR5,      45
FOUR4,      44
FOUR3,      43
THREE4,     34
FIVE6,      56
COMBIT,     252525
```

/BIT AND NO BIT CONSTANTS

BIT0,	400000
BIT1,	200000
BIT2,	100000
BIT3,	40000
BIT4,	20000
BIT5,	10000
BIT6,	4000
BIT7,	2000
BIT8,	1000
BIT9,	400
BIT10,	200
BIT11,	100
BIT12,	40
BIT13,	20
BIT14,	10
BIT15,	4
BIT16,	2
BIT17,	1
NBIT0,	377777
NBIT1,	577777
NBIT2,	677777
NBIT3,	737777
NBIT4,	757777
NBIT5,	767777
NBIT6,	773777
NBIT7,	775777
NBIT8,	776777
NBIT9,	777377
NBIT10,	777577
NBIT11,	777677
NBIT12,	777737
NBIT13,	777757
NBIT14,	777767
NBIT15,	777773
NBIT16,	777775
NBIT17,	777776
KALL7,	777777

```
/MESSAGE CONSTANTS
/ERROR TYPEOUT HEADERS
/AC CONTENTS
HDR1,          TEXT /
      C(AC)
START /
```

```
/AC AND MQ
```

```
HDR2,          TEXT /
      C(AC)     C(MQ)
START /
```

```
/LINK AC AND MQ
```

```
HDR3,          TEXT /
      L        C(AC)  C(MQ)
START /
```

```
/SC AC
```

```
HDR4,          TEXT /
      SC       C(AC)
START /
```

```
/3 SPACES
```

```
SPACE3,       TEXT / /
```

```
/4 SPACES
```

```
SPACE4,       TEXT /
/
```

HDR5, TEXT /
L C(AC) C(MQ)/

TYPATR, TEXT /PAT/

TYCOR, TEXT /COR/

TYINCO, TEXT /INCO/

CRCODE, TEXT /
/

TYNRMS, TEXT /
NORMS /

TYINTE, TEXT /
NO PROGRAM INTERRUPT/

INDAT, TEXT /
INTERRUPT DATA ERROR/
TYSTRT, TEXT /START/

/OPERATION TYPEOUTS
/EAE NO OPERATION

TYNOP, TEXT /
EAENOP /

/EAE CLA
TYCLA, TEXT /
EAECLA /

/CLEAR MQ
TYCLO, TEXT /
CLO /

/COMPLIMENT Q
TYCMQ, TEXT /
CMQ /

/OR MQ TO AC
TYRMQ, TEXT /
ORMQAC /

/ACØ TO LINK
TYSLK, TEXT /
ACØTOL /

/OR AC TO MQ
TYSMQ, TEXT /
ORACMQ /

/LOAD AC WITH MQ
TYLACQ, TEXT /
LACQ /

/LLS
TYLLS, TEXT /
LLS /

/LLSS
TYLLSS, TEXT /
LLSS /

Digital-7-92-M
Page 46

/LRS
TYLRS, TEXT /
LRS /

/RESULT
TYSIMR, TEXT /
RESULT /

TYLRSS, TEXT /
LRSS /

TYRDSQ, TEXT /
RANDOM DATA SEQUENCED/
TYRES, TEXT /RESULT/

/LOAD MQ WITH AC
TYLMQ, TEXT /
LMQ /

TYABS, TEXT /
ABS /

/CLR SC
TYCSC, TEXT /
CLR SC /

/LACS
TYLACS, TEXT /
LACS /

/SC ERROR
TYSCER, TEXT /
SC ERROR /

/NORM
TYNORM, TEXT /
NORM /

/SET SC
TYSSC, TEXT /
SET SC /

/SC+1
TYPLS1, TEXT /
SC+1 /

/ALS MQ TEST
TYALSQ, TEXT /
ALS MQ TEST/

/ALS
TYALS, TEXT /
ALS /

START

10.4.3 Shift Counter and Basic Shift Test

/SHIFT COUNTER AND
/AC MQ SHIFT TEST
/TAPE 3 OF PDP7 EAE TEST

/SHIFT COUNTER TEST
/UTILIZES NORMALIZE INSTRUCTION
/WITH NO SHIFT TO DATA TEST S.C
/DOES START KEY CLEAR THE SC

2200/

SCTST1,

LAC SEVSEV
DAC SCSTRT
LAC BIT1
NORM -43
HLT
LACS
DAC SCEND
SNA
JMP .+11
JMS ERROR
TYCSC
HDR4
SCSTRT 740000
TYCSC
TYLACS
SCEND 740000
0
JMS SWITCH
SCTST1+6
LAC NBIT16
DAC CHARK

/200000 ALREADY NORMALIZED
/SET SC TO -1 (77)
/PRESS START TO CLR SC
/SC TO AC

/READ SC#0'S TO AC?
/YFS, CONTINUE

/DOES EAE NOP ALTER THE SC

NOPSC, DZM SCSTRT
AND KALL7
EAE
LACS
DAC SCEND
SNA
JMP .+11
JMS ERROR
TYNOP
HDR4
SCSTRT 740000
TYNOP
TYLACS
SCEND 740000
0
JMS SWITCH
NOPSC

/MAKE MB ONES BEFORE
/NOP-
/GPT SC TO AC
/SC STILL ZERO'S

/DOES SC SET TO 76 AND +1 TO 77
SCT076,

LAC SCEND
DAC SCSTRT
LAC BIT17
DAC MQSTRT
LAC BIT1
DAC ACSTRT
NORM -43
DAC ACEND
LACS
DAC SCEND
SAD SEVSEV
JMP .+2
JMS SCERR
JMS SWITCH
SCT076

/NORM 01
/SET SC TO 76+1 TO 77

/DOES SC SET TO 74 AND +1 TO 75
SCT074,

LAC SCEND
DAC SCSTRT
LAC THREE
DAC MQSTRT
LAC BIT1
NORM -41
DAC ACEND
LACS
DAC SCEND
SAD SEVFIV
JMP .+2
JMS SCERR
JMS SWITCH
SCT074

/SC TO 74+1 TO 75
/SAVE FOR ERROR TYPE

/DOES SC SET TO 70 AND +1 TO 71

SCT070, LAC SCEND
 DAC SCSTRT
 LAC SEVEN
 DAC MQSTRT
 LAC BIT1
 NORM -35
 DAC ACEND
 LACS
 DAC SCEND
 SAD SEVONE
 JMP .+2
 JMS SCERR
 JMS SWITCH
 SCT070

/7, SC TO 70 AND +1 TO 71
 /SAVE FOR ERROR TYPE

/WILL SC SET TO 60 AND +1 TO 61

SCT060, LAC SCEND
 DAC SCSTRT
 LAC ONESEV
 DAC MQSTRT
 LAC BIT1
 NORM -25
 DAC ACEND
 LACS
 DAC SCEND
 SAD SIXONE
 JMP .+2
 JMS SCERR
 JMS SWITCH
 SCT060

/NORM 17

/SPT SC TO 60 AND +1 TO 61
 /SAVE FOR ERROR TYPE

/RFA

/WILL SC SET TO 40 AND +1 TO 41

SCT040, LAC SCEND
 DAC SCSTRT
 LAC THREE7
 DAC MQSTRT
 LAC BIT1
 NORM -5
 DAC ACEND
 LACS
 DAC SCEND
 SAD FOUR1
 JMP .+2
 JMS SCERR
 JMS SWITCH
 SCT040

/NORM 37

/20000 ALREADY NORMALIZED
/SPT SC TO 40 AND +1 TO 41

/GPT SC TO AC
 /SAVE FOR ERROR TYPE
 /READ 41 FROM SC TO AC

/YFS

/WILL SC SET TO 0 AND +1 TO 1

```
SCT000,      LAC SCEND
              DAC SCSTRT
              LAC SEVSEV      /NORM 77
              DAC MQSTRT
              LAC BIT1
              NORM +33        /SPT TO 00 +1 TO 01
              DAC ACEND
              LACS
              DAC SCEND
              SAD BIT17
              JMP .+2          /YFS      /SC READ 01?
              JMS SCERR
              JMS SWITCH
              SCT000
```

/WILL SC SET TO 01 AND +1 TO 02

```
SCT001,      LAC SCEND
              DAC SCSTRT
              LAC SEVSIX      /NORM 76
              DAC MQSTRT
              LAC BIT1
              NORM 32         /SPT SC TO 1 +1 TO 2
              DAC ACEND
              LACS
              DAC SCEND
              SAD BIT16
              JMP .+2
              JMS SCERR
              JMS SWITCH
              SCT001
```

/WILL SC SET TO 03 AND +1 TO 04

```
SCT003,      LAC SCEND
              DAC SCSTRT
              LAC SEVN4       /NORM 74
              DAC MQSTRT
              LAC BIT1
              NORM +30        /SPT SC TO 3 +1 TO 4
              DAC ACEND
              LACS
              DAC SCEND
              SAD BIT15
              JMP .+2          /SPT TO AC =4?
              JMS SCERR      /YFS
              JMS SWITCH
              SCT003
```

/WILL SC SET TO 07 AND +1 TO 10
SCT007,

LAC SCEND
DAC SCSTR
LAC SEVNTY
DAC MQSTR
LAC BIT1
NORM +24
DAC ACEND
LACS
DAC SCEND
SAD BIT14
JMP .+2
JMS SCERR
JMS SWITCH
SCT007

/NORM 70

/SPT SC TO 7 +1 TO 10

/SC TO AC = 10?

/YFS

/WILL SC SET TO 17 AND +1 TO 20
SCT017,

LAC SCEND
DAC SCSTR
LAC SIXTY
DAC MQSTR
LAC BIT1
NORM +14
DAC SCEND
LACS
DAC SCEND
SAD BIT13
JMP .+2
JMS SCERR
JMS SWITCH
SCT017

/NORM 60

/SC TO 17+1 TO 20

/SC TO AC = 20?

/YFS

/WILL SC SET TO 37 AND +1 TO 40

SCT037,

LAC SCEND
LAC SCSTR
LAC BIT12
DAC MQSTR
LAC BIT1
NORM -4
DAC ACEND
LACS
DAC SCEND
SAD BIT12
JMP .+2
JMS SCERR
JMS SWITCH
SCT037

/NORM 40

/SPT SC TO 37 +1 TO 40

/SC TO AC = 40?

/YFS

/WILL SC SET TO 77 AND +1 TO 00

```
SCT077,      LAC SCEND
              DAC SCSTRT
              LAC SEVSEV      /NORM 0
              DAC MQSTRT
              LAC RIT1
              NORM -44        /SFT SC TO 77 AND +1 TO 0
              DAC ACEND
              LACS            /GET SC TO AC
              DAC SCEND
              SNA              /SC TO AC = 00?
              JMP .+2         /YFS
              JMS SCERR
              JMS SWITCH
              SCT077
              JMP NOPSC1
```

```

SCERR,      JMP .
            LAC MQSTRT      /GET SC OF NORM
            CMA
            AND SEVSEV      /SHOULD SET SC TO
            DAC MQEND
            TAD RIT17
            AND SEVSEV      /SC SHOULD +1 TO
            DAC LKEND
            JMS ERROR      /TYPE OUT
            TYSCER          /SC ERROR
            SCERR 600000    /ERROR ADDRESS
            HDR4
            SCSTRT 740000  /SC AT START
            SPACE3
            ACSTRT 600000  /AC AT START
            TYNORM
            MQSTRT 740000  /SC PORTION OF NORM
            TYSSC
            MQEND 740000  /SHOULD SET SC TO
            TYPLS1
            LKEND 740000  /SC SHOULD +1 TO
            TYCOR
            TYLACS
            SCEND 740000  /SC TO AC EQUALED
            TYINCO
            SPACE3
            ACEND 600000  /AC AFTER NORM
            0
            JMP I SCERR

```

/DOES EAF NOP ALTER SC = 77

```

NOPSC1,    LAC SEVSEV
            DAC SCSTRT
            LAC RIT1
            NORM -43      /SET SC TO 77
            AND KALL7      /MAKE MB TO ONES BEFORE
            EAF 77        /NOP SHOULD NOT ALTER SC
            LACS          /GET SC TO AC
            DAC SCEND
            SAD SCSTRT    /SC TO AC = 77?
            JMP .+12
            JMS ERROR
            TYNOP
            HDR4
            SCSTRT 740000
            TYNOP
            SCSTRT 740000
            TYLACS
            SCEND 740000
            0
            JMS SWITCH
            NOPSC1
            JMS SWCHS
            SCT076

```

/SHIFT TESTS
/ALS = ACCUMULATOR LEFT SHIFT
/DOES ALS AC = 0'S ALTER THE AC?

```
ALSZER,      DZM ACSTRT
              DZM MQSTRT
              DZM LKSTRT
              DZM SCSTRT
              CLQ 1000
              CLL
              ALS
              DAC ACEND
              GLK
              DAC LKEND
              LACS
              DAC SCEND
              LAC ACEND
              SNA
              SKP
              JMS ALSERR
              JMS SWITCH
              ALSZER+4
```

/CLEAR AC = MQ AND LINK

/DOES ALS 01 AC = 0'S OK

```
ALS01,      IAC BIT17
              DAC SCSTRT
              DZM ACSTRT
              DZM MQSTRT
              DZM LKSTRT
              CLQ
              EAE 1000
              CLL
              ALS 01
              DAC ACEND
              GLK
              DAC LKEND
              LACS
              DAC SCEND
              LAC ACEND
              SNA
              SKP
              JMS ALSERR
              LAC MQSTRT
              LMQ
              JMS SWITCH
              ALS01+6
              LAC MQSTRT
              SZA
              JMP .+5
              CLQ
              DAC MQSTRT
              EAE 4
              JMP ALS01+6
```

/ALS 01
/AC 0'S TO START
/MQ 0'S
/LINK IS ZERO

/SHIFT AC LEFT 1

/LINK FOR TYPEOUTS
/SC FOR TYPEOUTS

/2ND PASS MQ = 1'S

/LINK TO AC 17

/BIT = 0 L=0, BIT = 0 L=1, BIT = 1 L = 0, BIT = 1 L = 1

```

ALSLNK,      DZM ACSTRT          /START AC 0'S
              DZM MQSTRT
              DZM LKSTRT      /LINK START 0
              CLO
              LAC LKSTRT
              RAR              /LINK = 0 OR 1
              LAC ACSTRT
              ALS 01
              DAC AGEND
              GLK
              DAC LKEND
              LACS
              DAC SCEND
              LAC ACSTRT
              RAL
              TAD LKSTRT
              SAD AGEND
              SKP
              JMS ALSERR
              JMS SWITCH
              ALSLNK+3
              LAC LKSTRT
              IS7 LKSTRT
              SNA              /2ND PASS L=1
              JMP ALSLNK+3
              DZM LKSTRT
              LAC ACSTRT
              IS7 ACSTRT
              SNA              /3RD AND 4TH PASS AC=1
              JMP ALSLNK+3

```

/DOES ALS ALTER THE LINK = 1 OR 0

```

LNKALS,      DZM MQSTRT          /MO ALWAYS = 0
              DZM ACSTRT      /START AC=0
              DZM LKSTRT      /LINK START 0
              LAC BIT17
              DAC SCSTRT      /SC = 01
              CLO
              LAC LKSTRT
              RAR              /LINK = 1 OR 0
              LAC ACSTRT      /AC = 0 OR 400000
              ALS 01
              DAC AGEND          /SAVE AC RESULT
              LACS
              DAC SCEND          /SAVE SC RESULT
              GLK
              DAC LKEND
              SAD LKSTRT        /LINK SAME AS STRT?
              SKP              /YRS
              JMS ALSERR
              JMS SWITCH
              LNKALS+5
              LAC LKSTRT

```

1S7 LKSTRT
SNA
JMP LNKALS+5

/2ND AND 4TH PAS 1 = 1

```
LAC ACSTRT
LAC RIT0
SAD ACSTRT /AC 0 AI READY = 1
JMP .+4
DZM LKSTRT /3RD AND 4TH PASS
DAC ACSTRT /AC=400000
JMP LNKALS+5
```

/DOES ALS ALTER THE MQ

```
ALSMOT, DZM MQSTRT /1ST PASSES MQ = 0'S
DZM ACSTRT
DZM LKSTRT
LAC RIT17
DAC SCSTRT /AIS 01 PLACE
LAC LKSTRT /L=1 OR 0
RAR
LAC MQSTRT /MQ = 0'S OR 1'S
LMO /AC = 0'S OR 1'S
LAC ACSTRT
ALS 01
DAC ACEND
GLK
DAC LKEND
LACS
DAC SCEND
LACQ
DAC MQEND
SAD MQSTRT /MQ SAME AS START?
JMP .+14 /YES
JMS ERROR
TYALSQ
HDR3
LKSTRT 700000
ACSTRT 600000
MQSTRT
TYALS
LKEND 700000
ACEND 600000
MQEND 600000
0
JMS SWITCH
ALSMOT+5
```

```

LAC LKSTRT
IS7 LKSTRT          /EVERY OTHER PASS L = 1
SNA
JMP ALSMQT+5
DZM LKSTRT          /NEXT PASS L = 0
LAC ACSTRT
CMA                /AC=0'S, 1'S, 0'S, 1'S
DAC ACSTRT
SZA
JMP ALSMQT+5
LAC MQSTRT          /M0 = 0'S 4 PASSES 1'S 4 PASSES
CMA
DAC MQSTRT
SZA                /MQSTRT BACK TO 0'S?
JMP ALSMQT+5        /No, TEST M1 = 1'S
JMS SWTCHS
ALSZER

```

```

/WILL AC0 GO TO LINK PROPERLY
/IMMEDIATELY FOLLOWING AN ALS LEFT SHIFT
/0-0:0-1,1-0,1-1

```

```

SGNSHF,           DZM LKSTRT          /LK TO 0 FIRST
                  DZM MQSTRT          /TO COMPARE LINK ONLY
                  LAC RIT0
                  DAC ACSTRT          /FIRST AC0=1 GOES TO 0
                  LAC RIT17
                  DAC SCSTRT          /SHIFT=1 0PLACE
                  LAC LKSTRT
                  RAR                /MAKE L=START
                  LAC ACSTRT
                  ALS 01             /AC0=1 GOES TO 0 OR = 0 GOES TO 1
                  FAF 00000          /GET SIGN OF AC
                  DAC ACEND          /SAVE FOR TYPEOUTS
                  GLK
                  DAC LKFND          /SAVE FOR TYPEOUTS
                  SAD MQSTRT        /L=CORRECT RESULT
                  JMP NSNERR        /YES
                  JMS ERROR
                  TYALS
                  SCSTRT 740000
                  TYSLK
                  HDR3
                  LKSTRT 700000
                  ACSTRT 600000
                  TYALS
                  TYSLK
                  LKFND 700000
                  ACEND 600000
                  TYRES
                  0
NSNERR,           JMS SWITCH          /END SCOPE LOOP
                  SGNSHF+6
                  LAC LKSTRT
                  IS7 LKSTRT
                  SNA
                  JMP SGNSHF+6      /THIS PASS L=1
                  DZM LKSTRT
                  LAC RIT1
                  SAD ACSTRT        /TESTED SIGN=1

```

```
JMP HSALS  
DAC ACSTRT  
IS7 MQSTRT  
JMP SGNSHF+6
```

/YES

/WILL ALS SHIFT 1 TO 18 PLACES?
/1ST PASS BIT 2ND PASS NO BIT

HSALS,	DZM MQSTRT	
	LAC BIT17	
	DAC ACSTRT	
	DZM LKSTRT	
	JMS SIMALS	
	LAC K1A	
	DAC 10	
	LAC KALS01	
	DAC HSALSE	
	LAC BIT17	
	DAC SCSTRT	
HSALSL,	LAC LKSTRT	
	RAR	
	LAC MQSTRT	/MO ALTERNATES
	LMQ	/FROM 1'S TO 0'S
	LAC ACSTRT	
HSALSE,	ALS 01	/1 TO 18 PLACES
	DAC ACEND	
	GLK	
	DAC LKEND	
	LACS	
	DAC SCEND	
	SZA	/SM GO TO ZERO?
	JMP .+6	
	LAC LKEND	
	SAD LKSTRT	/WAS LINK ALTERED
	LAC ACEND	
	SAD I SALS RP	/RESULT OF SHIFT OK?
	JMP .+2	
	JMS ALSERR	
	JMS SWITCH	
	HSALSL	
	LAC MQSTRT	
	CMA	/EVEN PASSES MO = 77777
	DAC MQSTRT	
	IS7 HSALSE	/INCREMENT COUNT
	IS7 SALS RP	/ADVANCE RESULT POINTER
	ISZ SCSTRT	/FOR TYPEOUTS SC+1
	ISZ 10	/SHIFT 18 TIMES?
	JMP HSALSL	
	ISZ LKSTRT	/NO BIT PASS L = 1
	LAC ACSTRT	
	CMA	/2ND PASS AC STRT=777776
	DAC ACSTRT	
	SPA	/MADE 2 PASSES?
	JMP HSALB+4	/NO, SHIFT NO BIT
	JMS SWITCHS	
	HSALS	
	JMP LLSTS1	

/ALS INSTRUCTION
/COMMON ERROR TYPEOUT

```
ALSERR,      JMP .  
             JMS ERROR  
             TYALS  
             SCSTRT 740000  
             ALSERR 400000  
             HDR5  
             LKSTRT 500000  
             ACSTRT 600000  
             MQSTRT 600000  
             TYPATR  
             LKEND 500000  
             ACEND 600000  
             TYRES  
             TYLACS  
             SCEND 740000  
             0  
             JMP I ALSERR
```

/SIMULATE ALS OPERATION
/STORES SHIFTS 1 TO 18 PLACES

```
SIMALS,      JMP .  
             LAR RESULT-1  
             DAC 17  
             DAC 15  
             TAD BIT17  
             DAC SALSRP  
             LAC K18  
             DAC 16  
             LAC LKSTRT  
             RAR  
             LAC ACSTRT  
             RAL  
             DAC I 17  
             IS7 16  
             LAC LKSTRT  
             RAR  
             LAC I 15  
             RAL  
             DAC I 17  
             IS7 16  
             JMP .+6  
             JMP I SIMALS
```

/SET RESULT POINTER TO START

```
SALSRP,      0  
RESULT,      0
```

RESULT+22/ /RESERVE 17 SHIFT LOCATIONS

```
ENDSHF,      LAS
              AND BIT6
              SNA                /COMMA AT END,
              JMP .+6            /NO
              LAW 15
              TY1
              ISZ CHARK
              JMP .+4
              JMS CRLF
              LAC NBIT16
              DAC CHARK
              LAS
              AND BIT5
              SNA                /CYCLE BOTH TESTS
              JMP SCT076         /NO, STAY IN SHIFT TEST
              JMP RANSHF         /REPEAT FROM SETUP TEST
COMMA,
START
```


/LLS AND LRS BASIC TESTS
/TAPE 4 OF EAE PDP7 TEST

/LONG LEFT SHIFT

/LLS 01 ALL ZERO'S

```
LLSTS1,      DZM ACSTRT
              DZM MOSTRT
              DZM LKSTRT
              IAC RIT17
              DAC SCSTRT
              CLQ
              CLA CLL-OPR
              LLS 01
              DAC ACEND
              GLK
              DAC LKEND
              LACS
              DAC SCEND
              LACQ
              DAC MOEND
              SNA
              LAC ACEND
              SNA
              LAC LKEND
              SNA
              LAC SCEND
              SNA
              JMP .+2
              JMS LLSERR
              JMS SWITCH
              LLSTS1+5

              /START SCOPE LOOP
              /CIR AC AND LINK

              /MO STILL 0'S?
              /AC STILL 0'S?
              /LINK STILL 0'S?
              /SM GO TO ZERO?
```

/DOES LINK GO TO MQ17 ON AN LLS
/0-0: 1=0, 0-1, 1-1

```
LLSTS2,      DZM MQSTRT
              DZM ACSTRT
              DZM LKSTRT
              LAC BIT17
              DAC SCSTRT
              LAC MQSTRT
              LMO
              LAC LKSTRT
              RAR
              LAC ACSTRT
              LLS 01
              DAC ACEND
              GLK
              DAC LKEND
              LACS
              DAC SCEND
              LACQ
              DAC MQEND
              LAC LKSTRT
              RAR
              LAC MQSTRT
              RAL
              SAD MQEND
              SKP
              JMS LLSERR
              JMS SWITCH
              LLSTS2+5
              LAC LKSTRT
              IS7 LKSTRT
              SNA
              JMP LLSTS2+5
              DZM LKSTRT
              LAC MQSTRT
              IS7 MQSTRT
              SNA
              JMP LLSTS2+5

              /LLS 01
              /2 PASSES = 0  START SCOPE LOOP
              /2 PASSES = 1 (MQ17)
              /L=1 EVERY 2ND PASS
              /AC ALWAYS = 0
              /SAVE RESULTS

              /END SCOPE LOOP

              /2ND OR 4TH PASS?
              /NEXT PASS L = 0
              /MADE WITH MQ17=17
```

/DOES LINK NOT GO TO AC17 ON AN LLS
/DOES MQ0 GO TO AC17 ON AN LLS

```
LLSACT,      DZM ACSTRT
              DZM MQSTRT
              DZM LKSTRT
              LAC BIT17
              DAC SCSTRT
              LAC MQSTRT
              IMG
              LAC LKSTRT
              RAR
              IAC ACSTRT
              LLS 01
              DAC ACEND
              GLK
              DAC LKEND
              LACS
              DAC SCEND
              LACQ
              DAC MQEND
              SAD LKSTRT
              SKP
              JMP .+7
              LAC MQSTRT
              RAL
              LAC ACSTRT
              RAL
              SAD ACEND
              SKP
              JMS LLSERR
              JMS SWITCH
              LLSACT+5
              LAC LKSTRT
              IS7 LKSTRT
              SNA
              JMP LLSACT+5
              DZM LKSTRT
              IAC ACSTRT
              IS7 ACSTRT
              SNA
              JMP LLSACT+5
              DZM ACSTRT
              LAC BIT0
              SAD MQSTRT
              JMP .+3
              DAC MQSTRT
              JMP LLSACT+5
              JMS SWTCHS
              LLSTS1

              /START SCOPE LOOP

              /L=0, 1, 0, 1
              /AC=0, 0, 1, 1

              /SAVE SC FOR TYPEOUT

              /MQ FOR TYPEOUT

              /LINK TO MQ17?
              /YFS, OK
              /MQ ERROR

              /AC0 SHOULD BE = MQ0

              /L=0,1,0, 1,0, 1, 0, 1

              /AC0 = 0, 0, 1, 1, 0, 0, 1, 1

              /TESTED MQ0 = 1?
              /YFS
              /MQ0 = 0, 4 PASSES
              /=1, 4 PASSES
```

/WILL EACH BIT OF THE MQ SHIFT TO THE NEXT
/1=0; AND 0=1 LEFT

```
LLSTS3,      LAC BIT17          /START MQ 17 TO MQ 16
              DAC MQSTRT
              DAC SCSTRT
              DZM LKSTRT
              DZM ACSTRT
              LAC MQSTRT          /START SCOPE LOOP
              LMQ
              CLA CLL-OPR        /AC AND L ALWAYS 0'S
              LLS 01
              DAC ACEND
              GLK
              DAC LKEND          /FOR TYPEOUTS
              LACS
              DAC SCEND          /FOR TYPEOUTS
              LACQ
              DAC MQEND
              LAC MQSTRT
              RAL
              SAD MQEND
              SKP
              JMP .+5
              LAC ACSTRT
              RAL
              SAD ACEND
              JMP .+2
              JMS LLSERR
              JMS SWITCH
              LLSTS3+5          /END SCOPE
              LAC MQSTRT        /SET UP NEXT MQ BIT
              CLL RAL-OPR
              DAC MQSTRT
              SNL
              JMP LLSTS3+5      /TESTED MQ0 = 1
```

CAH-347
→

/WILL EACH BIT OF THE MQ SHIFT TO THE NEXT
 /1-1, 0-1, 1-0 LEFT

```

LLSTS4,      LAC NBIT17      /START 777776
              DAC MQSTRT
              CMA
              DAC SCSTRT      /LIS 01
              DAC LKSTRT      /LINK ALWAYS = 1
              CLC
              DAC ACSTRT      /AC = 1'S ALL
              LAC MQSTRT      /START SCOPE LOOP
              LMQ
              STL CLC-OPR
              LLS 01
              DAC ACEND
              GLK
              DAC LKEND        /L, FOR TYPEOUT
              LACS
              DAC SCEND        /SC FOR TYPEOUT
              LACQ
              DAC MQEND
              LAC MQSTRT      /SIMULATE LLS
              STL              /TO GET
              RAL              /COMPARE CONSTANT
              SAD MQEND        /MQ SHIFT OK?
              SKP              /YFS
              JMP .+5
              LAC ACSTRT
              RAL
              SAD ACEND        /AC SHIFT OK?
              JMP .+2
              JMS LLSERR
              JMS SWITCH      /END SCOPE
              LLSTS4+7
              STL
              LAC MQSTRT
              RAL
              DAC MQSTRT
              SZL              /TESTED MQ0 = 0
              JMP LLSTS4+7
              JMS SWCHS
              LLSTS3
  
```

/WILL MQ AC SHIFT A 1 BIT 1 TO 44 PLACES
SW888 LL8 8 NED

```

LLST85,      DZM ACSTRT          /AC START ZEROS
              LAC BIT17
              DAC SCSTRT        /SC INCREMENTED TO 44
              DAC MQSTRT        /MQ START BIT 17 = 1
              DAC LKSTRT
              DAC MQCOMK
              DZM ACCOMK
              LAC KLLSS1
              DAC LLSSEX        /RRESET SHIFT TO 1
LLSSL1,      LAC MQCOMK
              CLL RAL-OPR
              DAC MQCOMK
              LAC ACCOMK
              RAL
              DAC ACCOMK
              LAC MQSTRT        /START SCOPE LOOP
              LMQ
LLSSFX,      STL CLA-OPR
              LLSS 01          /SC = 1 TO 44
              DAC ACEND
              LACS
              DAC SCEND
              GLK
              DAC LKEND
              LACQ
              DAC MQEND
              SAD MQCOMK
              SKP
              JMP .+4
              LAC ACCOMK
              SAD ACEND
              SKP
              JMP .+6
              LAC LKEND
              SNA
              LAC SCEND
              SNA
              JMP .+2
              JMS LLSSER
              JMS SWITCH
              LLSSEX-3
              ISZ LLSSEX
              ISZ SCSTRT
              LAC SCSTRT
              XOR FOUR5
              SZA
              JMP LLSSL1
              JMS SWTCHS
              LLST85

```

/WILL MQ AC SHIFT A NO BIT 1 TO 44 PLACES

LLSTS6,	DZM LKSTRT	
	LAC BIT17	
	DAC SCSTRT	
	CMA	
	DAC MQSTRT	
	DAC MQCOMK	
	CLC	
	DAC ACSTRT	
	DAC ACCOMK	
	LAC KLLSS1	
	DAC LLSSX2	
LLSSL2,	LAC MQCOMK	/FORM AC
	STL	
	RAL	/AND MQ
	DAC MQCOMK	/COMPARE CONSTANTS
	LAC ACCOMK	
	RAL	
	DAC ACCOMK	
	LAC MQSTRT	/SET UP SHIFT START SCOPE LOOP
	LMQ	
	CLL CLC-OPR	
LLSSX2,	LLSS 01	/SC=1 TO 44 PLACES
	DAC ACEND	
	LACS	
	DAC SCEND	/GET SC FOR TEST 4
	GLK	
	DAC LKEND	/LINK SHOULD BE 1
	LACQ	
	DAC MQEND	
	SAD MQCOMK	/MQ SHIFT OK?
	SKP	
	JMP .+4	
	LAC ACCOMK	
	SAD ACEND	/AC SHIFT OK?
	SKP	
	JMP .+4	
	LAC LKEND	
	SAD BIT17	/LINK SET TO 1?
	SKP	
	JMP .+4	
	LAC SCEND	
	SNA	/SC GO TO 0?
	SKP	
	JMS LLSSER	
	JMS SWITCH	
	LLSSX2-3	
	ISZ LLSSX2	/ADVANCE TO NEXT SHIFT
	ISZ SCSTRT	
	LAC SCSTRT	
	XOR FOUR5	/SHIFTED 44 PLACES?
	SZA	
	JMP LLSSL2	
	JMS SWTCHS	/REPEAT SEQUENCE SET?
	LLSTS6	
	JMP LRSTS1	

/COMMON ERROR TYPEOUT LLS

```
LLSERR,      JMP .  
              JMS ERROR  
              TYLLS  
              SCSTRT 740000  
              LLSERR 400000  
              HDR5  
              LKSTRT 500000  
              ACSTRT 600000  
              MQSTRT 600000  
              TYPATR  
              LKEND 500000  
              ACEND 600000  
              MQEND 600000  
              TYRES  
              TYLACS  
              SCEND 740000  
              0  
              JMP I LLSERR  
KLLSS1,      LLSS 01
```

/TO SET UP LONG LEFT SHIFTS

/COMMON ERROR TYPEOUT
/LLS SIGNED

```
LLSSFR,      JMP .  
              JMS ERROR  
              TYLLSS  
              SCSTRT 740000  
              LLSSFR 400000  
              HDR5  
              LKSTRT 500000  
              ACSTRT 600000  
              MQSTRT 600000  
              TYPATR  
              SPACE4  
              ACCOMK 600000  
              MQCOMK 600000  
              TYCOR  
              LKEND 500000  
              ACEND 600000  
              MQEND 600000  
              TYINCO  
              TYLACS  
              SCEND 740000  
              0  
              JMP I LLSSFR
```


/LONG RIGHT SHIFT
/LRS 01 AC, MQ AND L = 0'S

LRSTS1, DZM ACSTRT
 DZM MQSTRT
 DZM LKSTRT
 LAC BIT17
 DAC SCSTRT
 CLO
 CLA CLL-OPR
 LRS 01
 DAC ACEND
 GLK
 DAC LKEND
 LACS
 DAC SCEND
 LACQ
 DAC MQEND
 SNA
 LAC ACEND
 SNA
 LAC SCEND
 SNA
 LAC LKEND
 SNA
 SKP
 JMS LRSERR
 JMS SWITCH
 LRSTS1+5

/SFT INITIAL CONDITIONS

/START SCOPE LOOP

/MQ SHOULD BE 0

/AC=0?

/SQ GO TO 0?

/LINK STILL 0?

/END SCOPE

/DOES LINK GO TO AC 0 ON AN LRS
/0-0, 1-0, 0-1, 1-1

LRSTS2,

DZM MQSTRT
DZM ACSTRT
DZM LKSTRT
LAC BIT17
DAC SCSTRT
LAC LKSTRT
RAR
CLQ
LAC ACSTRT
LRS 01
DAC ACEND
LACQ
DAC MQEND
LACS
DAC SCEND
QLK
DAC LKEND
SAD LKSTRT
SKP
JMP .+12
LAC MQEND
SZA
JMP .+7
LAC LKSTRT
RAR
LAC ACSTRT
RAR
SAD ACEND
SKP
JMS LRSERR
JMS SWITCH
LRSTS2+5
LAC LKSTRT
ISZ LKSTRT
SNA
JMP LRSTS2+5
DZM LKSTRT
LAC BIT0
SAD ACSTRT
JMP .+3
DAC ACSTRT
JMP LRSTS2+5

/START SCOPE LOOP

/SET UP COMPLETE

/SAVE RESULTS

/LINK SHOULD NOT CHANGE

/END SCOPE

/DOES AC17 GO TO MQ0 ON AN LRS
/0-0, 1-0, 0-1, AND 1-1

LRSTS3,

DZM LKSTRT
DZM MQSTRT
DZM ACSTRT
LAC BIT17
DAC SCSTRT
CLL
LAC MQSTRT
LMQ
LAC ACSTRT
LRS 01
DAC ACEND
GLK
DAC LKEND
LACS
DAC SCEND
LACQ
DAC MQEND
LAC ACSTRT
RAR
LAC MQSTRT
RAR
SAD MQEND
SKP
JMP .+3
LAC ACEND
SZA
JMP .+4
LAC LKEND
SNA
SKP
JMS LRSERR
JMS SWITCH
LRSTS3+5
LAC ACSTRT
IS7 ACSTRT
SNA
JMP LRSTS3+5
LAC BIT0
DZM ACSTRT
SAD MQSTRT
JMP .+3
DAC MQSTRT
JMP LRSTS3+5

/LINK ALWAYS 0

/SHIFT OF 1

/SFT MQ

/GENERATE MQ

/COMPARE

/CONSTANT

/AC17 TO MQ0 OK?

/AC GO TO 0?

/DOES AC17 NOT GO TO LINK ON AN LRS

```
LRSTS4      DZM LKSTRT
             DZM ACSTRT
             DZM MQSTRT      /MO ALWAYS ZERO
             IAC BIT17
             DAC SCSTRT      /SHIFT OF 1
             CLQ
             LAC LKSTRT
             RAR              /SPT LINK INITIAL 0 OR 1
             LAC ACSTRT      /AC=1 OR 0
             IRS 01
             DAC ACEND
             LACS
             DAC SCEND
             LACQ
             DAC MQEND
             GLK
             DAC LKEND
             SAD LKSTRT      /WAS LINK ALTERED
             SKP
             JMS LRSERR
             JMS SWITCH
             LRSTS4+5
             LAC LKSTRT
             IS7 LKSTRT
             SNA              /TESTED L=1?
             JMP LRSTS4+5     /NO
             DZM LKSTRT
             LAC ACSTRT
             IS7 ACSTRT
             SNA              /TESTED AC 17=1
             JMP LRSTS4+5
             JMS SWCHS
             LRSTS1
```

/WILL AC MQ SHIFT 4 1 BIT EACH POSITION RIGHT

```
LRSTS5,      DZM LKSTRT
              DZM MQSTRT
              DZM MQCOMK
              LAC HIT17
              DAC SCSTRT
              IAC RIT0
              DAC ACSTRT
              DAC ACCOMK
              LAC ACCOMK
              CLL RAR-OPR
              DAC ACCOMK
              LAC MQCOMK
              RAR
              DAC MQCOMK
LRST5L,      LAC MQSTRT
              LMQ
              CLL
              LAC ACSTRT
              LRS 01
              DAC ACEND
              GLK
              DAC LKEND
              LACS
              DAC SCEND
              LACQ
              DAC MQEND
              SAD MQCOMK
              SKP
              JMP .+4
              LAC ACCOMK
              SAD ACEND
              SKP
              JMS LRSERR
              JMS SWITCH
              LRST5L
              IAC ACSTRT
              CLL RAR-OPR
              DAC ACSTRT
              LAC MQSTRT
              RAR
              DAC MQSTRT
              SNL
              JMP LRST5L-6
              JMS SWITCHS
              LRSTS5
              /GENERATE COMPARE
              /CONSTANTS
              /MQ SHIFT OK?
              /AC SHIFT OK?
```

/WILL AC-MQ SHIFT A NO BIT 1 POSITION
/RIGHT FROM EACH BIT

```

LRSTS6,      LAC BIT17
              DAC SCSTRT
              DAC LKSTRT
              LAC NBIT0           /377777
              DAC ACSTRT
              DAC ACCOMK
              CLC
              DAC MQSTRT
              DAC MQCOMK
              LAC ACCOMK         /GENERATE NEXT
              STL
              RAR
              DAC ACCOMK         /SFT OF
              LAC MQCOMK         /AC MQ COMPARE
              RAR                 /CONSTANTS
              DAC MQCOMK
LRST6L,      LAC MQSTRT         /SFT UP LRS
              LMQ
              STL
              LAC ACSTRT
              LRS #1
              DAC ACEND
              LACS
              DAC SCEND         /FOR TYPEOUTS
              GLK
              DAC LKEND         /FOR TYPEOUTS
              LACQ
              DAC MQEND
              SAD MQCOMK         /MQ SHIFT OK?
              SKP
              JMP .+4
              LAC ACCOMK
              SAD ACEND         /AC SHIFT OK?
              SKP
              JMS LRSERR
              JMS SWITCH
              LRST6L
              LAC ACSTRT
              STL
              RAR
              DAC ACSTRT
              LAC MQSTRT
              RAR
              DAC MQSTRT
              SZL
              JMP LRST6L-7       /SHIFTED TILL MQ17=0
              JMS SWCHS
              LRSTS6

```

/WILL AC MQ SHIFT A 1 BIT
/RIGHT TO 44 PLACES

```
LRSTS7,      DZM MQSTRT
              DZM LKSTRT
              LAC RIT17
              DAC SCSTRT
              LAC BIT0
              DAC ACSTRT
              DAC ACCOMK
              DZM MQCOMK
              LAC LRST6L+4
              DAC LRST7E
LRST7L,      LAC ACCOMK
              CLL RAR-OPR
              DAC ACCOMK
              LAC MQCOMK
              RAR
              DAC MQCOMK
              LAC MQSTRT
              LMQ
              CLL
              LAC ACSTRT
LRST7E,      LRS 01
              DAC ACEND.
              GLK
              DAC LKEND
              LACS
              DAC SCEND
              LACQ
              DAC MQEND
              SAD MQCOMK
              SKP
              JMP .+4
              LAC ACCOMK
              SAD ACEND
              SKP
              JMS LRSER1
              JMS SWITCH
              LRST7E-4
              ISZ LRST7E
              ISZ SCSTRT
              LAC FOUR5
              SAD SCSTRT
              SKP
              JMP LRST7L
              JMS SWTCH8
              LRSTS7

              /LRS 01
              /FOR EXECUTE

              /GENERATE AC/MQ
              /COMPARE CONSTANTS

              /SPT UP LRS

              /1 TO 44 PLACES

              /MQ SHIFT OK?

              /AC END OK?

              /INCREMENT SHIFT COUNT
              /FOR TYPEOUTS

              /SHIFTED 44 PLACES?
              /YPS
```

/WILL AC MQ SHIFT A NO BIT RIGHT
/1 TO 44 PLACES

LRSTS8,	CLC	
	DAC MQSTRT	/MQ START = 1'S
	DAC MQCOMK	
	LAC NBIT0	/AC START BIT 0'S
	DAC ACSTRT	
	DAC ACCOMK	
	LAC BIT17	
	DAC SCSTRT	
	DAC LKSTRT	
	LAC LRST6L+4	/LRS 01
	DAC LRST8E	/FOR EXECUTE
LRSTAL,	LAC ACCOMK	/GENERATE
	STL	
	RAR	/NXT
	DAC ACCOMK	/COMPARE CONSTANTS
	LAC MQCOMK	
	RAR	
	DAC MQCOMK	
	LAC MQSTRT	/SFT UP LRS
	LMQ	
	LAC ACSTRT	
	STL	
LRST8E,	LRS 01	/1 TO 44 PLACES
	DAC ACEND	
	GLK	
	DAC LKEND	
	LACS	
	DAC SCEND	
	LACQ	
	DAC MQEND	
	SAD MQCOMK	/MQ SHIFT OK?
	SKP	
	JMP .+4	
	LAC ACCOMK	
	SAD ACEND	/AC SHIFT OK?
	SKP	
	JMS LRSER1	
	JMS SWITCH	
	LRST8E-4	
	IS7 LRST8E	/ADVANCE SHIFT
	IS7 SCSTRT	/COUNT
	LAC FOUR5	
	SAD SCSTRT	/SHIFTED 44 PLACES
	SKP	
	JMP LRST8L	
	JMS SWITCH8	
	LRSTS8	

/WILL MQ SHIFT LEFT 1
/EVERY COMBINATION OF BITS

LLSSFQ,
DZM MQSTRT
DZM ACSTRT
IAC BIT17
DAC SCSTRT
DZM LKSTRT
LAC MQSTRT
FAF 20000
RAL
DAC ACCOMK
DAC MQCOMK
LAC MQSTRT
LMO
LLSS 01
DAC ACEND
LACS
DAC SCEND
QLK
DAC LKEND
IACQ
DAC MQEND
SAD ACEND
SKP
JMS LLSSER
JMS SWITCH
LLSSFQ+5
IS7 ACSTRT
NOP
IS7 MQSTRT
JMP LLSSEQ+5
JMS SWITCHS
LLSSEQ

/AC AND MQ WILL
/ALWAYS BE =
/SHIFT IS ALWAYS 1

/AC SHOULD
/=MQ

/MQ AND
/AC SHIFT OK

/WILL MQ SHIFT RIGHT 1 EVERY
/COMBINATION OF BITS

```
LRSSEQ,      DZM ACSTRT      /AC AND MQ
              DZM MQSTRT      /ALWAYS =
              LAC BIT17
              DAC SCSTRT      /ALWAYS SHIFT OF 1
              LAC ACSTRT
              AND BIT17
              DAC LKSTRT      /LINK = AC 17
              RAR           /SO THAT AC WILL = MQ
              LAC MQSTRT
              RAR
              DAC MQCOMK      /AC AND MQ
              DAC ACCOMK      /SHOULD BE =
              RAL
              LMQ
              LRS 01
              DAC ACEND
              LACS
              DAC SCEND
              GLK
              DAC LKEND
              LACQ
              DAC MQEND
              SAD ACEND      /AC AND MQ R 1 OK
              SKP
              JMS LRSER1
              JMS SWITCH
              LRSSEQ+4
              ISZ ACSTRT
              NOP
              ISZ MQSTRT      /ALL COMBINATIONS
              JMP LRSSEQ+4
              JMS SWTCHS
              LRSSEQ
              JMP ENDSHF
```

/LRS COMMON ERROR TYPEOUT
/SHIFT OF 1
LRSERR,

JMP .
JMS ERROR
TYI RS
SCSTRT 740000
LRSERR 400000
HDR5
LKSTRT 500000
ACSTRT 600000
MQSTRT 600000
TYPATR
LKEND 500000
ACEND 600000
MQEND 600000
TYRES
TYLACS
SCEND 740000
0
JMP I LRSERR

```
/LRS COMMON ERROR TYPEOUT  
/SHIFTS OF MORE THAN 1  
LRSER1,      JMP .  
              LIMS ERROR  
              TYLRS  
              SCSTRT 740000  
              LRSER1 400000  
              WDR5  
              LKSTRT 500000  
              ACSTRT 600000  
              MQSTRT 600000  
              TYPATH  
              SPACE4  
              ACCOMK 600000  
              MQCOMK 600000  
              TYCOR  
              LKEND 500000  
              ACEND 600000  
              MQEND 600000  
              TYINCO  
              TYIACS  
              SCEND 740000  
              0  
              JMP I LRSER1
```

```
ACCOMK,      0  
MQCOMK,      0  
SCCOMK,      0
```

SIART

10.4.4 Random Data, Normalize, and Interrupt Test

/TAPE 5
/RANDOM DATA SHIFTS
/NORMALIZE TEST
/INTERRUPT TEST

5000/

LAC NBIT16
DAC CHARK

/SET PASS K TO -3

/START RANDOM DATA SHIFTS
/LEFT 0 TO 44 PLACES

RANSHE,

LAC NBIT5
DAC PASSK
JMS RANGEN
DAC MOSTRT
JMS RANGEN
DAC SHFBUF
LAW SHFBUF
DAC 10
TAD BIT17
DAC 11
LAC MOSTRT
LMO
DAC I 10
LAC SHFBUF
DAC ACSTRT
LLSS 01
DAC I 10
LACQ

/GENERATE AC START

SETLLS,

DAC I 10
LAC I 11
IS7 11
LLS 01
DAC I 10
LACQ
DAC I 10
LAW SHFBUF+111
SAD 10
SKP
JMP SETLLS
GLK
DAC LKSTRT
DZM SCSTRT
LAC KLLSS
DAC LKANEX
LAW SHFBUF-1
DAC 10

/SHIFTED 44 PLACES?

```
LRANLP,      LAC I 10
              DAC ACCOMK
              LAC I 10
              DAC MQCOMK
              LAC MQSTRT
              LMO
              LAC LKSTRT
              RAR
              LAC ACSTRT
5057 LRANEX,  LLSS           /0 TO 44 PLACES
              DAC ACEND
              GLK
              DAC LKEND
              LACS
              DAC SCEND
              LACQ
              DAC MQEND
              SAD MQCOMK     /MO = PREDICTED?
              SKP
              JMP .+4
              LAC ACCOMK
              SAD ACEND     /AC END = PREDICTED?
              SKP
              JMS LLSSER
              JMS SWITCH
              LRANLP+4
              IS7 LRANEX
              IS7 SCSTRT
              LAC FOUR5
              SAD SCSTRT
              SKP
              JMP LRANLP
              JMS SWTCHS
              RANSHF+6
              RLSTAY,      IS7 PASSK
              JMP RANSHF+2
              /SHIFTED 44 PLACES?
```

```
/RANDOM DATA RIGHT 0 TO 44 PLACES
RANRIT,    LAC MBITS
           DAC PASSK
           JMS RANGEN          /GENERATE MQ START
           DAC MQSTRT
           JMS RANGEN          /GENERATE ACSTRT
           DAC ACSTRT
           LAW SHFBUF-1
           DAC 10
           DAC 11
           LAC ACSTRT
           DAC I 10
           LAC MQSTRT
           DAC I 10
           LMQ
           CLL
SETLRS,    LAC I 11
           ISZ 11          /GENERATE AC MQ
           LRS 01
           DAC I 10        /COMPARE CONSTANTS
           LACQ
           DAC I 10
           LAW SHFBUF+111
           SAD 10
           SKP
           JMP SETLRS
           LAC KLPS
           DAC RRANEX
           DZM LKSTRT
           DZM SCSTRT
           LAW SHFBUF-1
           DAC 10
RRANLP,    LAC I 10
           DAC ACCOMK
           LAC I 10
           DAC MQCOMK
           LAC MQSTRT
           LMQ
           LAC ACSTRT
           CLL
RRANEX,    LRS 0          /0 TO 44 PLACES
           DAC ACEND
           GLK
           DAC LKEND
           LACS
           DAC SCEND
           LACQ
           DAC MQEND
           SAD MQCOMK
           SKP
           JMP .+4
           LAC ACCOMK
           SAD ACEND
           SKP
           JMS LRSER1
           JMS SWITCH
```

```
RRANEX-4  
ISZ RRANEX  
ISZ SCSTRT  
LAC FOUR5  
SAD SCSTRT  
SKP  
JMP RRANLP  
JMS SWTCHS  
RANRIT+6  
ISZ PASSK  
JMP RANRIT+2  
  
RRSTAY,
```


/RANDOM DATA SEQUENCED

```
RANSEQ,      LAC NBITS
              DAC PASSK
              JMS RANGEN
              DAC ACSTRT
              EAF 21000
              GLK
              DAC SVSIGN
              RTR
              DAC SVSIGN+1
              JMS RANGEN
              AND NBIT17
              TAD SVSIGN
              DAC MQSTRT
              LAC NBIT17
              DAC SVMASK
              LAC NBIT0
              DAC SVMASK+1
              DZM SCSTRT
              DZM LKSTRT
RANSQ0,      LAC MQSTRT
              LMQ
              CLL
              LAC ACSTRT
              LLSS 1
              LRS 2
              LLSS 2
              LRS 1
              JMS SEQCOM
              JMS SWITCH
              RANSQ0
              JMS NXTSEQ

              /GET AC SIGN CLR AC

              /MAKE MQ17=AC0

              /SEQUENCE 0
```

/SEQUENCE 1
/RIGHT 2, L4, R4, L2

5253RANSQ1, LAC MQSTRT
 CLIT
 LMQ
 LAC ACSTRT
 LRSS 2
 LLSS 4
 LRS 4
 LLSS 2
 JMS SEQCOM
 JMS SWITCH
 RANSQ1
 JMS NXTSEQ

/SEQUENCE 1 R2, L4, R4, L2

/SPT UP

/COMPARE RESULTS

/LEFT 3, RIGHT 6, LEFT 6, RIGHT 3
/SEQUENCE 2
RANSQ2,

LAC MQSTRT
LMQ
CLL
LAC ACSTRT
LLSS 3
LRS 6
LLSS 6
LRS 3
JMS SEQCOM
JMS SWITCH
RANSQ2
JMS NXTSEQ

/SEQUENCE 5 RIGHT 6, LEFT 12, RIGHT 12, LEFT 6

RANS05, LAC MQSTRY
 LMQ
 CLL
 LAC ACSTRY
 LRSS 6
 LLSS 14
 LRS 14
 LLSS 6
 JMS SEQCOM
 JMS SWITCH
 RANS05
 JMS NXTSEQ

/SEQUENCE 6 LEFT 7 RIGHT 14, LEFT 12, RIGHT 7

RANS06, LAC MQSTRY
 LMQ
 CLL
 LAC ACSTRY
 LLSS 7
 LRS 16
 LLSS 16
 LRS 7
 JMS SEQCOM
 JMS SWITCH
 RANS06
 JMS NXTSEQ

/SEQUENCE 7 RIGHT 8, LEFT 16, RIGHT 16, LEFT 8

RANSQ7, LAC MQSTRT
 LMQ
 LAC ACSTRT
 CLL
 LRSS 10
 LLSS 20
 LRS 20
 LLSS 10
 JMS SEQCOM
 JMS SWITCH
 RANSQ7
 JMS NXTSEQ

/SEQUENCE 8 LEFT 9, RIGHT 18, LEFT 78, RIGHT 9

RANSQ8, LAC MQSTRT
 LMQ
 LAC ACSTRT
 CLL
 LLSS 11
 LRS 22
 LLSS 22
 LRS 11
 JMS SEQCOM
 JMS SWITCH
 RANSQ8
 1S7 PASSK
 JMP RANSEQ+2
 JMS SWITCHS
 RANSEQ
 JMP NRMLZE

/SET AC SIGN INTO NEXT AC
/AND MQ BITS

NXTSEQ, JMP .
 LAC SVSIGN
 CLL RAL-OPR
 DAC SVSIGN
 LAC SVSIGN+1
 CLL RAR-OPR
 DAC SVSIGN+1
 LAC SVMASK
 STL
 RAL
 DAC SVMASK
 AND MQSTRT
 TAD SVSIGN
 DAC MQSTRT
 LAC SVMASK+1
 STL
 RAR
 DAC SVMASK+1
 AND ACSTRT
 TAD SVSIGN+1
 DAC ACSTRT
 ISZ SCSTRT
 JMP I NXTSEQ

SVSIGN, 0
 0
SVMASK, 0
 0

/To FILL MQ

/CIR MQ BIT
/MAKE MQ = AC 0

/CIR AC BIT
/MAKE ACX = AC 0

/INDICATE NEXT SEQUENCE

/RANDOM DATA SEQUENCED
/COMMON COMPARE AND ERROR TYPE

```
SEQCOM,      JMP .
              DAC ACEND
              GLK
              DAC LKEND
              LACS
              DAC SCEND
              LACQ
              DAC MQEND
              SAD MQSTRT      /MQ SAME AS START
              SKP
              JMP .+3        /ERROR MQ
              LAC SCEND
              SZA
              JMP .+4        /ERROR SC
              LAC ACSTRT
              SAD ACEND
              SKP
              JMP .+5        /ERROR AC
              EAE 21000      /GET AC SIGN CLR AC
              GLK
              SAD LKEND      /LINK END = AC SIGN?
              JMP I SEQCOM   /ALL OK - EXIT
              JMS ERROR
              TYRDSQ
              SPACE3
              SCSTRT 740000
              SEQCOM 400000
              HDR5
              LKSTRT 500000
              ACSTRT 600000
              MQSTRT 600000
              TYSTRT
              LKEND 500000
              ACEND 600000
              MQEND 600000
              TYRES
              TYLACS
              SCEND 740000
              0
              JMP I SEQCOM   /ERROR EXIT
```

```

/RANDOM NUMBER GENERATOR
/18 BIT
RANGEN,      JMP .
              LAC RANNO
              CLL RAR-OPR
              SZL
              XOR BIT0
              XOR RANNO+1
              ADD RANNO+1
              DAC RANNO
              JMP I RANGEN
RANNO,       736425
              335671
PASSK,       0
KLRS,        LRS
KLLSS,       LLSS
SHFBUF,      0
SHFBUF+112/

```

```

/NORMALIZE TEST
/DOES NORMS GET AC 0 = 0 TO L

```

```

NRMLZE,      DZM MQSTRT
              DZM SCSTRT
              LAC BIT1
              DAC ACSTRT
              DZM LKSTRT
              LAC LKSTRT
              RAR
              CLQ
              LAC ACSTRT
              NORMS -44
              DAC ACEND
              LACQ
              DAC MQEND
              LACS
              DAC SCEND
              GLK
              DZM SCCOMK
              DAC LKEND
              SZA
              JMS NORMSE
              JMS SWITCH
              NRMLZE+5
              LAC LKSTRT
              ISZ LKSTRT
              SNA
              JMP NRMLZE+5

```

/START SCOPE LOOP

/SC = 0

/SAVE RESULTS

/AC SIGN IS 0

/END SCOPE LOOP

/DOES NORMS GET ACQ=1 TO L

```
NRML71,      CLC
              DAC MQSTRT
              DZM SCSTRT
              DZM LKSTRT
              CLC
              DAC ACSTRT
              LAC LKSTRT
              RAR
              LAC ACSTRT
              CLO 4
              NORMS 0-44
              DAC ACEND
              LACQ
              DAC MQEND
              LACS
              DAC SCEND
              GLK
              DZM SCCOMK
              DAC LKEND
              SNA
              JMS NORMSE
              JMS SWITCH
              NRMLZ1+6
              LAC LKSTRT
              ISZ LKSTRT
              SNA
              JMP NRMLZ1+6

              /START SCOPE LOOP

              /SPT MQ = 1'S

              /END SCOPE LOOP
```

/WILL NORM STOP SHIFT WITH
/AC 0 AND AC 1 UNEQUAL? 01, 10

```
NRMLZ2,      DZM MQSTRT
              DZM LKSTRT
              LAC SEVSEV
              DAC SCCOMK
              LAC BIT1
              DAC ACSTRT
              LAC BIT17
              DAC SCSTRT
              LAC MQSTRT
              LMO
              CLL
              LAC ACSTRT
              NORM -43
              DAC ACEND
              LACQ
              DAC MQEND
              GLK
              DAC LKEND
              LACS
              DAC SCEND
              SAD SEVSEV
              SKP
              JMS NORMER
              JMS SWITCH
              NRMLZ2+10
              LAC ACSTRT
              CMA
              DAC ACSTRT
              LAC MQSTRT
              CMA
              DAC MQSTRT
              SZA
              JMP NRMLZ2+10

              /START SCOPE LOOP

              /SET UP COMPLETE
              /SC = 1

              /SAVE RESULTS

              /SC = -1?

              /END SCOPE LOOP
```

/DOES NORM NOT STOP SHIFT
/ON AC 0 = AC1 00, 11,

```
NRM123,      DZM MQSTRT
              DZM LKSTRT
              LAC BIT16           / NORMALIZE SC = 2
              DAC SCSTRT
              LAC SEVSEV
              DAC SCCOMK
              LAC BIT2
              DAC ACSTRT
              LAC MQSTRT         /START SCOPE LOOP
              LMQ
              CLL
              LAC ACSTRT         /COMPLETE SET UP
              NORMS -42         /SC = 2
              DAC ACEND
              LACS
              DAC SCEND         /SAVE RESULTS
              GLK
              DAC LKEND
              LACQ
              DAC MQEND
              SPA
              CMA
              SZA
              JMP .+6
              LAC ACEND
              SPA
              CMA
              SAD BIT1
              SKP
              JMP .+4
              LAC SCEND
              SAD SEVSEV
              SKP
              JMS NORMSE
              JMS SWITCH
              NRMLZE+6
              LAC ACSTRT
              CMA
              DAC ACSTRT
              LAC MQSTRT
              CMA
              DAC MQSTRT
              SZA
              JMP NRMLZ3+10
              /MO = ALL 0'S OR ALL 1'S
              /ERROR IN MQ
              /AC NEGATIVE?
              /MAKE POSITIVE
              /AC NORMALIZE CORRECT?
              /AC IN ERROR
              /SC = -1?
              /END SCOPE LOOP
```

/WILL NORMALIZE NORMALIZE A POSITIVE
/NUMBER WITH A 1 FROM AC BIT 1 TO M1 17 WITH SC=44 AT START
/AND A NEGATIVE NUMBER WITH A0 IN AC BIT 1
/TO MQ 17

```
NRMLZ4,      DZM MQSTRT
              DZM LKSTRT
              LAC FOUR4
              DAC SCSTRT
              LAC BIT1
              DAC ACSTRT
              LAC THREE4
              DAC SCCOMK      /TO COMPARE SC
              LAC MQSTRT     /SCOPE LOOP START
              LMG
              CLL
              LAC ACSTRT     /SET UP COMPLETE
              NORMS         /SC = 44
              DAC ACEND
              LACQ
              DAC MQEND      /SAVE RESULTS
              QLK
              DAC LKEND
              LACS
              DAC SCEND
              SAD SCCOMK
              SKP
              JMP .+6        /SC, ERROR
              LAC ACEND
              SPA
              CMA
              SAD BIT1
              SKP
              JMP .+5
              LAC MQEND
              SPA
              CMA
              SZA
              JMS NORMER
              JMS SWITCH
              NRMLZ4+10
              LAC MQSTRT
              LMG
              CLL
              LAC ACSTRT
              ISZ SCCOMK
              LRSS 1
              DAC ACSTRT
              LACQ
              DAC MQSTRT
              SAD ACSTRT     /AC AND MQ = 0'S OR 1'S
              SKP           /END 1 PASS
              JMP NRMLZ4+10
              SZA
              JMP NRMLZ5     /DONE ALL 1'S YET?
                          /YFS
```

LAC NBIT1
DAC ACSTRT
CLC
DAC MQSTRT
JMP NRMLZ4+6

/2ND SERIES
/NEGATIVE NUMBERS

/WILL A COMPLEMENT BIT PATTERN NORMALIZE
/MQ = 252525 AND 525252 AC = 0'S OR 1'S

```
NRML75,      DZM ACSTRT
              IAC COMBIT           /252525 PATTERN
              DAC MQSTRT
              LAC FOUR4
              DAC SCSTRT
              DZM LKSTRT
              LAC MQSTRT           /SCOPE LOOP START
              LMO
              CLL
              LAC ACSTRT
              NORMS
              DAC ACEND
              LACS
              DAC SCEND
              GLK
              DAC LKEND
              LACQ
              DAC MQEND
              SPA
              CMA
              SZA
              JMP .+4
              LAC ACEND           /ACEND SHOULD
              SAD MQSTRT         /= MQSTRT
              SKP
              JMP .+4           /AC ERROR
              LAC FIVE6
              DAC SCCOMK
              SAD SCEND         /SC INDICATE SHIFT 18
              SKP
              JMS NORMSE
              JMS SWITCH       /END SCOPE LOOP
              NRML75+6
              CLC
              DAC ACSTRT
              LAC MQSTRT
              CMA
              DAC MQSTRT
              SPA
              JMP NRMLZ5+6
              JMS SWTCHS       /TEST REPEAT SEQUENCE
              NRMLZE
              JMP INTEST       /GO TO INTERRUPT TEST
```

/NORMALIZE ERROR TYPEOUTS

```
NORMER,      JMP .  
              JMS ERROR  
              TYNORM  
              SCSTRT 740000  
              NORMER 400000      /ERROR ADDRESS  
              HDR5  
              LKSTRT 500000  
              ACSTRT 600000  
              MQSTRT 600000  
              TYPATR  
              LKEND 500000  
              ACEND 600000  
              MQEND 600000  
              TYRES  
              TYLACS  
              SCCOMK 740000  
              TYCOR  
              TYLACS  
              SCEND 740000  
              TYRES  
              0  
              JMP I NORMER
```

/NORMALIZE SIGNED ERROR TYPEOUTS

```
NORMSE,      JMP .  
              JMS ERROR  
              TYNRMS  
              SCSTRT 740000  
              NORMSE 400000  
              HDR5  
              LKSTRT 500000  
              ACSTRT 600000  
              MQSTRT 600000  
              TYPATR  
              LKEND 500000  
              ACEND 600000  
              MQEND 600000  
              TYRES  
              TYLACS  
              SCCOMK 740000  
              TYCOR  
              TYLACS  
              SCEND 740000  
              TYRES  
              0  
              JMP I NORMSE
```

/TEST PROGRAM INTERRUPT
/AFTER EAE OPERATIONS

	INTEST,	TSP	/PRINTER FLAG
		SKP	/NO CHECK DAD
		JMP .+4	ASP
		LAW 0	++ SML
		TLS	/TYPE NULL DAI
		TSP	/WAIT PRINTER FLAG
		JMP .-1	SKP
		LAC (JMP INTS1	++ SML
		DAC 1	/LOAD INT JMP
		ION	AMB
		EAE ←	TESTNI SML
		IOF	/SHOULD NOT GET HERE
6306		JMS ERROR	
		TYINTE	
		TYNOP	
		(.5) 400000	
		0	
6314	INTS1,	TSP	/WAIT IN CASE
		JMP .-1	/ON ERROR TESTNI
		JMS SWITCH	
		INTEST+11	
		LAC BIT17	
		DAC MQSTRT	
		DZM ACSTRT	
		DZM LKSTRT	
		LAC FOUR3	
		DAC SCSTRT	
		LAC JUMP INTS2	
		DAC 1	
	INTS2L,	LAC MQSTRT	/PREPARE FOR LLS
		LMQ	SHOTW SML
		CLA CLL-OPR	TESTNI
		ION	SAD
		LLS 43	/EXECUTE THIS
		IOF	/SHOULD NOT GET HERE
		JMS ERROR	
		TYINTE	
6332		TYLLS	
		SCSTRT 740000	
		(.6) 400000	
		0	
		JMP INTS2E	


```
INTS2,      DAC ACEND          /SAVE RESULTS
            LACS
            DAC SCEND
            LACQ
            DAC MQEND
            SZA              /Mn SHIFT OK?
            JMP .+4
            LAC ACEND
            SAD BIT0        /Ac SHIFT OK?
            SKP
            JMP .+4
            LAC SCEND      /SC GO TO 0?
            SNA
            JMP INTS2E
            JMS ERROR
            INDAT
            TYLLS
            SCSTRT 740000
            HDR5
            LKSTRT 500000
            ACSTRT 600000
            MQSTRT 600000
            TYPATR
            LKSTRT 500000
            ACEND 600000
            MQEND 600000
            TYLACS
            SCEND 740000
            0
INTS2E,     TSF              /WAIT IN CASE OF
            JMP .-1         /ERROR TIMEOUT
            JMS SWITCH
            INTS2L
            JMS SWTCHS
            INTEST
            LAS
            AND BIT6
            SNA              /TYPE AT END SET?
            JMP .+4
            LAW 13
            TY1
            IS7 CHARK
            JMP .+4
            JMS CRLF
            LAC NBIT16
            DAC CHARK
            LAS
            AND BIT5
            SNA              /CYCLE ALL TESTS
            JMP RANSHF      /ST?
            JMP NOPAC      /NO, STAY IN RANDOMS
                           /START SET UP TEST
```

START

4 101

I/O TRAP TEST

PROGRAM OPERATING INSTRUCTIONS

1. Read tape in via RIM start.
2. Set I/O TRAP switch to a one.
3. Press continue.
4. Program will HLT at final HLT with 505050 in the accumulator.
 - A. Any other HLTS are illegal and the program listing provided, should be consulted.
 - B. After final HLT set ACSO 1. This will allow the program to run continuously.
5. The program may be restarted at location 101g.