

DMP11 DMR11

M8203 STATIC DIAG #1
CZDMRCO

AH-E232C-MC
FICHE 1 OF 1

MAY 1980
COPYRIGHT © 79 80
MADE IN USA



The main body of the document is a large grid of approximately 15 columns and 25 rows of small, dense technical diagrams. Each cell in the grid contains a complex schematic or data table, likely representing a static diagnostic for a specific component or system. The diagrams are arranged in a regular, repeating pattern across the page.

.NLIST
.TITLE CZDMRC M8203 STATIC DIAG #1
.SBTTL PROGRAM DOCUMENT
.LIST

SEQ 0001

.REM @

IDENTIFICATION

PRODUCT CODE: AC-E231C-MC
PRODUCT NAME: CZDMRCO M8203 STATIC DIAG #1
PRODUCT DATE: FEBRUARY 1980
MAINTAINER: DIAGNOSTIC ENGINEERING
AUTHOR: DAVID HOFFMAN

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

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

COPYRIGHT (C) 1979, 1980 BY DIGITAL EQUIPMENT CORPORATION

THE FOLLOWING ARE TRADEMARKS OF DIGITAL EQUIPMENT CORPORATION:

DIGITAL	PDP	UNIBUS	MASSBUS
DEC	DECUS	DECTAPE	

CONTENTS

- 1.0 INTRODUCTION
- 2.0 HARDWARE REQUIREMENTS
- 3.0 PRELIMINARY PROGRAM REQUIREMENTS
- 4.0 GENERAL PROGRAM CONSIDERATIONS
 - 4.1 DIAGNOSTIC SUPERVISOR
 - 4.2 EXECUTION TIME
 - 4.3 XXDP+
 - 4.4 AC1/SLIDE
 - 4.5 APT
 - 4.6 MEMORY MANAGEMENT
 - 4.7 MEMORY PARITY OPTION
 - 4.8 ERROR LOGGING
- 5.0 PROGRAM LOAD MEDIA
- 6.0 OPERATING INSTRUCTIONS
 - 6.1 LOADING AND STARTING PROCEDURES
 - 6.1.1 LOADING PROCEDURES
 - 6.1.2 STARTING PROCEDURES
 - 6.1.3 STEPS FOR QUICK AND SIMPLE EXECUTION
 - 6.2 INITIAL DIALOGUE
 - 6.3 PROGRAM OPTIONS
 - 6.3.1 START COMMAND
 - 6.3.1.1 TESTS SWITCH
 - 6.3.1.2 PASS SWITCH
 - 6.3.1.3 FLAGS SWITCH
 - 6.3.1.4 END OF PASS SWITCH
 - 6.3.1.5 EFFECT OF START COMMAND
 - 6.3.2 RESTART COMMAND
 - 6.3.2.1 TESTS, PASS, AND FLAG SWITCHES
 - 6.3.2.2 UNITS SWITCH
 - 6.3.2.3 EFFECT OF RESTART COMMAND
 - 6.3.3 CONTINUE COMMAND
 - 6.3.3.1 PASS SWITCH
 - 6.3.3.2 FLAGS SWITCH
 - 6.3.3.3 EFFECT OF CONTINUE COMMAND
 - 6.3.4 PROCEED COMMAND
 - 6.3.4.1 FLAGS SWITCH
 - 6.3.4.2 EFFECT OF PROCEED COMMAND
 - 6.3.5 ADD COMMAND
 - 6.3.5.1 UNITS SWITCH
 - 6.3.5.2 EFFECT OF ADD COMMAND
 - 6.3.6 DROP COMMAND
 - 6.3.6.1 UNITS SWITCH
 - 6.3.6.2 EFFECT OF DROP COMMAND
 - 6.3.7 PRINT COMMAND
 - 6.3.7.1 EFFECT OF PRINT COMMAND
 - 6.3.8 DISPLAY COMMAND
 - 6.3.8.1 UNITS SWITCH
 - 6.3.8.2 EFFECT OF DISPLAY COMMAND

6.3.9 FLAGS COMMAND

6.3.9.1 EFFECT OF FLAGS COMMAND

6.3.10 ZFLAGS COMMAND

6.3.10.1 EFFECT OF ZFLAGS COMMAND

6.3.11 CONTROL CHARACTERS

6.3.12 HARDWARE PARAMETERS

6.3.13 SOFTWARE PARAMETERS

6.3.14 EXTENDED DISCUSSION OF P-TABLE DIALOGUE

7.0 DEVICE INFORMATION TABLES

8.0 TEST DESCRIPTIONS

8.1 DATA PATTERNS USED

9.0 ERROR INFORMATION

9.1 ERROR REPORTING

1.0 INTRODUCTION

THE M8203 IS A SINGLE-LINE SYNCHRONOUS LINE UNIT MODULE WHICH SUPPORTS BOTH CHARACTER-ORIENTED (DDCMP, BSC, ETC.) AND BIT-ORIENTED (SDLC, MDLC, ETC.) PROTOCOLS. THE PURPOSE OF THIS PROGRAM IS TO PERFORM DIAGNOSTIC TESTING OF ALL M8203 LOGIC IN A RELATIVELY STATIC MANNER. THE FOLLOWING FUNCTIONS WILL BE PERFORMED: LINE UNIT REGISTER ADDRESSING, USYRT ADDRESSING, STATIC BIT INTERACTION AND READ/WRITE LOGIC TESTS, BASIC TRANSMITTER AND RECEIVER SEQUENCING AND DATA BUFFERING AND STATIC OPERATIONS IN CHARACTER AND BIT-STUFFING MODES. IN ADDITION DATA MESSAGES WILL BE SENT AT SPEEDS OF 2400 BAUD TO 1 MEGABAUD, WITH LOOPBACK IN THE USYRT, ON THE LINE UNIT AT TTL LEVEL, OR THROUGH AN EXTERNAL TEST CONNECTOR WITH A SPECIFIC MODEM INTERFACE SELECTED.

THE STATIC LOGIC TESTS WILL PROVIDE EXTENSIVE TROUBLESHOOTING CAPABILITIES, SUCH AS TIGHT SCOPE LOOPS, SWITCH OPTIONS, AND ABILITY TO 'LOCK' ONTO INTERMITTENT ERRORS. IN ADDITION TESTS WILL BE DESIGNED AND STRUCTURED TO ACHIEVE MAXIMUM FAULT RESOLUTION AND FACILITATE REPLACEMENT OF THE SMALLEST FIELD REPLACEABLE UNIT.

THIS PROGRAM WILL BE IMPLEMENTED USING THE DIAGNOSTIC SUPERVISOR AND A STRUCTURED PROGRAMMING APPROACH. BECAUSE THE DESIGN WILL CONFORM TO THE SUPERVISOR (STANDALONE VERSION) THE PROGRAM WILL BE COMPATIBLE WITH ACT, APT, XXDP+, AND SLIDE.

THROUGH DIALOGUE WITH THE OPERATOR, THE PROGRAM WILL ALLOW MODIFICATION OF DEVICE PARAMETERS, SUCH AS UNIBUS ADDRESS, VECTOR ADDRESSES AND DEVICE PRIORITY. IN ADDITION, THE OPERATOR CAN SPECIFY PARTICULAR TESTS TO BE RUN AND A VARIETY OF LOOPING, RUNNING, AND REPORTING MODES.

DEVICE ERRORS WILL BE REPORTED AS THEY OCCUR. THE REPORT WILL INCLUDE A TEST NUMBER AND DESCRIPTION OF THE ERROR, GOOD AND BAD TEST DATA, AND APPLICABLE DEVICE REGISTER CONTENTS.

2.0 HARDWARE REQUIREMENTS

THE FOLLOWING HARDWARE IS REQUIRED TO RUN THE M8203 STATIC LOGIC TESTS:

PDP-11/04,05,10,20,30,34,35,40,45,50,60, OR 70
16K MEMORY
CONSOLE TERMINAL
DMC-11 OR KMC-11 MICROPROCESSOR
M8203 LINE UNIT AND BC08S-1 CABLE AND BERG CONNECTORS

3.0 PRELIMINARY PROGRAM REQUIREMENTS

THIS PROGRAM OPERATES THE MICROPROCESSOR EXTENSIVELY IN ORDER TO TEST THE LINE INIT. FOR THIS REASON, THE MICROPROCESSOR DIAGNOSTIC AND SUBSYSTEM FUNCTIONAL TESTS SHOULD BE RUN FIRST, AND ANY FAULTS FOUND IN THE MICROPROCESSOR MODULE SHOULD BE REPAIRED, PRIOR TO RUNNING THE M8203 STATIC LOGIC TESTS.

4.0 GENERAL PROGRAM CONSIDERATIONS

4.1 DIAGNOSTIC SUPERVISOR

THIS PROGRAM IS COMPATIBLE WITH THE STANDALONE DIAGNOSTIC SUPERVISOR, AND MUST BE LOADED TO BE CO-RESIDENT WITH THE SUPERVISOR, OR BE PREVIOUSLY COMBINED WITH THE SUPERVISOR AND LOADED AS A SINGLE FILE. IN EITHER CASE, THE COMBINED PROGRAM WILL NOT EXCEED 16K OF MEMORY.

4.2 EXECUTION TIME

THE MAXIMUM TIME REQUIRED TO RUN THE M8203 STATIC LOGIC TESTS IS ABOUT 45 SECONDS PER PASS FOR EACH UNIT.

4.3 XXDP+

THIS PROGRAM MAY BE LOADED UNDER XXDP+, AND MAY BE RUN IN DUMP MODE OR CHAIN MODE.

4.4 ACT/SLIDE

THIS PROGRAM MAY BE LOADED UNDER ACT OR SLIDE AND MAY BE RUN IN DUMP MODE OR CHAIN MODE.

4.5 APT

THIS PROGRAM MAY BE LOADED BY THE APT SYSTEM (INCLUDING APT-RD) AND RUN IN PROGRAM MODE OR SCRIPT MODE.

4.6 MEMORY MANAGEMENT

MEMORY MANAGEMENT IS NOT UTILIZED IN THIS PROGRAM. IF IT IS INSTALLED, IT IS DISABLED BY THE PROGRAM.

4.7 MEMORY PARITY OPTION

IF PARITY MEMORY IS INSTALLED, MEMORY PARITY TRAPS ARE DISABLED BY THE PROGRAM.

4.8 ERROR LOGGING

AT THE END OF EACH PASS ON ALL UNITS, THE PROGRAM PRINTS OUT THE CUMULATIVE TOTAL NUMBER OF ERRORS SINCE THE LAST START OR RESTART COMMAND.

5.0 PROGRAM LOAD MEDIA

THIS PROGRAM CAN BE LOADED FROM PAPER TAPE USING THE ABSOLUTE LOADER OR FROM ACT, SLIDE, OR APT SYSTEMS, OR FROM ANY MEDIA SUPPORTED BY XXDP+. WHEN USING THE PAPER TAPE ABSOLUTE LOADER, THE PROGRAM SHOULD BE LOADED FIRST, FOLLOWED BY THE DIAGNOSTIC SUPERVISOR. WHEN USING XXDP+, THE DIAGNOSTIC SUPERVISOR SHOULD BE LOADED FIRST, FOLLOWED BY THE DIAGNOSTIC PROGRAM.

6.0 OPERATING INSTRUCTIONS

6.1 LOADING AND STARTING PROCEDURES

6.1.1 LOADING PROCEDURES

THIS PROGRAM MAY BE LOADED FROM PAPER TAPE USING THE ABSOLUTE LOADER. IT MAY ALSO BE LOADED FROM ANY XXDP+ LOAD MEDIA. WHEN LOADED UNDER XXDP+, THE DIAGNOSTIC SUPERVISOR WILL BE LOADED AUTOMATICALLY.

6.1.2 STARTING PROCEDURES

THE PROGRAM STARTS AT LOCATION 200. USE STANDARD DEC PROCEDURES TO START THE PROGRAM.

6.1.3 STEPS FOR QUICK AND SIMPLE EXECUTION

THE DIAGNOSTIC CAN BE EXECUTED STANDALONE UNDER XXDP+, WITHOUT READING THE REMAINDER OF THIS DOCUMENT, AS FOLLOWS:

- A) LOAD AND START DIAGNOSTIC USING RUN COMMAND
- B) RECEIVE DIAGNOSTIC SUPERVISOR IDENTIFICATION AND PROMPT (DRS-C>)
- C) ENTER STA<CR>
- D) ANSWER HARDWARE AND SOFTWARE QUESTIONS
- E) GET END OF PASS MESSAGES OR ERROR MESSAGES
- F) TO END EXECUTION, ENTER CONTROL/C

6.2 INITIAL DIALOGUE

AFTER THE PROGRAM AND THE SUPERVISOR ARE LOADED AND THE PROGRAM IS STARTED, THE FOLLOWING IDENTIFICATION IS TYPED :

```
DRS LOADED
DIAG. RUN-TIME SERVICES
CZDMR-C-0
M8203 STATIC LOGIC TESTS - PART 1 OF 2
UNIT IS M8203
DR>
```

THE OPERATOR THEN PROCEEDS BY TYPING ONE OR MORE OF THE COMMANDS DESCRIBED IN THE FOLLOWING SECTION 6.3. (FOR MORE DETAILED INFORMATION, REFER TO THE DIAGNOSTIC SUPERVISOR FUNCTIONAL SPECIFICATION).

6.3 PROGRAM OPTIONS

6.3.1 START COMMAND

```
*****
STA(RT)/TESTS:<TEST-LIST>/PASS:<PASS-CNT>/FLAGS:
<FLAG-LIST>/EOP:<INCR>
*****
```

6.3.1.1 TESTS SWITCH (/TESTS:<TEST-LIST>)

<TEST-LIST> IS A SEQUENCE OF DECIMAL NUMBERS (1:2 ETC.) OR RANGES OF DECIMAL NUMBERS (1-5:8-10 ETC.) THAT SPECIFY THE TESTS TO BE EXECUTED. THE NUMBERS ARE SEPARATED BY COLONS. THE NUMBERS RANGE FROM 1 TO THE LARGEST TEST NUMBER IN THE DIAGNOSTIC. THEY MAY BE SPECIFIED IN ANY ORDER. TESTS WILL BE EXECUTED IN NUMERICAL ORDER REGARDLESS OF THE ORDER OF SPECIFICATION. THE DEFAULT IS TO EXECUTE ALL TESTS. ON THIS AND ALL SWITCHES, THE ANGLE BRACKETS <> ARE PUNCTUATION USED IN THE DEFINITION ONLY, AND ARE NOT TO BE TYPED BY THE OPERATOR. SEE EXAMPLE AT END OF 6.3.1.5.

6.3.1.2 PASS SWITCH (/PASS:<PASS-CNT>)

<PASS-CNT> IS A DECIMAL NUMBER INDICATING THE DESIRED NUMBER OF PASSES. A PASS IS DEFINED AS THE EXECUTION OF THE FULL DIAGNOSTIC (ALL SELECTED TESTS) AGAINST ALL UNITS SUBMITTED. THE DEFAULT IS NON-ENDING EXECUTION. IN THIS CASE EXIT FROM THE PROGRAM IS ACCOMPLISHED EITHER BY TYPING A CONTROL/C OR BY OCCURANCE OF AN ERROR WITH THE HALT ON ERROR FLAG BEING SET. THE EXIT IS A RETURN TO COMMAND MODE. SEE EXAMPLE AT END OF 6.3.1.5.

6.3.1.3 FLAGS SWITCH (/FLAGS:<FLAG-LIST>)

<FLAG-LIST> IS A SEQUENCE OF ELEMENTS OF THE FORM <FLAG>, <FLAG=1>, OR <FLAG=0>, SEPARATED BY COLONS, WHERE <FLAG> HAS ONE OF THE FOLLOWING VALUES:

HOE HALT ON ERROR, CAUSING COMMAND MODE TO BE ENTERED WHEN AN ERROR IS ENCOUNTERED
 LOE LOOP ON ERROR, CAUSING THE DIAGNOSTIC TO LOOP CONTINUOUSLY WITHIN THE SMALLEST DEFINED BLOCK OF CODING (SEGMENT, SUBTEST, OR TEST) CONTAINING THE ERROR
 IER INHIBIT ERROR REPORTING
 IBE INHIBIT BASIC ERROR REPORTS
 IXE INHIBIT EXTENDED ERROR REPORTS
 PRI DIRECT ALL MESSAGES TO A LINE PRINTER
 PNT PRINT NUMBER OF TEST BEING EXECUTED
 BOE BELL ON ERROR
 UAM RUN IN UNATTENDED MODE, BYPASSING MANUAL INTERVENTION TESTS
 ISR INHIBIT STATISTICAL REPORTS
 IDU INHIBIT DROPPING OF UNITS BY DIAGNOSTIC LOOP ON TEST
 LOT LOOP ON TEST

THE FLAGS NAMED OR EQUATED TO 1 ARE SET, THOSE EQUATED TO 0 ARE CLEARED. A FLAG NOT SPECIFIED IS CLEARED. IF THE FLAGS SWITCH IS NOT GIVEN ALL FLAGS ARE CLEARED. SEE EXAMPLE AT END OF 6.3.1.5.

6.3.1.4 END OF PASS SWITCH (/EOP:<INCR>)

<INCR> IS A DECIMAL NUMBER INDICATING HOW OFTEN (IN TERMS OF PASSES) IT IS DESIRED THAT THE END OF PASS MESSAGE BE PRINTED. THE DEFAULT IS AT THE END OF EVERY PASS. SEE EXAMPLE AT END OF 6.3.1.5.

6.3.1.5 EFFECT OF START COMMAND

THE EFFECT OF THE START COMMAND IS TO INITIATE THE HARDWARE PARAMETER DIALOGUE, THE SOFTWARE PARAMETER DIALOGUE, AND THEN THE DIAGNOSTIC TESTS THEMSELVES.

THE HARDWARE PARAMETER DIALOGUE COMMENCES WITH THE QUESTION "# UNITS?" TO WHICH THE OPERATOR REPLIES WITH A DECIMAL NUMBER N FROM 1 TO 16. THE TERM 'UNIT' REFERS TO THE DEVICE TO WHICH THIS SERIES OF DIAGNOSTICS IS DEDICATED. FOLLOWING THIS ARE THE QUESTIONS WHEREBY THE P-TABLES THEMSELVES WILL BE BUILT. EACH P-TABLE IS A CORE-RESIDENT TABLE CONTAINING ALL THE HARDWARE INFORMATION FOR ONE UNIT. THE OPERATOR MUST SUPPLY N (NUMBER OF UNITS) VALUES FOR EACH QUESTION. HE MAY DO THIS BY GIVING ONE ANSWER TO EACH QUESTION (IN WHICH CASE THE SERIES OF QUESTIONS WILL BE POSED N TIMES) OR BY GIVING N VALUES, SEPARATED BY COMMAS, TO EACH QUESTION (SERIES WILL BE POSED ONCE). EACH QUESTION IS FOLLOWED BY THE RESPONSE RADIX (D FOR DECIMAL, B FOR BINARY, O FOR OCTAL, L FOR YES/NO) IN PARENTHESES AND THE DEFAULT VALUE AFTER THE PARENTHESES.

FOLLOWING THE HARDWARE QUESTIONS ARE THE SOFTWARE QUESTIONS TO BUILD THE SOFTWARE TABLES, WHICH DEFINE THE MODE (QUICK VERIFY ETC.) THAT THE DIAGNOSTIC WILL EXECUTE IN.

WHEN THE QUESTION '# UNITS?' IS ANSWERED, MEMORY STORAGE IS ALLOCATED FOR THE P-TABLES, AND IF THERE IS NOT ENOUGH TO ACCOMMODATE THEM THE MESSAGE 'TOO MANY UNITS' IS ISSUED. IN THIS CASE THE DIAGNOSTIC MUST BE EXECUTED MORE THAN ONCE TO TEST ALL UNITS.

EXAMPLE:

STA/TESTS:1:2-4:6:8-10/PASS:3/FLAGS:IER:HOE=1:UAM:LOE

THIS COMMAND WILL CAUSE THREE PASSES TO BE MADE, EACH PASS CONSISTING OF TESTS 1,2,3,4,6,8,9, AND 10 EXECUTED AGAINST ALL UNITS. THERE IS NO DIFFERENCE BETWEEN SAYING <FLAG> AND SAYING <FLAG=1>. THE NOTATION <FLAG=0> IS MEANINGFUL ONLY ON A COMMAND OTHER THAN START TO CLEAR A FLAG THAT WAS PREVIOUSLY SET. NOTE THAT ON ALL COMMANDS ONLY THE FIRST THREE LETTERS ARE SCANNED.

6.3.2 RESTART COMMAND

```
*****
RES(TART)/TESTS:<TEST-LIST>/PASS:<PASS-CNT>/FLAGS:
<FLAG-LIST>/UNITS:<UNIT-LIST>
*****
```

6.3.2.1 TESTS, PASS, AND FLAGS SWITCHES

<TEST-LIST>, <PASS-CNT>, AND <FLAG-LIST> ARE AS IN THE START COMMAND.

6.3.2.2 UNITS SWITCH (/UNITS:<UNIT-LIST>)

<UNIT-LIST> IS A SEQUENCE OF DECIMAL NUMBERS (0,1 ETC.) OR RANGES OF DECIMAL NUMBERS (0-5, 8-10 ETC.) THAT SPECIFY THE UNITS TO BE TESTED. THE NUMBERS ARE SEPARATED BY COLONS. THE NUMBERS MAY RANGE FROM 0 THRU N-1 (N IS THE NUMBER OF UNITS SPECIFIED IN THE PREVIOUS START COMMAND). THE NUMBER INDICATES THE POSITION OF THE P-TABLE AS THE DATA WAS ENTERED DURING THE HARDWARE DIAGLOGUE. THE UNITS WHICH ARE SELECTED MUST NOT HAVE BEEN DROPPED BY THE DROP COMMAND. SEE THE DISCUSSION OF ADD AND DROP COMMANDS BELOW. DEFAULT IS TO TEST ALL UNITS WHICH HAVE NOT BEEN DROPPED BY A DROP COMMAND.

6.3.2.3 EFFECT OF RESTART COMMAND

THE RESTART COMMAND DIFFERS FROM THE START COMMAND IN THAT THE P-TABLES FROM THE PREVIOUS START COMMAND (THERE MUST HAVE BEEN ONE) ARE USED, INSTEAD OF NEW ONES BEING BUILT. THE UNITS SWITCH GIVES THE ABILITY TO SELECT A SUBSET OF THESE. THE SOFTWARE DIALOGUE MAY OPTIONALLY BE REEXECUTED (OPERATOR WILL BE ASKED). THE COMMAND CAN BE USED AFTER COMMAND MODE HAS BEEN REENTERED IN ANY OF THE THREE NORMAL WAYS: A) THE REQUESTED NUMBER OF PASSES HAVE BEEN MADE B) AN ERROR WAS ENCOUNTERED WITH THE HALT ON ERROR FLAG SET C) A CONTROL/C WAS ENTERED BY THE OPERATOR.

6.3.3 CONTINUE COMMAND

CON(TINUE)/PASS:<PASS-CNT/FLAGS:<FLAG-LIST>

6.3.3.1 PASS SWITCH (/PASS:<PASS-CNT>)

<PASS-CNT> IS SAME AS IN START COMMAND, BUT THE DEFAULT IS THE UNSATISFIED PASS-CNT FROM THE PREVIOUS START OR RESTART. IF NONE REMAINS, THE DEFAULT IS NON-ENDING EXECUTION.

6.3.3.2 FLAG SWITCH (/FLAGS:<FLAG-LIST>)

<FLAG-LIST> IS SAME AS IN START COMMAND, BUT UNSPECIFIED FLAGS RETAIN THEIR CURRENT VALUE.

6.3.3.3 EFFECT OF CONTINUE COMMAND

CONTINUE MUST FOLLOW A START OR RESTART, AND COMMAND MODE MUST HAVE BEEN ENTERED DUE TO A HALT ON ERROR OR A CONTROL/C. THE EFFECT OF THE COMMAND IS TO GO TO THE BEGINNING OF THE TEST THAT WAS BEING EXECUTED WHEN THE HALT OR CONTROL/C TOOK PLACE. SOFTWARE DIALOGUE MAY OPTIONALLY BE REEXECUTED. HARDWARE PARAMETERS MAY NOT BE CHANGED.

6.3.4 PROCEED COMMAND

PRO(CEED)/FLAGS:<FLAG-LIST>

6.3.4.1 FLAGS SWITCH (/FLAGS:<FLAG-LIST>)

<FLAG-LIST> IS AS IN THE START COMMAND, BUT UNSPECIFIED FLAGS RETAIN THEIR CURRENT VALUE.

6.3.4.2 EFFECT OF PROCEED COMMAND

PROCEED MUST FOLLOW A START, RESTART, OR CONTINUE. COMMAND MODE MUST HAVE BEEN ENTERED VIA A HALT ON ERROR. THE EFFECT OF THE COMMAND IS TO BEGIN EXECUTION AT THE LOCATION FOLLOWING THE ERROR CALL. NEITHER HARDWARE NOR SOFTWARE PARAMETERS MAY BE ALTERED.

6.3.5 ADD COMMAND

ADD/UNITS:<UNIT-LIST>

6.3.5.1 UNITS SWITCH (/UNITS:<UNIT-LIST>)

<UNIT-LIST> IS AS IN THE RESTART COMMAND.

6.3.5.2 EFFECT OF ADD COMMAND

THE UNITS SPECIFIED ARE ADDED TO THE TEST SEQUENCE. EACH UNIT MUST HAVE A P-TABLE IN MEMORY DUE TO AN EARLIER HARDWARE DIALOGUE. THIS COMMAND MUST BE FOLLOWED BY A RESTART OR CONTINUE. THE UNITS SWITCH MUST BE SPECIFIED. THE ADD COMMAND IS MEANINGFUL ONLY FOR UNITS THAT WERE PREVIOUSLY DROPPED.

6.3.6 DROP COMMAND

DRO(P)/UNITS:<UNIT-LIST>

6.3.6.1 UNITS SWITCH (/UNITS:<UNIT-LIST>)

<UNIT-LIST> IS AS IN THE RESTART COMMAND.

6.3.6.2 EFFECT OF DROP COMMAND

THE UNITS SPECIFIED WILL BE DROPPED FROM TESTING. THE UNITS WILL BE RESELECTED ONLY BY THE EXECUTION OF AN ADD OR START COMMAND. THE UNITS SWITCH MUST BE ENTERED. THIS COMMAND MUST BE FOLLOWED BY A RESTART OR A CONTINUE COMMAND.

6.3.7 PRINT COMMAND

PRI(NT)

6.3.7.1 EFFECT OF PRINT COMMAND

THE TOTAL NUMBER OF ERRORS FOR EACH UNIT SINCE THE LAST START OR RESTART COMMAND ARE PRINTED. THE ISR (INHIBIT STATISTICAL REPORTING) FLAG IS CLEARED.

6.3.8 DISPLAY COMMAND

DIS(PLAY)/UNITS:<UNIT-LIST>

6.3.8.1 UNITS SWITCH (/UNITS:<UNIT-LIST>)

<UNIT-LIST> IS AS IN THE RESTART COMMAND.

6.3.8.2 EFFECT OF DISPLAY COMMAND

THE HARDWARE P-TABLES FOR ALL UNITS UNDER TEST ARE PRINTED OUT IN THE FORMAT IN WHICH THEY WERE ENTERED. ANY UNITS THAT WERE DROPPED BY THE OPERATOR 'DROP' COMMAND ARE SO DESIGNATED.

6.3.9 FLAGS COMMAND

FLA(GS)

6.3.9.1 EFFECT OF FLAGS COMMAND

THE CURRENT SETTINGS OF ALL FLAGS ARE PRINTED.

6.3.10 ZFLAGS COMMAND

ZFL(AGS)

6.3.10.1 EFFECT OF ZFLAGS COMMAND

ALL FLAGS ARE CLEARED.

6.3.11 CONTROL CHARACTERS

A CONTROL C (C) ENTERED DURING THE EXECUTION OF A DIAGNOSTIC CAUSES A RETURN TO COMMAND MODE.

A CONTROL Z (Z) ENTERED DURING ONE OF THE THREE OPERATOR DIALOGUES- HARD CORE QUESTIONS (SEE 6.2), HARDWARE DIALOGUE (SEE 6.3.1.5), OR SOFTWARE DIALOGUE (SEE 6.3.1.5) CAUSES THE DEFAULTS TO BE TAKEN FOR THE REMAINDER OF THAT DIALOGUE.

A CONTROL O (O) ENTERED DURING THE EXECUTION OF A DIAGNOSTIC CAUSES ALL TELETYPE OUTPUT TO BE SUPPRESSED FOR THE REMAINDER OF THE DIAGNOSTIC OR UNTIL ANOTHER O IS TYPED, WHICH RESTORES NORMAL TELETYPE OUTPUT.

6.3.12 HARDWARE PARAMETERS

THE FOLLOWING 4 QUESTIONS WILL BE ASKED ON A START COMMAND. THE VALUE LOCATED TO THE LEFT OF THE QUESTION MARK IS THE DEFAULT VALUE THAT WILL BE TAKEN ON A CARRIAGE RETURN RESPONSE.

1. DEVICE CSR ADDRESS : (O) 160170?

THIS IS THE ADDRESS AT WHICH THE CSR REGISTERS (SELO) RESIDE ON THE UNIBUS. THE ALLOWABLE RANGE IS 160000-177776 (OCTAL), AND THE DEFAULT VALUE IS 160170.

2. DEVICE VECTOR ADDRESS : (O) 300 ?

THIS IS THE ADDRESS OF THE INPUT INTERRUPT VECTOR FOR THIS DEVICE. THE ALLOWABLE RANGE IS 000-674 (OCTAL), AND THE DEFAULT VALUE IS 300.

3. DEVICE PRIORITY LEVEL : (O) 5 ?

THIS IS THE CPU PRIORITY AT WHICH THE INTERRUPT HANDLERS OF THIS DEVICE WILL BE EXECUTED. THE ALLOWABLE RANGE IS 0-7, AND THE DEFAULT VALUE IS 5.

4. MICROPROCESSOR RUN SWITCH - TYPE 0 IF OFF, 1 IF ON : (O) 1 ?

THIS TELLS THE PROGRAM IF THE RUN SWITCH ON THE MICROPROCESSOR IS SET OR NOT. IF IT IS SET, RUN IS NOT INHIBITED, AND TESTS REQUIRING THE RUN STATE CAN BE EXECUTED. THE ALLOWABLE VALUES ARE 0 AND 1, AND THE DEFAULT VALUE IS 1 (RUN IS NOT INHIBITED).

6.3.13 SOFTWARE PARAMETERS

NO SOFTWARE PARAMETER QUESTIONS ARE ASKED BY PART 1 OF THE STATIC LOGIC TESTS.

6.3.14 EXTENDED DISCUSSION OF P-TABLE DIALOGUE

THE FULL CAPABILITY OF THE HARDWARE DIALOGUE IS REVEALED BY THE FOLLOWING DISCUSSION OF WHAT HAPPENS INTERNALLY.

AS SOON AS THE QUESTION '# UNITS?' IS ANSWERED (WITH THE NUMBER N, SAY) SPACE IN CORE IS ALLOCATED FOR N P-TABLES. ALL OF THE P-TABLES ARE OF THE SAME FORMAT, AND THERE IS A ONE-TO ONE CORRESPONDENCE BETWEEN THE HARDWARE PARAMETER QUESTIONS AND THE SLOTS IN THE P-TABLE FORMAT.

ON THE FIRST TRIP THRU THE QUESTIONS, ALL OF THE SLOTS IN ALL OF THE P-TABLES ARE FILLED. IF THE OPERATOR TYPES IN LESS THAN N EXPLICIT VALUES IN RESPONSE TO A PARTICULAR QUESTION, THESE VALUES ARE PLACED IN THE P-TABLES (ONE VALUE GOING INTO THE PROPER SLOT OF EACH P-TABLE BEGINNING WITH THE FIRST P-TABLE) UNTIL THE STRING OF VALUES IS EXHAUSTED. THE LAST VALUE IN THE STRING BECOMES THE NEW DEFAULT AND IS USED TO FILL THAT SLOT IN THE REMAINING P-TABLES.

ON SUBSEQUENT TRIPS THRU THE QUESTIONS, THE SAME PROCESS IS CARRIED OUT, EXCEPT THAT THE EARLIEST P-TABLE NOT TO HAVE RECEIVED AN EXPLICIT VALUE IN ANY OF ITS SLOTS NOW ASSUMES THE ROLE THAT TABLE NUMBER ONE PLAYED IN THE FIRST TRIP.

THE SERIES OF QUESTIONS IS REISSUED UNTIL AT LEAST ONE QUESTION HAS RECEIVED N EXPLICIT VALUES FROM THE OPERATOR.

IN GIVING A STRING OF VALUES, COMMAS WITHOUT INTERVENING VALUES MAY BE USED TO INDICATE A REPETITION OF THE LAST NAMED VALUE.

A STRING OF VALUES MAY BE GIVEN AS A RANGE (6-10 FOR EXAMPLE). IF THE VALUES REPRESENT PURE NUMERICAL DATA, THIS SAMPLE RANGE TRANSLATES TO THE STRING 6,7,8,9,10 (AN INCREMENT OF 1). IF THE VALUES ARE ADDRESSES, THE SAMPLE RANGE TRANSLATES TO THE STRING 6,8,10 (AN INCREMENT OF 2).

NOW LET US SEE HOW WE COULD USE THESE CAPABILITIES TO CONSTRUCT A SET OF P-TABLES. ASSUME THAT WE HAVE 16 UNITS, AND THAT THERE ARE THREE HARDWARE PARAMETERS FOR EACH (THREE SLOTS IN THE P-TABLE, THREE HARDWARE QUESTIONS IN THE DIALOGUE). LET THE DESIRED VALUE FOR THE FIRST PARAMETER BE THE NUMBER 75 FOR ALL 16 TABLES. LET THE DESIRED VALUE FOR THE SECOND PARAMETER BE EQUAL TO THE UNIT NUMBER (0,1,2,...,15) EXCEPT FOR UNIT 12, WHICH SHOULD RECEIVE THE VALUE 11. LET THE DESIRED VALUE FOR THE THIRD PARAMETER BE THE NUMBER 76 FOR THE FIRST 7 UNITS AND THE NUMBER 77 FOR THE LAST 9 UNITS.

THE FOLLOWING DIALOGUE WOULD ACCOMPLISH THIS GOAL:

UNITS (D) ? 16

UNIT 0

<QUESTION 1> ? 75

<QUESTION 2> ? 0-6

<QUESTION 3> ? 76

UNIT 7

<QUESTION 1> ?

<QUESTION 2> ? 7-11,,13-15

<QUESTION 3> ? 77

THE FIRST TIME THE SERIES IS ASKED, SLOT ONE RECEIVES A 75 IN ALL 16 TABLES. SLOT TWO RECEIVES THE VALUES 0,1,2,...,6 IN TABLES 0 THRU 6 AND A CONSTANT 6 IN TABLES 7 THRU 15. SLOT THREE RECEIVES A CONSTANT 76 IN ALL 16 TABLES.

THE SECOND TIME THRU THE SERIES, TABLES 7 THRU THE END ARE GOING TO BE AFFECTED (NOTE THAT THIS PIECE OF INFORMATION IS PRINTED OUT FOR THE OPERATOR IN THE FORM 'UNIT XX' AT THE BEGINNING OF EACH SERIES). QUESTION 1 IS RESPONDED TO BY A <CR>, SO SLOT ONE STAYS AT CONSTANT 75 IN TABLES 7 THRU 15, SINCE NO NEW EXPLICIT VALUES ARE TYPED IN. SLOT TWO GETS THE VALUES 7,8,9,10,11 IN TABLES 7 THRU 11, AND GETS AN 11 IN SLOT 12, AND GETS THE VALUES 13,14,15 IN TABLES 13 THRU 15. SLOT THREE GETS THE VALUE 77 IN TABLES 7 THRU 15.

THE DIALOGUE IS TERMINATED WHEN THE SOFTWARE RECOGNIZES THAT 16 EXPLICIT VALUES HAVE BEEN GIVEN FOR AT LEAST ONE QUESTION (NAMELY QUESTION 2).

7.0 DEVICE INFORMATION TABLES

* MAINTENANCE REGISTER - BSEL1

RUN = BIT7
MCLR = BIT6
STEPLU = BIT4
LULOOP = BIT3
ROMO = BIT2
ROMI = BIT1
STEPMP = BIT0

* OBUS REG 10 - TRANSMITTER BUFFER

TX7 = BIT7
TX6 = BIT6
TX5 = BIT5
TX4 = BIT4
TX3 = BIT3
TX2 = BIT2
TX1 = BIT1
TX0 = BIT0

* OBUS REG 11

OC = BIT7
GOAH = BIT3
ABORT = BIT2
EOM = BIT1
SOM = BIT0

```

:*****
* OBUS REG 12
:*****
IC      = BIT7
BPOLL   = BIT6
LULP    = BIT5

:*****
* OBUS REG 13
:*****
POLL    = BIT7
DTR     = BIT6
SEIFR   = BIT5
HDX     = BIT4
MAINT1  = BIT3
MAINT2  = BIT2
SELSBY  = BIT1

:*****
* OBUS REG 14
:*****
TXEN    = BIT6
DISSI   = BIT5
RDAX    = BIT4
WAX     = BIT3
ENAX    = BIT2
AX2     = BIT1
AX1     = BIT0

:*****
* OBUS REG 17
:*****
CRC2    = BIT7
CRC1    = BIT6
IDLE    = BIT5
SECA    = BIT4
STRIP   = BIT3
RDALL   = BIT2
IERR    = BIT1
DDCMP   = BIT0

:*****
* IBUS REG 10 - RECEIVER BUFFER
:*****
RX7     = BIT7
RX6     = BIT6
RX5     = BIT5
RX4     = BIT4
RX3     = BIT3
RX2     = BIT2
RX1     = BIT1
RX0     = BIT0

```

```
*****
* IBUS REG 11
*****
OC      = BIT7
OACT    = BIT6
SW3     = BIT5
ORDY    = BIT4
SW2     = BIT3
SW1     = BIT2
SW0     = BIT1
UNRR    = BIT0
```

```
*****
* IBUS REG 12
*****
IC      = BIT7
IACT    = BIT6
LULP    = BIT5
IRDY    = BIT4
OVR     = BIT3
RAB     = BIT2
EBLK    = BIT1
BCC     = BIT0
```

```
*****
* IBUS REG 13
*****
RING    = BIT7
DTR     = BIT6
RTS     = BIT5
HDX     = BIT4
MODR    = BIT3
CS      = BIT2
STBY    = BIT1
CARR    = BIT0
```

```
*****
* IBUS REG 14
*****
READY   = BIT7
TXEN    = BIT6
DISSI   = BIT5
RDAX    = BIT4
WAX     = BIT3
ENAX    = BIT2
AX2     = BIT1
AX1     = BIT0
```

```
*****
* IBUS REG 17
*****
SIGH    = BIT7
SIGQ    = BIT6
TXDATA  = BIT5
OCOR    = BIT4
ICIR    = BIT3
TESTMD  = BIT2
MCLK    = BIT1
DDCMP   = BIT0
```



```

:*****
* AX0-15 - USYRT REG 0 (READ ONLY)
:*****
RX7    = BIT7
RX6    = BIT6
RX5    = BIT5
RX4    = BIT4
RX3    = BIT3
RX2    = BIT2
RX1    = BIT1
RX0    = BIT0

```

```

:*****
* AX0-16 - USYRT REG 1 (READ ONLY)
:*****
RERR   = BIT7
ASBC2  = BIT5
ASBC1  = BIT5
ASBC0  = BIT4
ROR    = BIT3
RABT   = BIT2
REOM   = BIT1
RSOM   = BIT0

```

```

:*****
* AX1-15 - USYRT REG 2
:*****
TX7    = BIT7
TX6    = BIT6
TX5    = BIT5
TX4    = BIT4
TX3    = BIT3
TX2    = BIT2
TX1    = BIT1
TX0    = BIT0

```

```

:*****
* AX1-16 - USYRT REG 3
:*****
TERR   = BIT7
TXGA   = BIT3
TXAB   = BIT2
TEOM   = BIT1
TSOM   = BIT0

```

```

:*****
* AX2-15 - USYRT REG 4
:*****
SYN7   = BIT7
SYN6   = BIT6
SYN5   = BIT5
SYN4   = BIT4
SYN3   = BIT3
SYN2   = BIT2
SYN1   = BIT1
SYN0   = BIT0
SYNCH  = 226

```

```

:*****
* AX2-16 - USYRT REG 5
:*****
APA      = BIT7
DDC      = BIT6
STR      = BIT5
SEC      = BIT4
IDL      = BIT3
CRCTY2   = BIT2
CRCTY1   = BIT1
CRCTY0   = BIT0

```

```

:*****
* AX3-15 - USYRT REG 6
:*****
I422     = BIT7
XYZ      = BIT6
C32BCC   = BIT5
V35      = BIT4
INTGRL   = BIT3
C32ENB   = BIT2
OP       = BIT1
TEST     = BIT0
AX315U   = I422!XYZ!C32BCC.V35.INTGRL OP

```

```

:*****
* AX3-16 - USYRT REG 7
:*****
TXLEN2   = BIT7
TXLEN1   = BIT6
TXLENO   = BIT5
RXLEN2   = BIT2
RXLEN1   = BIT1
RXLENO   = BIT0

```

8.0 TEST DESCRIPTIONS

```

:*****
TEST 1 - MICROPROCESSOR CSR ADDRESSING TEST (SELO)
*
* THIS TEST ADDRESSES THE FIRST MICROPROCESSOR CSR (SELO), TO MAKE SURE
* THAT A NON-EXISTENT MEMORY TIME-OUT TRAP DOES NOT OCCUR WHILE
* ATTEMPTING TO ADDRESS THE MICROPROCESSOR.
:*****

```

```

:*****
TEST 2 - INBUS/OUTBUS REG 14 INITIALIZATION TEST
*
* MASTER CLEAR (MCLR) IS SET IN THE MICROPROCESSOR, IBUS REG 14 IS READ
* AND COMPARED TO 200.
:*****

```

```

*****
TEST 3 - INBUS/OUTBUS REG 14 READ/WRITE BIT TEST
*
* WRITE, READ, AND COMPARE ALL WORDS OF DATA PATTERN A INTO REG 14,
* A BYTE AT A TIME. NON-R/W BITS ARE MASKED OFF TO 0 BEFORE WRITING AND
* READING.
* DATA PATTERN A = 125,252,000,377,001,002,004,010,020,040,100,200,376,
*                   375,373,367,357,337,277,177.
*****

```

```

*****
TEST 4 - REG 14 MASTER CLEAR TEST
* WRITE 377 INTO REG 14, ISSUE MASTER CLEAR, READ REG 14 AND COMPARE
* TO 200.
*****

```

```

*****
TEST 5 - REG 14 UNIBUS RESET (INIT) TEST
* WRITE 377 INTO REG 14, ISSUE UNIBUS RESET (INIT), READ REG 14 AND COMPARE
* TO 200.
*****

```

```

*****
TEST 6 - LINE UNIT FALSE SELECTION TEST
*
* FIRST, A MASTER CLEAR IS PERFORMED. THEN, THE PROGRAM SINGLE-STEPS THE
* MICROPROCESSOR THROUGH AN INSTRUCTION WHICH LOADS 041 (OCT) INTO THE MAR
* REGISTER (OBUS* ADRS 14). THEN, THE LINE UNIT REGISTER 14 IS READ AND CHECKED
* TO BE UNAFFECTED (STILL = 0). THIS TEST IS INTENDED TO DETECT A FALSE
* SELECTION OF THE LINE UNIT REGISTERS, WHEN THE LINE UNIT IS NOT BEING
* ACCESSED.
*****

```

```

*****
TEST 7 - INBUS REG MASTER CLEAR TEST
*
* FIRST, ALL READ/WRITE BITS OF REGS 10-17 ARE SET BY LOADING A
* DIFFERENT WORD OF PATTERN G INTO EACH REG. THEN,
* A MASTER CLEAR IS ISSUED AND EACH REG IS READ AND COMPARED TO A WORD OF
* PATTERN M. WHICH CONTAINS THE INITIALIZED STATES OF THE REGS. (UNPREDICTABLE
* BITS ARE MASKED OFF TO 0 BEFORE COMPARISON).
* PATTERN G = 000,000,240,120,177,000,000,001
* PATTERN M = 000,020,000,000,200,000,000,051
*****

```

```

;*****
TEST 8 - REGISTER 10-17 ADDRESSING TEST
*
* FIRST, A MASTER CLEAR IS ISSUED. THEN,
* WRITE A DIFFERENT WORD OF DATA PATTERN B INTO EACH OF REGS 10-17,
* AND AFTER EACH WRITE, READ AND COMPARE ALL REGS TO EXPECTED VALUES.
* UNPREDICTABLE BITS ARE MASKED OFF TO 0 WHEN READ FOR COMPARISON.
* PATTERN B = 000,000,040,100,220,000,000,051
;*****

```

```

;*****
TEST 9 - REG 11 READ/WRITE BIT TEST
*
* WRITE, READ, AND COMPARE EACH WORD OF DATA PATTERN C INTO REG 11 :
* DATA PATTERN C = 020,020,020.
;*****

```

```

;*****
TEST 10 - REG 12 READ/WRITE BIT TEST
*
* WRITE, READ, AND COMPARE EACH WORD OF DATA PATTERN D INTO REG 12 :
* DATA PATTERN D = 000,040,000.
;*****

```

```

;*****
TEST 11 - REG 13 READ/WRITE BIT TEST
*
* WRITE, READ, AND COMPARE EACH WORD OF DATA PATTERN E INTO REG 13 :
* DATA PATTERN E = 000,120,020,100,120,000.
;*****

```

```

;*****
TEST 12 - REG 17 READ/WRITE BIT TEST
*
* WRITE, READ, AND COMPARE EACH WORD OF DATA PATTERN F INTO REG 17 :
* DATA PATTERN F = 050,051,050.
;*****

```



```

*****
TEST 13 - MAINTENANCE CLOCK BIT TEST
*
* FIRST, A MASTER CLEAR IS ISSUED TO INIT ALL REGS. THEN, THE MICROPROCESSOR
* IS PLACED IN A LOOP ON AN INSTRUCTION, BY SETTING THE INSTRUCTION IN SEL6
* AND SETTING ROMI AND RUN IN BSEL1. THE INSTRUCTION IS ONE WHICH REPETITIVELY
* READS LINE UNIT REG 17 INTO BSEL2. THE PDP-11 CAN THEN SCAN BSEL2 TO MONITOR
* THE MAINTENANCE CLOCK BIT, MCLK. THE FOLLOWING SEQUENCE IS THEN PERFORMED
* TO MONITOR MCLK :
* - THE PROGRAM REPEATEDLY CHECKS THE MCLK BIT FOR THE 1 STATE, AND IF IT IS
* NOT FOUND WITHIN SEVERAL HUNDRED MILLI-SEC (DEPENDING ON THE PROCESSOR)
* AN ERROR IS REPORTED. (THE MAINTENANCE CLOCK HAS A PERIOD OF 41.6 MICRO-
* SEC).
* - THE PROGRAM NEXT REPEATEDLY CHECKS THE MCLK BIT FOR THE 0 STATE, AND IF
* IT IS NOT FOUND WITHIN SEVERAL HUNDRED MILLI-SEC AN ERROR IS REPORTED.
* - THE PROGRAM NEXT REPEATEDLY CHECKS MCLK BIT FOR THE 1 STATE AGAIN, AND
* IF IT IS NOT FOUND WITHIN SEVERAL HUNDRED MILLI-SEC, AN ERROR IS REPORTED.
*
* IF THE P-TABLE FOR THIS UNIT INDICATES THAT THE MICROPROCESSOR RUN SWITCH
* IS OFF, THE TEST WILL BE SKIPPED.
*****

```

```

*****
TEST 14 - EXTENDED REGISTER MASTER CLEAR TEST
*
* FIRST, ALL READ/WRITE BITS OF EXTENDED REGS AX0-AX3 ARE SET BY LOADING
* A DIFFERENT WORD OF PATTERN H INTO EACH REG. THEN, A MASTER CLEAR IS
* ISSUED AND EACH REG IS READ AND COMPARED TO A WORD OF PATTERN I, WHICH
* CONTAINS THE INITIALIZED STATES OF ALL THE EXTENDED REGS.
* PATTERN H = 000,000,377,017,377,377,375,377
* PATTERN I = 000,000,000,000,000,103,000,000
*****

```

```

*****
TEST 15 - EXTENDED REGISTER ADDRESSING TEST
*
* FIRST, ISSUE A MASTER CLEAR TO PUT REGS INTO INITIALIZED STATES SHOWN IN
* PATTERN I. THEN, WRITE A DIFFERENT WORD OF PATTERN J INTO EACH EXTENDED (AX)
* REG, AND AFTER EACH WRITE, READ AND COMPARE ALL EXTENDED REGS TO EXPECTED
* VALUES.
* PATTERN I = 000,000,000,000,000,103,000,000
* PATTERN J = 000,000,001,002,004,103,040,100
*****

```

```

*****
TEST 16 - REGS 15,16 / AX2-15,AX2-16 READ/WRITE BIT TEST
*
* USING REGS 15,16, THE INDIRECT REGS AX2-15,AX2-16 (USVRT REGS 4,5) ARE
* WRITTEN AND READ USING EACH WORD OF PATTERN K. AX2-15 IS COMPARED
* TO THE WORD WRITTEN, AND AX2-16 IS ALWAYS COMPARED TO 103. (AX2-16 IS NOT
* WRITEABLE).
* PATTERN K =
*   FOR REG 15: 000,377,125,252,001,002,004,010,020,040,100,200,000,000,
*               000,000,000,000,000,000,376,375,373,367,357,337,277,177,
*               377,377,377,377,377,377,377,377
*   FOR REG 16: 000,377,125,252,000,000,000,000,000,000,000,000,001,002,
*               004,010,020,040,100,200,377,377,377,377,377,377,377,377,
*               376,375,373,367,357,337,277,177.
*****

```

```

*****
TEST 17 - AX0-15,AX0-16 READ/WRITE BIT TEST
*
* IN THIS TEST, A MASTER CLEAR IS DONE, AND THEN A WRITE, READ, AND COMPARE
* ARE PERFORMED IN REGS AX0-15,AX0-16 USING EACH WORD OF PATTERN L.
* ANY BITS IN AX0-15,AX0-16 WHICH ARE NOT READ/WRITE ARE MASKED OFF (TO 0)
* IN THE EXPECTED VALUE BEFORE COMPARISON.
* PATTERN L =
*   FOR REG 15: 000,377,000
*   FOR REG 16: 000,377,000.
*****

```

```

*****
TEST 18 - AX1-15,AX1-16 READ/WRITE BIT TEST
*
* IN THIS TEST, A MASTER CLEAR IS DONE, AND THEN A WRITE, READ, AND COMPARE
* ARE PERFORMED IN REGS AX1-15,AX1-16 USING EACH WORD OF PATTERN K.
* ANY BITS IN AX1-15,AX1-16 WHICH ARE NOT READ/WRITE ARE MASKED OFF (TO 0)
* IN THE EXPECTED VALUE BEFORE COMPARISON.
*****

```

```

;*****
TEST 19 - AX3-15,AX3-16 READ/WRITE BIT TEST
*
* IN THIS TEST A MASTER CLEAR IS DONE AND THEN A WRITE, READ, AND COMPARE ARE
* PERFORMED IN REGS AX3-15,AX3-16 USING EACH WORD OF PATTERN V FOR WRITING,
* AND PATTERN U FOR COMPARING.
* ANY BITS IN AX3-15,AX3-16 WHICH ARE NOT READ/WRITE ARE MASKED OFF (TO 0)
* IN THE EXPECTED VALUE BEFORE COMPARISON.
* PATTERN V =
*   FOR REG 15 : 000,333,331,323,313,233,133,000,000,000,000,
*                000,000,000,000,000,000,000,000
*   FOR REG 16 : 000,000,000,000,000,000,000,001,002,004,040,
*                100,200,346,345,343,307,247,147
* PATTERN U
*   FOR REG 15 : 000,001,013,011,021,101,301,000,000,000,000,
*                000,000,000,000,000,000,000,000
*   FOR REG 16 : 000,000,000,000,000,000,000,001,002,004,040,
*                100,200,346,345,343,307,247,147
;*****

```

```

;*****
TEST 20 - REG 17 - AX2-16 READ/WRITE, MASTER CLEAR TEST
*
* THIS TEST CONSISTS OF 2 SUBTESTS. IN THE FIRST SUBTEST, EACH BYTE OF PAT O
* IS WRITTEN INTO REG 17 AND AFTER EACH WRITE, AX2-16 IS READ AND COMPARED
* TO A BYTE OF PAT P.
* PATTERN O = 000,041,004,010,020,040,100,101,200,201,300,111,301,375
* PATTERN P = 000,113,200,040,020,010,001,104,007,105,007,144,107,157
* IN THE SECOND SUBTEST, REG 17 IS LOADED WITH 375, A MASTER CLEAR IS ISSUED,
* AND AX2-16 IS COMPARED TO ITS INITIALIZED STATE (103).
;*****

```

```

;*****
TEST 21 - TRANSMITTER BUFFER DATA TEST
* A MASTER CLEAR IS DONE FIRST, AND THEN A BYTE OF PATTERN N IS LOADED INTO
* REG 11 AND THE NEXT BYTE IS LOADED TWICE INTO REG 10. THE PROGRAM THEN WAITS
* AT LEAST 50 MICRO-SEC, AND THEN IT READS AND COMPARES AX1-15 TO THE BYTE WHICH
* WAS LOADED INTO REG 10, AND IT READS AND COMPARES AX1-16 TO THE BYTE WHICH WAS
* LOADED INTO REG 11. THIS PROCESS IS REPEATED (INCLUDING THE MASTER CLEAR)
* FOR EACH PAIR OF BYTES IN PATTERN N.
* PATTERN N =
*   FOR REG 10: 000,125,252,377,000,000,000
*   FOR REG 11: 000,000,000,000,005,012,017
;*****

```

```

*****
TEST 22 - TRANSMITTER BUFFER SEQUENCING TEST
*
* FIRST, A MASTER CLEAR IS DONE, AND THE PROGRAM CHECKS FOR ORDY=1, OCOR=0.
* THEN, 2 TSOM CHARS ARE LOADED INTO THE TX SILO, AND ALLOWED TO RIPPLE
* DOWN TO THE OUTPUT. THE PROGRAM CHECKS FOR ORDY=1, OCOR=1.
* NEXT, THE PROGRAM CYCLES THE STEPLU BIT UNTIL OCOR=0 AGAIN, AND CHECKS FOR
* THIS TO OCCUR WITHIN 3 CYCLES.
* THE SILO IS THEN FILLED WITH 64 BYTES OF A 256-BYTE BINARY COUNT PATTERN
* (000-377) AND THE PROGRAM CHECKS FOR ORDY=0 AFTER THE 64TH CHAR IS LOADED.
* THE PROGRAM CYCLES STEPLU FOR 8 CYCLES AND CHECKS THAT AFTER THE 8TH, ORDY=1.
* AX1-15 IS READ AND COMPARED TO EXPECTED DATA.
* THE REST OF THE BINARY COUNT DATA BYTES ARE LOADED, CYCLED 8 CLOCKS, READ AND
* COMPARED, A BYTE AT A TIME. UPON COMPLETION, THE SILO IS CHECKED TO BE EMPTY
* WITH ORDY=1, OCOR=0.
*****

```

```

*****
TEST 23 - TX MSG TIMING TEST, CHAR MODE, WITH CRC
*
* IN THIS TEST, AN ENTIRE MESSAGE IS TRANSMITTED (USING STEPLU AND LULOOP)
* AND THE PROGRAM MONITORS THE OCCURRENCE OF USYRT TX BUFFER EMPTY FLAGS
* (BY SCANNING ORDY AND OCOR) AND OACT, AT EACH STEP. THE TEST IS DONE IN
* CHARACTER ORIENTED PROTOCOL MODE, USING 8-BIT CHARS AND CRC-16.
* THE FOLLOWING STEPS ARE DONE:
* A MASTER CLEAR IS DONE, AND THE LINE UNIT IS PLACED IN CHAR MODE.
* SOM IS SET TWICE TO SEND 2 SYNCH CHARS. THEN, 2 000 CHARS ARE SENT, AND
* THEN 2 TERMINATING SYNCHS ARE SENT.
* THE TRANSMITTER IS THEN DISABLED, USING OC, AND OACT IS MONITORED FOR THE
* CLEARED STATE AFTER THE 3RD SYNCH COMPLETES.
*****

```

```

*****
TEST 24 - TX MSG TIMING TEST, BIT MODE, WITH CRC
*
* IN THIS TEST, AN ENTIRE MESSAGE IS TRANSMITTED (USING STEPLU AND LULOOP)
* AND THE PROGRAM MONITORS THE OCCURRENCE OF USYRT TX BUFFER EMPTY FLAGS
* (BY SCANNING ORDY AND OCOR) AND OACT, AT EACH STEP. THE TEST IS DONE IN
* BIT ORIENTED PROTOCOL MODE, USING 8-BIT CHARS AND CRC-CCITT-1.
* THE FOLLOWING STEPS ARE DONE:
* A MASTER CLEAR IS DONE, AND THE LINE UNIT IS PLACED IN BIT MODE.
* SOM IS SET TWICE TO SEND 2 FLAG CHARS. THEN, 2 000 CHARS ARE SENT, AND
* THEN 2 TERMINATING FLAGS ARE SENT.
* THE TRANSMITTER IS THEN DISABLED, USING OC, AND OACT IS MONITORED FOR THE
* CLEARED STATE.
*****

```



```

*****
TEST 25 - TX MSG TIMING TEST, CHAR MODE, WITH NO CRC
*
* IN THIS TEST, AN ENTIRE MESSAGE IS TRANSMITTED (USING STEPLU AND LULOOP)
* AND THE PROGRAM MONITORS THE OCCURRENCE OF USYRT TX BUFFER EMPTY FLAGS
* (BY SCANNING OPDY AND OCOR) AND OACT, AT EACH STEP. THE TEST IS DONE IN
* CHARACTER ORIENTED PROTOCOL MODE, USING 8-BIT CHARS AND NO ERROR CHECKING.
* THE FOLLOWING STEPS ARE DONE:
* A MASTER CLEAR IS DONE, AND THE LINE UNIT IS PLACED IN CHAR MODE.
* SOM IS SET TWICE TO SEND 2 SYNCH CHARS. THEN, 2 000 CHARS ARE SENT, AND
* THEN 2 TERMINATING SYNCHS ARE SENT.
* THE TEST IS PERFORMED WITH TXEN (REG 14, BIT6) SET, AND THE PROGRAM CHECKS
* THAT THIS HOLDS RTS HIGH PAST THE END OF THE MESSAGE.
* THE TRANSMITTER IS THEN DISABLED, USING OC, AND OACT IS MONITORED FOR THE
* CLEARED STATE.
*****

```

```

*****
TEST 26 - TX UNDERRUN SET AND CLEAR TEST - CHAR MODE
*
* IN THIS TEST, A TX UNDERRUN ERROR IS FORCED IN EACH OF 2 SITUATIONS,
* AND THEN CLEARED DIFFERENTLY IN EACH.
* IN THE FIRST, A MESSAGE IS INITIATED, A 000 CHAR IS SENT, AND THE TX
* BUFFER IS NOT SERVICED IN RESPONSE TO THE USYRT TX BUFFER EMPTY FLAG,
* WHICH CAUSES UNRR TO SET IN REG 11. THEN, SOM IS SET TO CLEAR THE ERROR,
* AND THIS IS VERIFIED.
* IN THE SECOND SITUATION, A MSG IS INITIATED, A 000 CHAR IS SENT, AND THE TX
* BUFFER IS NOT SERVICED IN RESPONSE TO THE USYRT TX BUFFER EMPTY FLAG, WHICH
* AGAIN CAUSES UNRR TO SET. THEN, A MASTER CLEAR IS DONE, AND THE UNRR BIT
* IS CHECKED TO BE CLEARED.
*****

```

```

*****
TEST 27 - TRANSMIT CHAR LENGTH TIMING TEST - CHAR MODE, CRC
*
* THE LINE UNIT IS PLACED IN CHAR MODE (DDCMP) AND A MESSAGE IS INITIATED
* WITH AN 8-BIT SYNCH AND A 5-BIT SYNCH CHAR. NEXT, A 000 CHAR IS SENT WITH
* EACH OF THE FOLLOWING TX CHAR LENGTHS : 5 BITS, 6 BITS, 7 BITS, 8 BITS.
* (FOR EXAMPLE, A 5-BIT CHAR REQUIRES 5 CLOCK CYCLES TO BE TRANSMITTED). TWO
* TERMINATING SYNCHS ARE SENT AFTER THE DATA.
*****

```

```

*****
TEST 28 - TRANSMIT CHAR LENGTH TIMING TEST - BIT MODE, CRC
*
* THE LINE UNIT IS PLACED IN BIT MODE AND A MESSAGE IS INITIATED
* WITH 2 FLAG CHARS. NEXT, 2 8-BIT 000 CHARS ARE SENT, FOLLOWED BY 000 CHARS
* WITH EACH OF THE FOLLOWING TRANSMITTER CHAR LENGTHS:
* 1 BIT, 2 BITS, 3 BITS, 4 BITS, 5 BITS, 6 BITS, 7 BITS, AND 8 BITS.
* (FOR EXAMPLE, A 5-BIT CHAR REQUIRES 5 CLOCK CYCLES TO BE TRANSMITTED).
* TWO TERMINATING FLAGS ARE SENT AFTER THE DATA.
*****

```

```

*****
TEST 29 - TXDATA BIT TEST - CHAR MODE, CRC
*
* THE LINE UNIT IS INITIALIZED AND A MSG IS INITIATED (USING STEPLU) WITH CRC-
* 16 SELECTED IN CHAR MODE. TWO SYNCHS, 000,125,252,377,000, AND 2 TERMINATING
* SYNCHS ARE THEN SENT. THE PROGRAM CHECKS EACH BIT OF THE TRANSMITTED
* DATA CHARS, BY MONITORING TXDATA (REG 17) AS THE DATA IS CLOCKED OUT OF
* THE USYRT TRANSMITTER.
*****

```

```

*****
TEST 30 - USYRT RECEIVER MSG TEST - CHAR MODE, CRC
*
* THE LINE UNIT IS INITIALIZED AND A MESSAGE IS INITIATED (USING STEPLU) WITH
* LULP (REG 12) SET TO LOOP THE DATA INTERNALLY IN THE USYRT, AND WITH CRC-16
* SELECTED. TWO SYNCHS, 000,125,252,377,000, AND FOUR TERMINATING SYNCHS ARE
* SENT. THE PROGRAM MONITORS IACT, AND THE RCV'D CHARS AND CRC BYTES ARE READ
* FROM AX0-15 AND COMPARED TO EXPECTED VALUES. THE PROGRAM THEN CHECKS FOR IACT
* STILL SET AFTER THE MESSAGE.
*****

```

```

*****
TEST 31 - USYRT RECEIVER MSG TEST - BIT MODE, CRC
*
* THE LINE UNIT IS INITIALIZED AND A MESSAGE IS INITIATED (USING STEPLU) WITH
* LULP (REG 12) SET TO LOOP THE DATA INTERNALLY IN THE USYRT, AND WITH CRC-
* CCITT-1. TWO FLAGS, 000,125,252,377,000, AND TWO TERMINATING FLAGS ARE THEN
* SENT. THE PROGRAM MONITORS IACT, RSOM, AND THE RCV'D CHARS ARE READ
* FROM AX0-15 AND COMPARED TO EXPECTED VALUES. THE PROGRAM THEN CHECKS FOR
* IACT = 0, SETS IC TO CLEAR THE RECEIVER, AND CHECKS FOR IACT STILL = 0.
*****

```

```

*****
TEST 32 - USYRT RECEIVER MSG TEST - CHAR MODE, NO CRC
*
* THE LINE UNIT IS INITIALIZED AND A MESSAGE IS INITIATED (USING STEPLU) WITH
* LULP (REG 12) SET TO LOOP THE DATA INTERNALLY IN THE USYRT, AND WITH NO
* ERROR DETECTION. TWO SYNCHS, 000,125,252,377,000, AND TWO SYNCHS ARE
* THEN SENT. THE PROGRAM MONITORS IACT, AND THE RECEIVED CHARS ARE READ FROM
* AX0-15 AND COMPARED TO EXPECTED VALUES. THE PROGRAM THEN CHECKS FOR IACT
* STILL = 0, SETS IC TO CLEAR THE RECEIVER, AND CHECKS FOR IACT = 0.
*****

```

```

*****
TEST 33 - USYRT RECEIVER MSG TEST - BIT MODE, NO CRC
*
* THE LINE UNIT IS INITIALIZED AND A MESSAGE IS INITIATED (USING STEPLU) WITH
* LULP (REG 12) SET TO LOOP THE DATA INTERNALLY IN THE USYRT, AND WITH ERROR
* DETECTION INHIBITED. TWO FLAGS, 000,125,252,377,000, AND TWO TERMINATING FLAGS
* ARE THEN SENT. THE PROGRAM MONITORS IACT, RSOM, AND THE RCV'D CHARS ARE
* READ FROM AX0-15 AND COMPARED TO EXPECTED VALUES. THE PROGRAM THEN CHECKS FOR
* IACT = 0, SETS IC TO CLEAR THE RECEIVER, AND CHECKS FOR IACT STILL = 0.
*****

```

```

*****
TEST 34 - SILO-DISABLED TRANSMITTER LOAD TEST
*
* THIS TEST DISABLES THE SILOS, LOADS A 125 CHARACTER INTO THE TX SILO, AND
* READS AX1-15 AND CHECKS THAT THE DATA DID NOT GET LOADED INTO THE USYRT TX
* BUFFER.
*****

```

```

*****
TEST 35 - SILO-DISABLED MESSAGE TEST - BIT MODE, NO CRC
*
* THE LINE UNIT IS INITIALIZED AND A MESSAGE IS INITIATED (USING STEPLU)
* WITH LULP (REG 12) SET TO LOOP THE DATA INTERNALLY IN THE USYRT, WITH SILO
* DISABLE SET, AND WITH NO ERROR DETECTION. TWO FLAGS, 000,125,252, AND
* TERMINATING FLAGS ARE THEN SENT BY LOADING THE TRANSMITTED CHARS INTO
* REG AX1. THE PROGRAM MONITORS OACT, IACT, RSOM, REOM, ORDY, OCOR, ICIR,
* IRDY, AND THE RECEIVED CHARS ARE READ FROM AX0 AND COMPARED TO EXPECTED
* VALUES.
*****

```

```

*****
TEST 36 - RECEIVER BUFFER TEST - CHAR MODE, CRC
*
* FIRST, A MASTER CLEAR IS DONE AND THE PROGRAM CHECKS FOR ICIR = 1 AND IRDY
* - 0. THEN, 2 SOM CHARS ARE LOADED AND CLOCKED INTO THE USYRT, AND 64
* BYTES OF A 256-BYTE BINARY COUNT DATA PATTERN (000-377) ARE LOADED INTO
* THE TX SILO.
* THE LINE UNIT IS THEN CLOCKED UNTIL IRDY = 1, AND THE PROGRAM CHECKS FOR
* THIS TO OCCUR WITHIN 40-43 CYCLES. THE PROGRAM READS THE RCV SILO, CHECKS THE
* CHAR FOR 000, AND CHECKS FOR IRDY = 0 AGAIN.
* THE LINE UNIT IS THEN CLOCKED IN GROUPS OF 8 CYCLES, AND AFTER EACH, THE
* PROGRAM CHECKS FOR ICIR = 1, IRDY = 1, UNTIL THE 64TH GROUP, AFTER WHICH
* IT CHECKS FOR ICIR = 0, IRDY = 1. THE SECOND DATA CHAR IS READ FROM THE
* RECEIVER SILO AND COMPARED TO 001. THEN, THE PROGRAM CHECKS FOR ICIR = 1,
* IRDY = 1 AGAIN.
* THE REST OF THE BINARY COUNT DATA BYTES ARE CYCLED 8 CLOCKS AND READ AND
* COMPARED A BYTE AT A TIME.
*****

```

```

*****
TEST 37 - RECEIVER CHAR LENGTH TIMING TEST - CHAR MODE, NO CRC
*
* THE LINE UNIT IS PLACED IN CHAR MODE, WITH NO ERROR DETECTION, AND A MSG IS
* INITIATED WITH 2 SYNCH CHARS. NEXT, FIFTEEN 000 CHARS ARE LOADED INTO THE
* TRANSMITTER SILO. THE LINE UNIT IS THEN CLOCKED USING STEPLU WITH LULOOP
* SET, WHILE THE RECEIVER CHAR LENGTH IS SET TO THE FOLLOWING VALUES : 5,6,7,8.
* FOR EACH RCV CHAR LENGTH, THE PROGRAM CHECKS TO MAKE SURE THET USYRT RECEIVER
* FLAGS OCCUR THE PROPER NO. OF CYCLES APART, FOR EACH RCV CHAR LENGTH.
* (FOR EXAMPLE A 5-BIT CHAR TAKES 5 CLOCK CYCLES TO BE RECEIVED). A MASTER
* CLEAR IS THEN DONE TO TERMINATE THE OPERATION.
*****

```

```

*****
TEST 38 - RECEIVER CHAR LENGTH TIMING TEST - BIT MODE, NO CRC
*
* THE LINE UNIT IS PLACED IN BIT MODE WITH NO ERROR DETECTION, AND A MESSAGE IS
* INITIATED WITH 2 FLAG CHARS. NEXT, FIFTEEN 000 CHARS ARE LOADED INTO THE
* TRANSMITTER SILO. THE LINE UNIT IS THEN CLOCKED USING STEPLU WITH LULOOP
* SET, WHILE THE RCV CHAR LENGTH IS SET TO THE FOLLOWING VALUES : 8,8,8,7,6,5,
* 4,3,2,1. FOR EACH RCV CHAR LENGTH, THE PROGRAM CHECKS TO MAKE SURE THAT THE
* USYRT RECEIVER FLAGS OCCUR THE PROPER NO. OF CYCLES APART, FOR EACH RCV
* CHAR LENGTH. (FOR EXAMPLE, A 5 BIT CHAR TAKES 5 CLOCK CYCLES TO BE RECEIVED).
* A MASTER CLEAR IS THEN DONE TO TERMINATE THE OPERATION.
*****

```

```

*****
TEST 39 - TRANSMITTER UNDERRUN ERROR, IDLE MARKING, CHAR MODE,NO CRC
*
* THE LINE UNIT IS PLACED IN CHAR MODE, AND THE IDLE BIT IS SET. THEN, A
* MSG IS INITIATED, A 000 CHAR IS SENT, AND THE TX BUFFER IS NOT SERVICED
* IN RESPONSE TO THE USYRT TX BUFFER EMPTY FLAG, WHICH CAUSES A TX UNDERRUN
* ERROR. THEN, THE RECEIVER IS CLOCKED AND CHECKED FOR TWO 377 CHARS TO BE
* RECEIVED (LINE MARKING) BEFORE SHUTTING DOWN WITH A MASTER CLEAR.
*****

```

```

*****
TEST 40 - MSG TERMINATION WITH GA CHARS - BIT MODE,NO CRC
*
* THE DEVICE IS ENABLED FOR TRANSMIT AND RECEIVE, AND A MESSAGE IS
* INITIATED IN BIT MODE.
* 2 FLAG CHARACTERS ARE SENT, FOLLOWED BY
* THE FOLLOWING DATA CHARACTERS : 000, 125, 252, 377, 000. THEN THE LOOP
* MODE BIT (STRIP) IS SET AND 2 TERMINATING GO-AHEAD CHARACTERS ARE
* SENT. EACH USYRT RCV FLAG IS TIMED AS IT IS RECEIVED, AND THE 5 DATA
* WORDS ARE READ AND COMPARED TO EXPECTED VALUES.
* ALSO, THE FIRST GA CHAR IS CHECKED BY SCANNING THE TXDATA BIT AS THE GA
* IS BEING TRANSMITTED (GA = 376 OCTAL).
* THE TEST ALSO CHECKS FOR SETTING OF RAB AND EBLK
* IN LOOP MODE.
*****

```

```

*****
TEST 41 - IDLE SYNCHS TEST - CHAR MODE
*
* THE DEVICE IS ENABLED FOR TRANSMIT AND RECEIVE, AND A MESSAGE IS
* INITIATED IN CHAR MODE. 24(DEC) SYNCHS ARE SENT.
* EACH SYNCH IS TIMED AS IT IS RECEIVED, AND THE BITS ARE CHECKED
* FOR A VALID SYNCH CHAR FOR EACH OF THE 22 SYNCHS WHICH FOLLOW
* THE FIRST TWO (THESE PERFORM SYNCHRONIZATION, AND ARE NOT READ).
* WHILE THE LAST SYNCH IS BEING TRANSMITTED, OC IS SET, AND THE
* NEXT CHAR RCV'D AFTER THE SYNCH IS CHECKED TO BE 377 (LINE MARKING).
* THEN, A MASTER CLEAR IS ISSUED.
* THE SYNCH CHAR USED IS 226 (OCTAL).
*****

```

```

*****
TEST 42 - STRIP SYNCH TEST
*
* THE DEVICE IS ENABLED FOR TRANSMIT AND RECEIVE, AND A MESSAGE IS
* INITIATED IN CHAR MODE AND WITH THE STRIP SYNCH
* BIT SET. THEN 24 (DEC) SYNCHS ARE SENT
* FOLLOWED BY THE FOLLOWING DATA CHARACTERS : 377, 000, 125, 252,
* AND THEN 2 TERMINATING SYNCHS.
* EACH OF THE 23 SYNCHS AFTER THE FIRST ARE CHECKED AT THE TRANSMITTER OUTPUT,
* BY SCANNING THE TXDATA BIT.
* EACH USYRT RCV FLAG IS TIMED AS IT IS RECEIVED, AND THE 4 DATA WORDS
* ARE READ AND COMPARED TO EXPECTED VALUES.
* FINALLY, THE LINE UNIT IS CLOCKED FOR SEVERAL CHAR TIMES, AND A CHECK
* IS MADE FOR OACT = 0 (TEOM SHOULD CAUSE TX ENABLE TO DROP).
* THE ABOVE TEST IS REPEATED FOR EACH OF THE FOLLOWING SYNCH CHAR DATA
* PATTERNS : 226,000,125,252,376,177.
*****

```


***** DATA PATTERN A *****

PATA:

```
.BYTE 125
.BYTE 252
.BYTE 000
.BYTE 377
.BYTE 001
.BYTE 002
.BYTE 004
.BYTE 010
.BYTE 020
.BYTE 040
.BYTE 100
.BYTE 200
.BYTE 376
.BYTE 375
.BYTE 373
.BYTE 367
.BYTE 357
.BYTE 337
.BYTE 277
.BYTE 177
```

***** DATA PATTERN B *****

PATB:

```
.BYTE 000
.BYTE 000
.BYTE 040
.BYTE 100
.BYTE 220
.BYTE 000
.BYTE 000
.BYTE 051
```

***** DATA PATTERN C *****

PATC:

```
.BYTE 020
.BYTE 020
.BYTE 020
```

***** DATA PATTERN D *****

PATD:

```
.BYTE 000
.BYTE 040
.BYTE 000
```

***** DATA PATTERN E *****

PATE:

```
.BYTE 000
.BYTE 120
.BYTE 020
.BYTE 100
.BYTE 120
.BYTE 000
```

***** DATA PATTERN F *****

PATF:

.BYTE 050
.BYTE 051
.BYTE 050

***** DATA PATTERN G *****

PATG:

.BYTE 000
.BYTE 000
.BYTE 240
.BYTE 120
.BYTE 177
.BYTE 000
.BYTE 000
.BYTE 001

***** DATA PATTERN H *****

PATH:

.BYTE 000
.BYTE 000
.BYTE 377
.BYTE 017
.BYTE 377
.BYTE 377
.BYTE 375
.BYTE 377

***** DATA PATTERN I *****

PATI:

.BYTE 000
.BYTE 000
.BYTE 000
.BYTE 000
.BYTE 000
.BYTE 103
.BYTE 000
.BYTE 000

***** DATA PATTERN J *****

PATJ:

.BYTE 000
.BYTE 000
.BYTE 010
.BYTE 002
.BYTE 004
.BYTE 103
.BYTE 001
.BYTE 100

***** DATA PATTERN K *****

PATK: .BYTE 000
.BYTE 000
.BYTE 377
.BYTE 377
.BYTE 125
.BYTE 125
.BYTE 252
.BYTE 252
.BYTE 001
.BYTE 000
.BYTE 002
.BYTE 000
.BYTE 004
.BYTE 000
.BYTE 010
.BYTE 000
.BYTE 020
.BYTE 000
.BYTE 040
.BYTE 000
.BYTE 100
.BYTE 000
.BYTE 200
.BYTE 000
.BYTE 000
.BYTE 001
.BYTE 000
.BYTE 002
.BYTE 000
.BYTE 004
.BYTE 000
.BYTE 010
.BYTE 000
.BYTE 020
.BYTE 000
.BYTE 040
.BYTE 000
.BYTE 100
.BYTE 000
.BYTE 200
.BYTE 376
.BYTE 377
.BYTE 375
.BYTE 377
.BYTE 373
.BYTE 377
.BYTE 367
.BYTE 377
.BYTE 357
.BYTE 377
.BYTE 337
.BYTE 377
.BYTE 277
.BYTE 377
.BYTE 177
.BYTE 377
.BYTE 377
.BYTE 376

.BYTE 377
.BYTE 375
.BYTE 377
.BYTE 373
.BYTE 377
.BYTE 367
.BYTE 377
.BYTE 357
.BYTE 377
.BYTE 337
.BYTE 377
.BYTE 277
.BYTE 377
.BYTE 177

***** DATA PATTERN L *****

PATL:

.BYTE 000
.BYTE 000
.BYTE 377
.BYTE 377
.BYTE 000
.BYTE 000

***** DATA PATTERN M *****

PATM:

.BYTE 000
.BYTE 020
.BYTE 000
.BYTE 000
.BYTE 200
.BYTE 000
.BYTE 000
.BYTE 000
.BYTE 051

***** DATA PATTERN N *****

PATN:

.BYTE 000
.BYTE 000
.BYTE 000
.BYTE 125
.BYTE 000
.BYTE 252
.BYTE 000
.BYTE 377
.BYTE 005
.BYTE 000
.BYTE 012
.BYTE 000
.BYTE 017
.BYTE 000

***** DATA PATTERN O *****

PATO:

.BYTE 000
.BYTE 041
.BYTE 004
.BYTE 010
.BYTE 020
.BYTE 040
.BYTE 100
.BYTE 101
.BYTE 200
.BYTE 201
.BYTE 300
.BYTE 111
.BYTE 301
.BYTE 375

***** DATA PATTERN P *****

PATP:

.BYTE 000
.BYTE 113
.BYTE 200
.BYTE 040
.BYTE 020
.BYTE 010
.BYTE 001
.BYTE 104
.BYTE 007
.BYTE 105
.BYTE 007
.BYTE 144
.BYTE 107
.BYTE 157

***** DATA PATTERN U *****

PATU:

.BYTE 000
.BYTE 000
.BYTE 001
.BYTE 000
.BYTE 013
.BYTE 000
.BYTE 011
.BYTE 000
.BYTE 021
.BYTE 000
.BYTE 101
.BYTE 000
.BYTE 301
.BYTE 000
.BYTE 000
.BYTE 001
.BYTE 000
.BYTE 002
.BYTE 000
.BYTE 004
.BYTE 000
.BYTE 040
.BYTE 000
.BYTE 100

.BYTE 000
.BYTE 200
.BYTE 000
.BYTE 346
.BYTE 000
.BYTE 345
.BYTE 000
.BYTE 343
.BYTE 000
.BYTE 307
.BYTE 000
.BYTE 247
.BYTE 000
.BYTE 147

***** DATA PATTERN V *****

PATV: .BYTE 000
.BYTE 000
.BYTE 333
.BYTE 000
.BYTE 331
.BYTE 000
.BYTE 323
.BYTE 000
.BYTE 313
.BYTE 000
.BYTE 233
.BYTE 000
.BYTE 133
.BYTE 000
.BYTE 000
.BYTE 001
.BYTE 000
.BYTE 002
.BYTE 000
.BYTE 004
.BYTE 000
.BYTE 040
.BYTE 000
.BYTE 100
.BYTE 000
.BYTE 200
.BYTE 000
.BYTE 346
.BYTE 000
.BYTE 345
.BYTE 000
.BYTE 343
.BYTE 000
.BYTE 307
.BYTE 000
.BYTE 247
.BYTE 000
.BYTE 147

9.0 ERROR INFORMATION

9.1 ERROR REPORTING

ERRORS ARE REPORTED BY THE PROGRAM AS THEY OCCUR (IF NOT INHIBITED). THE REPORT CONFORMS TO THE DIAGNOSTIC SUPERVISOR ERROR REPORT FORMAT, AND CONSISTS OF A DESCRIPTION OF THE ERROR, THE TEST NUMBER, SUBTEST NUMBER, PC OF THE ERROR CALL, DEVICE ADDRESS, AND BASIC AND EXTENDED ERROR INFORMATION.

THE FOLLOWING EXAMPLE PROVIDES A TYPICAL ERROR REPORT, WHICH DESCRIBES AN 'IRDY NOT SET' ERROR, AND PROVIDES THE PC OF THE ERROR CALL AND THE PC OF THE CALL TO THE SUBROUTINE REPORTING IT, THE FAILING REGISTER NAME, AND DEVICE REGISTER CONTENTS :

```
CZDMR DVC FTL ERR 00017 ON UNIT 00 TST 034 SUB 000 PC: 006210
IRDY NOT SET
PC OF SUBR CALL: 030044
DEVICE CSR ADDRESS : 160170
```

FAILING REG: INBUS/OUTBUS REG 12

```
LINE UNIT INBUS REGS:
REG10  REG11  REG12  REG13
000    120    000    257
REG14  REG15  REG16  REG17
024    377    377    035
```

```
LINE UNIT EXTENDED REGS:
AX0-15 AX0-16 AX1-15 AX1-16
000    000    000    000
AX2-15 AX2-16 AX3-15 AX3-16
000    000    000    000
```

FOR OTHER ERRORS, THE REPORT MAY BE MORE EXTENSIVE, AND REQUIRE ADDITIONAL DATA TO BE REPORTED.

IF EXTENDED ERROR INFORMATION HAD BEEN INHIBITED USING THE IXE FLAG PRIOR TO RUNNING THE TEST, THE ABOVE ERROR WOULD HAVE BEEN REPORTED IN THE FOLLOWING SHORTENED FORM :

```
CZDMR DVC FTL ERR 00017 ON UNIT 00 TST 034 SUB 000 PC:006210
IRDY NOT SET
PC OF SUBR CALL: 030044
DEVICE CSR ADDRESS : 160170
```

FAILING REG: INBUS/OUTBUS REG 12

```
2194          .TITLE CZDMRC M8203 STATIC DIAG #1
2203          002000          .=2000
2204
2205
2206
2207
2208
2209
2210          .MCALL  SVC
2211 002000          SVC          ; INITIALIZE SUPERVISOR MACROS
2212
2213
2214
2215
2216
2217 002000          BGNMOD  LU1MOD
2218
2219
2220          000001          $LSTIN= 1
2221          000001          $LSTTAG= 1
2222          000001          SVCINS= 1          ; LIST INSTRUCTIONS, SHIFTED RIGHT
2223          000001          SVCTST= 1          ; LIST TEST TAGS, SHIFTED RIGHT
2224          000001          SVCSUB= 1          ; LIST SUBTEST TAGS, SHIFTED RIGHT
2225          000001          SVCGBL= 1          ; LIST GLOBAL TAGS, SHIFTED RIGHT
2226          000001          SVCTAG= 1          ; LIST OTHER TAGS, SHIFTED RIGHT
2227
2228          ; CHANGE THE VALUES OF THE SVC... SYMBOLS TO BE ZERO IF YOU WISH
2229          ; TO ALIGN THE MACRO CALLS AND THEIR EXPANSIONS. CHANGE THE
2230          ; SYMBOLS TO BE MINUS-ONE TO NOT LIST THE EXPANSIONS. YOU MAY
2231          ; CHANGE THE SYMBOLS AT ANY POINT IN YOUR PROGRAM.
2232
2233
```

2235
2236
2237
2238
2239
2240
2241 002000
2242
2244
2245
2246
2247
2248
2250
2251 002000
(4) 002000
(4) 002000 103
(4) 002001 132
(4) 002002 104
(4) 002003 115
(4) 002004 122
(6) 002005 000
(6) 002006 000
(5) 002007 000
(5) 002010
(4) 002010 103
(5) 002011
(4) 002011 060
(5) 002012
(4) 002012 000000
(5) 002014
(4) 002014 000055
(5) 002016
(4) 002016 035556
(5) 002020
(4) 002020 000000
(5) 002022
(4) 002022 002252
(5) 002024
(4) 002024 000000
(5) 002026
(4) 002026 036234
(5) 002030
(4) 002030 000000
(5) 002032
(4) 002032 000000
(5) 002034
(4) 002034 000000
(5) 002036
(4) 002036 000000
(5) 002040
(4) 002040 002124
(5) 002042
(4) 002042 000000
(5) 002044
(4) 002044 000000

```
.SBTTL PROGRAM HEADER
:++
: THE PROGRAM HEADER IS THE INTERFACE BETWEEN
: THE DIAGNOSTIC PROGRAM AND THE SUPERVISOR.
:--

        POINTER BGNAU,BGNDU

:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
: IF ANY OPTIONAL POINTERS ARE TO BE USED IN THE 'HEADER', CHANGE
: 'POINTER' TO CONTAIN THE CORRECT ARGUMENTS. IF ALL OPTIONAL
: POINTERS ARE TO BE USED, CHANGE 'POINTER' TO BE 'POINTER ALL'.
:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX

        HEADER CZDMP,C,0,45.,0
```

```
LSNAME::
        .ASCII /C/
        .ASCII /Z/
        .ASCII /D/
        .ASCII /M/
        .ASCII /R/
        .BYTE 0
        .BYTE 0
        .BYTE 0

LSREV::
        .ASCII /C/

LSDEPO::
        .ASCII /O/

LSUNIT::
        .WORD 0

LSTIML::
        .WORD 45.

LSHPCP::
        .WORD LSHARD

LSSPCP::
        .WORD 0

LSHPTP::
        .WORD LSHW

LSSPTP::
        .WORD 0

LSLADP::
        .WORD L$LAST

L$STA::
        .WORD 0

L$CO::
        .WORD 0

L$DTYP::
        .WORD 0

L$APT::
        .WORD 0

L$DTP::
        .WORD 0

L$DISPATCH
        .WORD 0

L$ENVI::
        .WORD 0
```

(5) 002046
(4) 002046 000000
(5) 002050
(4) 002050 003
(3) 002051 003
(5) 002052
(4) 002052 000000
(5) 002054 000000
(5) 002056
(4) 002056 000000
(5) 002060
(4) 002060 003462
(5) 002062
(4) 002062 000000
(5) 002064
(4) 002064 000000
(5) 002066
(4) 002066 000000
(5) 002070
(4) 002070 021634
(5) 002072
(4) 002072 021552
(5) 002074
(4) 002074 000000
(5) 002076
(4) 002076 003470
(5) 002100
(4) 002100 104035
(5) 002102
(4) 002102 000000
(5) 002104
(4) 002104 021120
(5) 002106
(4) 002106 021550
(5) 002110
(4) 002110 021470
(5) 002112
(4) 002112 021112
(5) 002114
(4) 002114 000000
(5) 002116
(4) 002116 000000
(5) 002120
(4) 002120 000000

L\$EXP1::
L\$MREV:: .WORD 0
 .BYTE C\$REVISION
 .BYTE C\$EDIT
L\$EF::
 .WORD 0
 .WORD 0
L\$SPC::
 .WORD 0
L\$DEVP::
 .WORD L\$DVTYP
L\$REPP::
 .WORD 0
L\$EXP4::
 .WORD 0
L\$EXP5::
 .WORD 0
L\$AUT::
 .WORD L\$AU
L\$DUT::
 .WORD L\$DU
L\$LUN::
 .WORD 0
L\$DESP::
 .WORD L\$DESC
L\$LOAD::
 EMT E\$LOAD
L\$ETP::
 .WORD 0
L\$ICP::
 .WORD L\$INIT
L\$CCP::
 .WORD L\$CLEAN
L\$ACP::
 .WORD L\$AUTO
L\$PRT::
 .WORD L\$PROT
L\$TEST::
 .WORD 0
L\$DLY::
 .WORD 0
L\$HIME::
 .WORD 0

2252
2254
2255
2256
2258
2259
2260
2261
2262
2263
2264
2265

;
: CHANGE THE 'HEADER' TO CONTAIN THE PROPER ARGUMENTS.
;
.EVEN

2288
2289
2290
2291
2292
2293
2294
2295
2296
(3)
(3)
(3)
2297
2298
2299
2300
2301
2302
2303
2304
2305
2306
2307
2308
2309
2310
(3)
2311
2312
2313
2314
2315

002250
002250 000013
002252
002252
002252 000007
002254 160170
002256 000300
002260 005000
002262 000003
002264 000056
002266 000000
002270 000000
002272 000000
002274 000004
002276 000001
002300
002300

.SBTTL DEFAULT HARDWARE P-TABLE

:/ THE DEFAULT HARDWARE P-TABLE CONTAINS DEFAULT VALUES OF
:/ THE TEST-DEVICE PARAMETERS. THE STRUCTURE OF THIS TABLE
:/ IS IDENTICAL TO THE STRUCTURE OF THE RUN-TIME P-TABLE.

BGNHW DFPTBL

.WORD 7
.WORD 160170
.WORD 300
.WORD 5000
.WORD 3
.WORD 056
.WORD 000
.WORD 000
.WORD 0
.WORD 4
.WORD 1

.WORD L10000-LSHW/2
LSHW::
DFPTBL::
:MICROPROCESSOR TYPE = M8207
:DMC11 OR KMC11 CSR UNIBUS ADDRESS
:DMC11 OR KMC11 INTERRUPT VECTOR
:DMC11 OR KMC11 INTERRUPT PRIORITY LEVEL = 5
:LINE UNIT = M8203
:SWITCH PACK #1 (REG 11)
:SWITCH PACK #2 (REG 15)
:SWITCH PACK #3 (REG 16)
:H3254&H3255 USED
:BAUD RATE = 56 K
:RUN SWITCH ON MICROPROCESSOR IS ON

ENDHW

L10000:

2317
2318
2319
2320
2321
2322
2323
2324
(3)
(3)
(3)
2325
2326
2327
(3)
2328
2329
2330
2331
2332
2333

.SBTTL SOFTWARE P-TABLE
:////////////////////
:/ THE SOFTWARE P-TABLE CONTAINS THE VALUES OF THE PROGRAM
:/ PARAMETERS THAT CAN BE CHANGED BY THE OPERATOR.
:////////////////////

002300
002300 000000
002302
002302

BGNSW SFPTBL

.WORD L10001-LSSW/2
LSSW::
SFPTBL::

002302
002302

ENDSW

L10001:

2335
2336
2337
2338
2339
2340
2341
2342
2343
2344
2345
2346
2347
2348
2349
2350
2351
2352
2353
2354

002302

.SBTTL GLOBAL EQUATES SECTION

:/
:/ THE GLOBAL EQUATES SECTION CONTAINS PROGRAM EQUATES THAT
:/ ARE USED IN MORE THAN ONE TEST.
:/

EQUALS

:
: BIT DEFINITIONS

(1) 100000
(1) 040000
(1) 020000
(1) 010000
(1) 004000
(1) 002000
(1) 001000
(1) 000400
(1) 000200
(1) 000100
(1) 000040
(1) 000020
(1) 000010
(1) 000004
(1) 000002
(1) 000001
(1) 001000
(1) 000400
(1) 000200
(1) 000100
(1) 000040
(1) 000020
(1) 000010
(1) 000004
(1) 000002
(1) 000001

BIT15== 100000
BIT14== 40000
BIT13== 20000
BIT12== 10000
BIT11== 4000
BIT10== 2000
BIT09== 1000
BIT08== 400
BIT07== 200
BIT06== 100
BIT05== 40
BIT04== 20
BIT03== 10
BIT02== 4
BIT01== 2
BIT00== 1
:
BIT9== BIT09
BIT8== BIT08
BIT7== BIT07
BIT6== BIT06
BIT5== BIT05
BIT4== BIT04
BIT3== BIT03
BIT2== BIT02
BIT1== BIT01
BIT0== BIT00

:
: EVENT FLAG DEFINITIONS
: EF32:EF17 RESERVED FOR SUPERVISOR TO PROGRAM COMMUNICATION

(1) 000040
(1) 000037

EF.START== 32. ; START COMMAND WAS ISSUED
EF.RESTART== 31. ; RESTART COMMAND WAS ISSUED

; CONTINUE COMMAND WAS ISSUED
; A NEW PASS HAS BEEN STARTED
; A POWER-FAIL/POWER-UP OCCURRED

```
(1) 000036 EF.CONTINUE== 30.  
(1) 000035 EF.NEW== 29.  
(1) 000034 EF.PWR== 28.  
(1) :  
(1) :  
(1) : PRIORITY LEVEL DEFINITIONS  
(1) :  
(1) 000340 PRI07== 340  
(1) 000300 PRI06== 300  
(1) 000240 PRI05== 240  
(1) 000200 PRI04== 200  
(1) 000140 PRI03== 140  
(1) 000100 PRI02== 100  
(1) 000040 PRI01== 40  
(1) 000000 PRI00== 0  
(1) :  
(1) : OPERATOR FLAG BITS  
(1) :  
(1) 000004 EVL== 4  
(1) 000010 LOT== 10  
(1) 000020 ADR== 20  
(1) 000040 IDU== 40  
(1) 000100 ISR== 100  
(1) 000200 UAM== 200  
(1) 000400 BOE== 400  
(1) 001000 PNT== 1000  
(1) 002000 PRI== 2000  
(1) 004000 IXE== 4000  
(1) 010000 IBE== 10000  
(1) 020000 IER== 20000  
(1) 040000 LOE== 40000  
(1) 100000 HOE== 100000
```

2355
2356
2357
2358
2359
2360
2361
2362
2363
2364
2365
2366
2367
2368
2369
2370
2371
2372
2373
2374
2375
2376
2377
2378

```
::*****  
;* PROGRAM EVENT FLAG DEFINITIONS  
:*****
```

```
::*****  
;* MAINTENANCE REGISTER - BSEL1  
:*****
```

```
RUN = BIT7  
MCLR = BIT6  
STEPLU = BIT4  
LULOOP = BIT3  
ROMO = BIT2  
ROMI = BIT1  
STEPMP = BIT0
```

```
::*****
```

```
2379 ;* OBUS REG 10 - TRANSMITTER BUFFER
2380 :*****
2381 000200 TX7 = BIT7
2382 000100 TX6 = BIT6
2383 000040 TX5 = BIT5
2384 000020 TX4 = BIT4
2385 000010 TX3 = BIT3
2386 000004 TX2 = BIT2
2387 000002 TX1 = BIT1
2388 000001 TX0 = BIT0
2389
2390 :*****
2391 ;* OBUS REG 11
2392 :*****
2393 000200 OC = BIT7
2394 000010 GOAH = BIT3
2395 000004 ABORT = BIT2
2396 000002 EOM = BIT1
2397 000001 SOM = BIT0
2398
2399 :*****
2400 ;* OBUS REG 12
2401 :*****
2402 000200 IC = BIT7
2403 000100 BPOLL = BIT6
2404 000040 LULP = BIT5
2405
2406 :*****
2407 ;* OBUS REG 13
2408 :*****
2409 000200 POLL = BIT7
2410 000100 DTR = BIT6
2411 000040 SELFR = BIT5
2412 000020 HDX = BIT4
2413 000010 MAINT1 = BIT3
2414 000004 MAINT2 = BIT2
2415 000002 SELSBY = BIT1
2416
2417 :*****
2418 ;* OBUS REG 14
2419 :*****
2420 000100 TXEN = BIT6
2421 000040 DISSI = BIT5
2422 000020 RDAX = BIT4
2423 000010 WAX = BIT3
2424 000004 ENAX = BIT2
2425 000002 AX2 = BIT1
2426 000001 AX1 = BIT0
2427
2428 :*****
2429 ;* OBUS REG 17
2430 :*****
2431 000200 CRC2 = BIT7
2432 000100 CRC1 = BIT6
2433 000040 IDLE = BIT5
2434 000020 SECA = BIT4
```

2435 000010 STRIP = BIT3
2436 000004 RDALL = BIT2
2437 000002 IERR = BIT1
2438 000001 DDCMP = BIT0

2439
2440 :*****
2441 :* IBUS REG 10 - RECEIVER BUFFER
2442 :*****

2443 000200 RX7 = BIT7
2444 000100 RX6 = BIT6
2445 000040 RX5 = BIT5
2446 000020 RX4 = BIT4
2447 000010 RX3 = BIT3
2448 000004 RX2 = BIT2
2449 000002 RX1 = BIT1
2450 000001 RX0 = BIT0

2451
2452 :*****
2453 :* IBUS REG 11
2454 :*****

2455 000200 OC = BIT7
2456 000100 OACT = BIT6
2457 000040 SW3 = BIT5
2458 000020 ORDY = BIT4
2459 000010 SW2 = BIT3
2460 000004 SW1 = BIT2
2461 000002 SW0 = BIT1
2462 000001 UNRR = BIT0

2463
2464 :*****
2465 :* IBUS REG 12
2466 :*****

2467 000200 IC = BIT7
2468 000100 IACT = BIT6
2469 000040 LULP = BIT5
2470 000020 IRDY = BIT4
2471 000010 OVRR = BIT3
2472 000004 RAB = BIT2
2473 000002 EBLK = BIT1
2474 000001 BCC = BIT0

2475
2476 :*****
2477 :* IBUS REG 13
2478 :*****

2479 000200 RING = BIT7
2480 000100 DTR = BIT6
2481 000040 RTS = BIT5
2482 000020 HDX = BIT4
2483 000010 MODR = BIT3
2484 000004 CS = BIT2
2485 000002 STBY = BIT1
2486 000001 CARR = BIT0

2487
2488 :*****
2489 :* IBUS REG 14
2490 :*****

2491	000200	READY	=	BIT7
2492	000100	TXEN	=	BIT6
2493	000040	DISSI	=	BIT5
2494	000020	RDAX	=	BIT4
2495	000010	WAX	=	BIT3
2496	000004	ENAX	=	BIT2
2497	000002	AX2	=	BIT1
2498	000001	AX1	=	BIT0

2500 :*****

2501 :* IBUS REG 17

2502 :*****

2503	000200	SIGR	=	BIT7
2504	000100	SIGQ	=	BIT6
2505	000040	TXDATA	=	BIT5
2506	000020	OCOR	=	BIT4
2507	000010	ICIR	=	BIT3
2508	000004	TESTMD	=	BIT2
2509	000002	MCLK	=	BIT1
2510	000001	DDCMP	=	BIT0

2512 :*****

2513 :* AX0-15 - USYRT REG 0 (READ ONLY)

2514 :*****

2515	000200	RX7	=	BIT7
2516	000100	RX6	=	BIT6
2517	000040	RX5	=	BIT5
2518	000020	RX4	=	BIT4
2519	000010	RX3	=	BIT3
2520	000004	RX2	=	BIT2
2521	000002	RX1	=	BIT1
2522	000001	RX0	=	BIT0

2524 :*****

2525 :* AX0-16 - USYRT REG 1 (READ ONLY)

2526 :*****

2527	000200	RERR	=	BIT7
2528	000100	ASBC2	=	BIT6
2529	000040	ASBC1	=	BIT5
2530	000020	ASBC0	=	BIT4
2531	000010	ROR	=	BIT3
2532	000004	RABT	=	BIT2
2533	000002	REOM	=	BIT1
2534	000001	RSOM	=	BIT0

2536 :*****

2537 :* AX1-15 - USYRT REG 2

2538 :*****

2539	000200	TX7	=	BIT7
2540	000100	TX6	=	BIT6
2541	000040	TX5	=	BIT5
2542	000020	TX4	=	BIT4
2543	000010	TX3	=	BIT3
2544	000004	TX2	=	BIT2
2545	000002	TX1	=	BIT1
2546	000001	TX0	=	BIT0


```
2547
2548
2549      ::*****
2550      :* AX1-16 - USYRT REG 3
2551      ::*****
2552      TERR      = BIT7
2553      TXGA      = BIT3
2554      TXAB      = BIT2
2555      TEOM      = BIT1
2556      TSOM      = BIT0
2557
2558      ::*****
2559      :* AX2-15 - USYRT REG 4
2560      ::*****
2561      SYN7      = BIT7
2562      SYN6      = BIT6
2563      SYN5      = BIT5
2564      SYN4      = BIT4
2565      SYN3      = BIT3
2566      SYN2      = BIT2
2567      SYN1      = BIT1
2568      SYN0      = BIT0
2569      SYNCH     = 226
2570
2571      ::*****
2572      :* AX2-16 - USYRT REG 5
2573      ::*****
2574      APA       = BIT7
2575      DDC       = BIT6
2576      STR       = BIT5
2577      SEC       = BIT4
2578      IDL       = BIT3
2579      CRCTY2    = BIT2
2580      CRCTY1    = BIT1
2581      CRCTY0    = BIT0
2582
2583      ::*****
2584      :* AX3-15 - USYRT REG 6
2585      ::*****
2586      I422      = BIT7
2587      XYZ       = BIT6
2588      C32BCC    = BIT5
2589      V35       = BIT4
2590      INTGRL    = BIT3
2591      C32ENB    = BIT2
2592      OP        = BIT1
2593      TEST      = BIT0
2594      AX315U    = I422!XYZ!C32BCC!V35!INTGRL!OP
2595
2596      ::*****
2597      :* AX3-16 - USYRT REG 7
2598      ::*****
2599      TXLEN2    = BIT7
2600      TXLEN1    = BIT6
2601      TXLEN0    = BIT5
2602      RXLEN2    = BIT2
2603      RXLEN1    = BIT1
```

2603 000001 RXLENO = BIT0

2604
 2605
 2606
 2607
 2608
 2609

 ;* TX CONTROL BITS DEFINED ON WORD BASIS

2611
 2612 004000
 2613 002000
 2614 001000
 2615 000400

TXGOA = BIT11
 TXABT = BIT10
 TXEOM = BIT9
 TXSOM = BIT8

2616
 2617
 2618
 2619
 2620

 ;* RCV CONTROL BITS DEFINED ON WORD BASIS

2621
 2622
 2623
 2624 004000
 2625 002000
 2626 001000
 2627 000400

RXOVR = BIT11
 RXABT = BIT10
 RXEBL = BIT9
 RXBCC = BIT8

2628
 2629
 2630
 2631
 2632

 ;* ADDRESS EQUATES FOR REGISTER STORAGE TABLE (LUREG:)

2633
 2634
 2635 002302
 2636 002304
 2637 002306
 2638 002310
 2639 002312
 2640 002314
 2641 002316
 2642 002320
 2643 002322
 2644 002324
 2645 002326
 2646 002330
 2647 002332
 2648 002334
 2649 002336
 2650 002340

LUR10 = LUREG+0 ;LINE UNIT IBUS REG 10
 LUR11 = LUREG+2 ;LINE UNIT IBUS REG 11
 LUR12 = LUREG+4 ;LINE UNIT IBUS REG 12
 LUR13 = LUREG+6 ;LINE UNIT IBUS REG 13
 LUR14 = LUREG+10 ;LINE UNIT IBUS REG 14
 LUR15 = LUREG+12 ;LINE UNIT IBUS REG 15
 LUR16 = LUREG+14 ;LINE UNIT IBUS REG 16
 LUR17 = LUREG+16 ;LINE UNIT IBUS REG 17
 AX0.15 = LUREG+20 ;USYRT REG 0
 AX0.16 = LUREG+22 ;USYRT REG 1
 AX1.15 = LUREG+24 ;USYRT REG 2
 AX1.16 = LUREG+26 ;USYRT REG 3
 AX2.15 = LUREG+30 ;USYRT REG 4
 AX2.16 = LUREG+32 ;USYRT REG 5
 AX3.15 = LUREG+34 ;USYRT REG 6
 AX3.16 = LUREG+36 ;USYRT REG 7

2651
 2652
 2653
 2654

2655
 2656 100000
 2657
 2658 100000

CHPCHK = BIT15
 BCCCHK = BIT15

2656 100000 CRCCHK = BIT15

2660
2661
2662
2663
2664
2665
2666
2667
2668
2669
2670
2671
2672
2673
2674
2675
2676
2677
2678
2679
2680
2681

021000
122000
121000

```
*****  
;* MICROINSTRUCTION DEFINITIONS  
*****  
MVIOX = 021000 ;MOVE IBUS TO OBUS*  
MVIXO = 122000 ;MOVE IBUS* TO OBUS  
MVIXOX = 121000 ;MOVE IBUS* TO OBUS*
```

```
***** ERROR1 BIT FLAG DEFINITIONS *****  
RRDYTO = BIT0  
WRDYTO = BIT1
```

```
2683 .SBTTL GLOBAL DATA SECTION
2684
2685 :////////////////////
2686 :// THE GLOBAL DATA SECTION CONTAINS DATA THAT ARE USED
2687 :// IN MORE THAN ONE TEST.
2688 :////////////////////
2689
2690 :*****
2691 :* STORAGE FOR DEVICE REGISTERS
2692 :*****
2693 002302 000020 LUREG: .BLKW 16.
2694
2695 :*****
2696 :* MISCELLANEOUS STORAGE
2697 :*****
2698 002342 000000 SCRACH: .WORD 0 ;GEN'L PURPOSE SCRATCH WORD
2699 002344 000000 LOGDEV: .WORD 0 ;LOGICAL DEVICE NUMBER
2700 002346 000000 PSTACK: .WORD 0 ;CONTAINS BASE LEVEL PROGRAM STACK POINTER
2701 002350 000000 PRIOR: .WORD 0 ;CPU PRIORITY FOR PRINTOUT
2702 002352 000000 SUBRPC: .WORD 0 ;PC OF SUBR CALL FOR ERROR REPORTS
2703 002354 000000 INTFLG: .WORD 0 ;INTERRUPT RECEIVED FLAGS
2704 ; BIT 0 FOR TX, BIT 1 FOR RCV
2705 002356 000000 ERRFLG: .WORD 0 ;SUBROUTINE ERROR FLAG
2706 002360 000000 TIMFLG: .WORD 0 ;EVENT TIME-OUT FLAG
2707 002362 000000 RETADR: .WORD 0 ;SUBR ERROR RETURN ADDRESS
2708 002364 000000 REDBYT: .WORD 0 ;LO BYTE CONTAINS BYTE READ FROM LU REG
2709 002366 000000 WRIBYT: .WORD 0 ;LO BYTE CONTAINS BYTE TO LOAD INTO LU REG
2710 002370 000000 RAX15: .WORD 0 ;LO BYTE CONTAINS BYTE READ FROM REG 15
2711 002372 000000 RAX16: .WORD 0 ;LO BYTE CONTAINS BYTE READ FROM REG 16
2712 002374 000000 WAX15: .WORD 0 ;LO BYTE CONTAINS BYTE TO LOAD INTO REG 15
2713 002376 000000 WAX16: .WORD 0 ;LO BYTE CONTAINS BYTE TO LOAD INTO REG 16
2714 002400 000000 REGNUM: .WORD 0 ;NUMBER (10-17) OF LINE UNIT REG BEING TESTED
2715 002402 000000 AXNUM: .WORD 0 ;NUMBER (0-7) OF EXTENDED REG BYTE BEING TESTED
2716 002404 000000 GOODAT: .WORD 0 ;STORAGE FOR EXPECTED DATA
2717 002406 000000 BADDAT: .WORD 0 ;STORAGE FOR ACTUAL DATA
2718 002410 000000 LOADAT: .WORD 0 ;CONTAINS TEST DATA LOADED INTO REG
2719 002412 000000 FRSTIM: .WORD 0 ;FLAG=0 IF PROGRAM JUST LOADED
2720 002414 000000 SAVE4: .WORD 0 ;SAVE LOC 4 HERE (ERROR TRAP VECTOR)
2721 002416 000000 SAVE6: .WORD 0 ;SAVE LOC 6 HERE (ERROR TRAP VECTOR)
2722 002420 000000 ERROR1: .WORD 0 ;SUBR ERROR BIT FLAGS (DEF'D IN GLOBAL EQUATES)
2723 002422 000000 TXWORD: .WORD 0 ;BITS 0-11 CONTAIN DATA TO LOAD INTO TX SILO
2724 002424 000000 RXWORD: .WORD 0 ;BITS 0-11 CONTAIN DATA READ FROM RCV SILO
2725 002426 000000 DISILO: .WORD 0 ;CONTAINS CURRENT STATE OF DISSI IN BIT 5
2726 002430 000000 CHPTYP: .WORD 0 ;USYRT CHIP TYPE, =0 FOR SIG, ELSE =1
2727 002432 000000 SAVLEN: .WORD 0 ;SAVED TX AND RCV CHAR LENGTHS
2728 002434 000000 DEVMAP: .WORD 0 ;BIT MAP OF ACTIVE DEVICES
2729 002436 000000 DEVPTR: .WORD 0 ;DEVICE MAP BIT POINTER
2730 002440 000000 UNIT: .WORD 0 ;CONTAINS UNIT NO. (1 TO N)
2731 002442 000000 TSTNUM: .WORD 0 ;CONTAINS TEST NUMBER FOR SOME TESTS
2732 002444 000000 STARES: .WORD 0 ;FLAG=0 IF FIRST PASS AFTER STA OR RES
2733
2734 :***** CURRENT DEVICE PARAMETERS *****
2735 002446 160170 MPCSR: .WORD 160170 ;POINTER TO MICROPROCESSOR CSR'S
2736 002450 160171 BSEL1: .WORD 160171 ;POINTER TO BSEL1
2737 002452 160172 BSEL2: .WORD 160172 ;POINTER TO BSEL2
2738 002454 BSEL4:
```

2739	002454	160174	SEL4: .WORD	160174	: POINTER TO SEL4
2740	002456	160176	SEL6: .WORD	160176	: POINTER TO SEL6
2741	002460	000300	MPIVEC: .WORD	300	: MICROPROCESSOR INPUT INTERRUPT VECTOR
2742	002462	000304	MPOVEC: .WORD	304	: MICROPROCESSOR OUTPUT INTERRUPT VECTOR
2743	002464	000240	MPRIOR: .WORD	240	: MICROPROCESSOR DEVICE PRIORITY
2744	002466	000000	LUSWI1: .WORD	0	: LINE UNIT SWITCH PACK #1
2745	002470	000000	LUSWI2: .WORD	0	: LINE UNIT SWITCH PACK #2
2746	002472	000000	LUSWI3: .WORD	0	: LINE UNIT SWITCH PACK #3
2747	002474	000000	TSTCON: .WORD	0	: TEST CONNECTOR INDICATOR
2748	002476	000000	RUNINH: .WORD	0	: RUN SWITCH INDICATOR
2749					
2750			;***** STORAGE FOR DATA READ IN ADDRESS TESTS *****		
2751	002500	000	REDDAT: .BYTE	0	
2752	002501	000		.BYTE	0
2753	002502	000		.BYTE	0
2754	002503	000		.BYTE	0
2755	002504	000		.BYTF	0
2756	002505	000		.BYTE	0
2757	002506	000		.BYTE	0
2758	002507	000		.BYTE	0
2759					
2760			;***** GEN'L PURPOSE SCRATCH STORAGE *****		
2761	002510	000000	REG0: .WORD	0	
2762	002512	000000	REG1: .WORD	0	
2763	002514	000000	REG2: .WORD	0	
2764	002516	000000	REG3: .WORD	0	
2765	002520	000000	REG4: .WORD	0	
2766	002522	000000	REG5: .WORD	0	
2767	002524	000000	REG6: .WORD	0	
2768	002526	000000	REG7: .WORD	0	
2769					
2770			;***** SCRATCH STORAGE FOR MESSAGE REPORTING *****		
2771	002530	000000	TMP0: .WORD	0	
2772	002532	000000	TMP1: .WORD	0	
2773	002534	000000	TMP2: .WORD	0	
2774	002536	000000	TMP3: .WORD	0	
2775	002540	000000	TMP4: .WORD	0	
2776	002542	000000	TMP5: .WORD	0	
2777	002544	000000	TMP6: .WORD	0	
2778	002546	000000	TMP7: .WORD	0	
2779					
2780			;***** INBUS LU REG BIT MASKS FOR UNPREDICTABLE BITS *****		
2781	002550		UPBITS: .BYTE	000	: MASK FOR REG 10
2782	002550	000		.BYTE	056
2783	002551	056		.BYTE	000
2784	002552	000		.BYTE	257
2785	002553	257		.BYTE	100
2786	002554	100		.BYTE	377
2787	002555	377		.BYTE	377
2788	002556	377		.BYTE	306
2789	002557	306		.BYTE	
2790					
2791	002560	200	R14NRW: .BYTE	200	: REG 14 NON-R/W BITS
2792					
2793			;***** MASKS FOR EXTENDED REGISTER NON-READ/WRITE BITS *****		
2794	002561		ANBITS:		

2795	002561	377	.BYTE	377	:MASK FOR AX0-15
2796	002562	377	.BYTE	377	:MASK FOR AX0-16
2797	002563	000	.BYTE	000	:MASK FOR AX1-15
2798	002564	360	.BYTE	360	:MASK FOR AX1-16
2799	002565	000	.BYTE	000	:MASK FOR AX2-15
2800	002566	000	.BYTE	000	:MASK FOR AX2-16
2801	002567	004	.BYTE	004	:MASK FOR AX3-15
2802	002570	030	.BYTE	030	:MASK FOR AX3-16

***** DATA PATTERN A *****

2803					
2804					
2805	002571				
2806	002571	125	.BYTE	125	
2807	002572	252	.BYTE	252	
2808	002573	000	.BYTE	000	
2809	002574	377	.BYTE	377	
2810	002575	001	.BYTE	001	
2811	002576	002	.BYTE	002	
2812	002577	004	.BYTE	004	
2813	002600	010	.BYTE	010	
2814	002601	020	.BYTE	020	
2815	002602	040	.BYTE	040	
2816	002603	100	.BYTE	100	
2817	002604	200	.BYTE	200	
2818	002605	376	.BYTE	376	
2819	002606	375	.BYTE	375	
2820	002607	373	.BYTE	373	
2821	002610	367	.BYTE	367	
2822	002611	357	.BYTE	357	
2823	002612	337	.BYTE	337	
2824	002613	277	.BYTE	277	
2825	002614	177	.BYTE	177	

***** DATA PATTERN B *****

2826					
2827					
2828	002615				
2829	002615	000	.BYTE	000	
2830	002616	000	.BYTE	000	
2831	002617	040	.BYTE	040	
2832	002620	100	.BYTE	100	
2833	002621	220	.BYTE	220	
2834	002622	000	.BYTE	000	
2835	002623	000	.BYTE	000	
2836	002624	051	.BYTE	051	

***** DATA PATTERN C *****

2837					
2838					
2839	002625				
2840	002625	020	.BYTE	020	
2841	002626	020	.BYTE	020	
2842	002627	020	.BYTE	020	

***** DATA PATTERN D *****

2843					
2844					
2845	002630				
2846	002630	000	.BYTE	000	
2847	002631	040	.BYTE	040	
2848	002632	000	.BYTE	000	

***** DATA PATTERN E *****

2849

2851	002633		PATE:		
2852	002633	000		.BYTE	00C
2853	002634	120		.BYTE	120
2854	002635	020		.BYTE	020
2855	002636	100		.BYTE	100
2856	002637	120		.BYTE	120
2857	002640	000		.BYTE	000

:***** DATA PATTERN F *****

2860	002641		PATF:		
2861	002641	050		.BYTE	050
2862	002642	051		.BYTE	051
2863	002643	050		.BYTE	050

:***** DATA PATTERN G *****

2866	002644		PATG:		
2867	002644	000		.BYTE	000
2868	002645	000		.BYTE	000
2869	002646	240		.BYTE	240
2870	002647	120		.BYTE	120
2871	002650	177		.BYTE	177
2872	002651	000		.BYTE	000
2873	002652	000		.BYTE	000
2874	002653	001		.BYTE	001

:***** DATA PATTERN H *****

2877	002654		PATH:		
2878	002654	000		.BYTE	000
2879	002655	000		.BYTE	000
2880	002656	377		.BYTE	377
2881	002657	017		.BYTE	017
2882	002660	377		.BYTE	377
2883	002661	377		.BYTE	377
2884	002662	375		.BYTE	375
2885	002663	377		.BYTE	377

:***** DATA PATTERN I *****

2888	002664		PATI:		
2889	002664	000		.BYTE	000
2890	002665	000		.BYTE	000
2891	002666	000		.BYTE	000
2892	002667	000		.BYTE	000
2893	002670	000		.BYTE	000
2894	002671	103		.BYTE	103
2895	002672	000		.BYTE	000
2896	002673	000		.BYTE	000

:***** DATA PATTERN J *****

2898			PATJ:		
2899	002674			.BYTE	000
2900	002674	000		.BYTE	000
2901	002675	000		.BYTE	000
2902	002676	010		.BYTE	010
2903	002677	002		.BYTE	002
2904	002700	004		.BYTE	004
2905	002701	103		.BYTE	103
2906	002702	001		.BYTE	001

2907	002703	100	.BYTE	100
2908				
2909			:***** DATA PATTERN K *****	
2910	002704	000	PATK: .BYTE	000
2911	002705	000	.BYTE	000
2912	002706	377	.BYTE	377
2913	002707	377	.BYTE	377
2914	002710	125	.BYTE	125
2915	002711	125	.BYTE	125
2916	002712	252	.BYTE	252
2917	002713	252	.BYTE	252
2918	002714	001	.BYTE	001
2919	002715	000	.BYTE	000
2920	002716	002	.BYTE	002
2921	002717	000	.BYTE	000
2922	002720	004	.BYTE	004
2923	002721	000	.BYIF	000
2924	002722	010	.BYTE	010
2925	002723	000	.BYTE	000
2926	002724	020	.BYTE	020
2927	002725	000	.BYTE	000
2928	002726	040	.BYTE	040
2929	002727	000	.BYTE	000
2930	002730	100	.BYTE	100
2931	002731	000	.BYTE	000
2932	002732	200	.BYTE	200
2933	002733	000	.BYTE	000
2934	002734	000	.BYTE	000
2935	002735	001	.BYTE	001
2936	002736	000	.BYTE	000
2937	002737	002	.BYTE	002
2938	002740	000	.BYTE	000
2939	002741	004	.BYTE	004
2940	002742	000	.BYTE	000
2941	002743	010	.BYTE	010
2942	002744	000	.BYTE	000
2943	002745	020	.BYTE	020
2944	002746	000	.BYTE	000
2945	002747	040	.BYTE	040
2946	002750	000	.BYTE	000
2947	002751	100	.BYTE	100
2948	002752	000	.BYTE	000
2949	002753	200	.BYTE	200
2950	002754	376	.BYTE	376
2951	002755	377	.BYTE	377
2952	002756	375	.BYTE	375
2953	002757	377	.BYTE	377
2954	002760	373	.BYTE	373
2955	002761	377	.BYTE	377
2956	002762	367	.BYTE	367
2957	002763	377	.BYTE	377
2958	002764	357	.BYTE	357
2959	002765	377	.BYTE	377
2960	002766	337	.BYTE	337
2961	002767	377	.BYTE	377
2962	002770	277	.BYTE	277

2963	002771	377	.BYTE	377
2964	002772	177	.BYTE	177
2965	002773	377	.BYTE	377
2966	002774	377	.BYTE	377
2967	002775	376	.BYTE	376
2968	002776	377	.BYTE	377
2969	002777	375	.BYTE	375
2970	003000	377	.BYTE	377
2971	003001	373	.BYTE	373
2972	003002	377	.BYTE	377
2973	003003	367	.BYTE	367
2974	003004	377	.BYTE	377
2975	003005	357	.BYTE	357
2976	003006	377	.BYTE	377
2977	003007	337	.BYTE	337
2978	003010	377	.BYTE	377
2979	003011	277	.BYTE	277
2980	003012	377	.BYTE	377
2981	003013	177	.BYTE	177

***** DATA PATTERN L *****

2982				
2983				
2984	003014			
2985	003014	000	.BYTE	000
2986	003015	000	.BYTE	000
2987	003016	377	.BYTE	377
2988	003017	377	.BYTE	377
2989	003020	000	.BYTE	000
2990	003021	000	.BYTE	000

***** DATA PATTERN M *****

2991				
2992				
2993	003022			
2994	003022	000	.BYTE	000
2995	003023	020	.BYTE	020
2996	003024	000	.BYTE	000
2997	003025	000	.BYTE	000
2998	003026	200	.BYTE	200
2999	003027	000	.BYTE	000
3000	003030	000	.BYTE	000
3001	003031	051	.BYTE	051

***** DATA PATTERN N *****

3002				
3003				
3004	003032			
3005	003032	000	.BYTE	000
3006	003033	000	.BYTE	000
3007	003034	000	.BYTE	000
3008	003035	125	.BYTE	125
3009	003036	000	.BYTE	000
3010	003037	252	.BYTE	252
3011	003040	000	.BYTE	000
3012	003041	377	.BYTE	377
3013	003042	005	.BYTE	005
3014	003043	000	.BYTE	000
3015	003044	012	.BYTE	012
3016	003045	000	.BYTE	000
3017	003046	017	.BYTE	017
3018	003047	000	.BYTE	000

3019
3020
3021 003050
3022 003050 000
3023 003051 041
3024 003052 004
3025 003053 010
3026 003054 020
3027 003055 040
3028 003056 100
3029 003057 101
3030 003060 200
3031 003061 201
3032 003062 300
3033 003063 111
3034 003064 301
3035 003065 375
3036
3037
3038 003066
3039 003066 000
3040 003067 113
3041 003070 200
3042 003071 040
3043 003072 020
3044 003073 010
3045 003074 001
3046 003075 104
3047 003076 007
3048 003077 105
3049 003100 007
3050 003101 144
3051 003102 107
3052 003103 157
3053
3054
3055 003104 000
3056 003105 000
3057 003106 001
3058 003107 000
3059 003110 013
3060 003111 000
3061 003112 011
3062 003113 000
3063 003114 021
3064 003115 000
3065 003116 101
3066 003117 000
3067 003120 301
3068 003121 000
3069 003122 000
3070 003123 001
3071 003124 000
3072 003125 002
3073 003126 000
3074 003127 004

:***** DATA PATTERN O *****

PATO:
.BYTE 000
.BYTE 041
.BYTE 004
.BYTE 010
.BYTE 020
.BYTE 040
.BYTE 100
.BYTE 101
.BYTE 200
.BYTE 201
.BYTE 300
.BYTE 111
.BYTE 301
.BYTF 375

:***** DATA PATTERN P *****

PATP:
.BYTE 000
.BYTE 113
.BYTE 200
.BYTE 040
.BYTE 020
.BYTE 010
.BYTE 001
.BYTE 104
.BYTE 007
.BYTE 105
.BYTE 007
.BYTE 144
.BYTE 107
.BYTE 157

:***** DATA PATTERN U *****

PATU:
.BYTE 000
.BYTE 000
.BYTE 001
.BYTE 000
.BYTE 013
.BYTE 000
.BYTE 011
.BYTE 000
.BYTE 021
.BYTE 000
.BYTE 101
.BYTE 000
.BYTE 301
.BYTE 000
.BYTE 000
.BYTE 001
.BYTE 000
.BYTE 002
.BYTE 000
.BYTE 004

3075	003130	000	.BYTE	000
3076	003131	040	.BYTE	040
3077	003132	000	.BYTE	000
3078	003133	100	.BYTE	100
3079	003134	000	.BYTE	000
3080	003135	200	.BYTE	200
3081	003136	000	.BYTE	000
3082	003137	346	.BYTE	346
3083	003140	000	.BYTE	000
3084	003141	345	.BYTE	345
3085	003142	000	.BYTE	000
3086	003143	343	.BYTE	343
3087	003144	000	.BYTE	000
3088	003145	307	.BYTE	307
3089	003146	000	.BYTE	000
3090	003147	247	.BYTE	247
3091	003150	000	.BYTE	000
3092	003151	147	.BYTE	147

***** PATTERN V *****
PATV:

3095	003152	000	.BYTE	000
3096	003153	000	.BYTE	000
3097	003154	333	.BYTE	333
3098	003155	000	.BYTE	000
3099	003156	331	.BYTE	331
3100	003157	000	.BYTE	000
3101	003160	323	.BYTE	323
3102	003161	000	.BYTE	000
3103	003162	313	.BYTE	313
3104	003163	000	.BYTE	000
3105	003164	233	.BYTE	233
3106	003165	000	.BYTE	000
3107	003166	133	.BYTE	133
3108	003167	000	.BYTE	000
3109	003170	000	.BYTE	000
3110	003171	001	.BYTE	001
3111	003172	000	.BYTE	000
3112	003173	002	.BYTE	002
3113	003174	000	.BYTE	000
3114	003175	004	.BYTE	004
3115	003176	000	.BYTE	000
3116	003177	040	.BYTE	040
3117	003200	000	.BYTE	000
3118	003201	100	.BYTE	100
3119	003202	000	.BYTE	000
3120	003203	200	.BYTE	200
3121	003204	000	.BYTE	000
3122	003205	346	.BYTE	346
3123	003206	000	.BYTE	000
3124	003207	345	.BYTE	345
3125	003210	000	.BYTE	000
3126	003211	343	.BYTE	343
3127				
3128	003212	000	.BYTE	000
3129	003213	307	.BYTE	307
3130	003214	000	.BYTE	000

3131	003215	247	.BYTE	247
3132	003216	000	.BYTE	000
3133	003217	147	.BYTE	147

3134				
3135	003220		ENDPAT:	
3136			.EVEN	
3137				
3138				
3139				
3140				
3141				

3142 ;*** TEST MESSAGES TO BE TRANSMITTED ***

3143				
3144	003220	000400	MSG1:	TXSOM
3145	003222	000400		TXSOM
3146	003224	000000		000
3147	003226	000125		125
3148	003230	000252		252
3149	003232	000377		377
3150	003234	000000		000
3151	003236	001000		TXEOM
3152	003240	001000		TXEOM
3153	003242	001000		TXEOM
3154	003244	001000		TXEOM

3155				
3156	003246	000377	MSG4:	377
3157	003250	000000		000
3158	003252	000125		125
3159	003254	000252		252
3160	003256	001000		TXEOM
3161	003260	001000		TXEOM

3162

3163

3164

3165

3166

3167 ;*** RECEIVED DATA BUFFER (64. WORDS) ***

3168 003262 000100 RCVBUF: .BLKW 64.

3169

3170

3171

3172

3173

3174

3175

3176

3177

3178

3180
3181
3182
3183
3184
3185
3186
3187
3188
3189
3190
3191
(4)
(3)
(2)

003462
003462
003462 034115 030062 000063

```
.SBTTL GLOBAL TEXT SECTION
:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
:% THE GLOBAL TEXT SECTION CONTAINS FORMAT STATEMENTS,
:% MESSAGES, AND ASCII INFORMATION THAT ARE USED IN
:% MORE THAN ONE TEST.
:XXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXXX
:*****
:* NAMES OF DEVICES SUPPORTED BY PROGRAM
:*****
DEV TYP <M8203>

LSDVTYP::
.ASCIZ /M8203/
.EVEN
```


3216
3217
3218
3219
3220
3221
3222
3223
3224
3225
3226
3227
3228
3229
3230
3231
3232
3233
3234
3235
3236
3237
3238
3239
3240
3241
3242
3243
3244
3245
3246
3247
3248
3249
3250
3251
3252
3253
3254
3255
3256
3257
3258
3259
3260
3261
3262
3263
3264
3265
3266
3267
3268
3269
3270
3271

```
.SBTTL GLOBAL SUBROUTINES  
:////////////////////  
:/ THE GLOBAL SUBROUTINES ARE CALLED BY MORE THAN ONE TEST  
:////////////////////
```

```
:*****  
:* STPCLK - THIS SUBROUTINE FORCES THE DMC11 OR KMC11 MICROPROCESSOR TO  
:* EXECUTE AN INSTRUCTION WHICH IS PASSED IN THE WORD FOLLOWING THE CALL.  
:*****
```

```
STPCLK:  
003540 152777 000006 176702 BISB #ROMO!ROMI,@BSEL1 ;SET ROMO, ROMI BITS IN BSEL1  
003540 017677 000000 176702 MOV @ (SP),@SEL6 ;PUT INSTRUCTION INTO SEL6  
003554 152777 000007 176666 BISB #ROMO!ROMI!STEPMP,@BSEL1 ;SET ROMO, ROMI, STEPMP IN BSEL1  
003562 142777 000007 176660 BICB #ROMO!ROMI!STEPMP,@BSEL1 ;CLEAR ROMO, ROMI, STEPMP IN BSEL1  
003570 062716 000002 ADD #2,(SP) ;FIX UP RETURN PC  
003574 000207 RTS PC ;RETURN
```

```
:*****  
:* MSTCLR - THIS SUBROUTINE ISSUES A MASTER CLEAR AND SETS LULOOP  
:*****
```

```
MSTCLR:  
003576 010146 MOV R1,-(SP) ;SAVE R1  
003600 013746 MOV REGNUM,-(SP) ;SAVE LU REG NO.  
003604 112777 000100 176636 MOVB #MCLR,@BSEL1 ;SET MASTER CLEAR BIT  
003612 142777 000300 176630 BICB #RUN!MCLR,@BSEL1 ;CLEAR RUN AND MCLR BITS  
003620 012701 000024 MOV #20.,R1 ;INITIALIZE STALL COUNTER  
2$: NOP ;STALL IN LOOP FOR SEVERAL MICRO-SEC  
DEC R1  
BNE 2$  
003632 152777 000010 176610 BISB #LULOOP,@BSEL1 ;SET LU LOOP  
003640 012737 000013 002400 MOV #13,REGNUM ;SET LU REG NO. = 13  
003646 005037 002366 CLR WRIBYT  
003652 004737 003750 JSR PC,WRITLU ;CLEAR REG 13  
003656 012637 002400 MOV (SP)+,REGNUM ;RESTORE LU REG NO.  
003662 012601 MOV (SP)+,R1 ;RESTORE R1  
003664 005037 002432 CLR SAVLEN ;CLEAR CHAR LENGTH FROM SETUP  
003670 000207 RTS PC ;RETURN
```

```
:*****  
:* READLU - THIS SUBROUTINE FORCES THE DMC11 OR KMC11 MICROPROCESSOR  
:* TO EXECUTE AN INSTRUCTION WHICH READS THE LINE UNIT REG WHOSE  
:* NUMBER IS PASSED IN REGNUM, INTO REDBYT.  
:*****
```

3272 003672
3273 003672 010146
3274 003674 013701 002400
3275 003700 006301
3276 003702 006301
3277 003704 006301
3278 003706 006301
3279 003710 052701 000004
3280 003714 052701 021000
3281 003720 010137 003730
3282 003724 004737 003540
3283 003730 000000
3284 003732 117737 176516 002364
3285 003740 105037 002365
3286 003744 012601
3287 003746 000207
3288
3289
3290
3291
3292
3293
3294
3295
3296
3297
3298 003750
3299 003750 010146
3300 003752 013701 002400
3301 003756 052701 000100
3302 003762 052701 122000
3303 003766 010137 004010
3304 003772 105037 002367
3305 003776 113777 002366 176450
3306 004004 004737 003540
3307 004010 000000
3308 004012 012601
3309 004014 000207
3310
3311
3312
3313
3314
3315
3316
3317
3318
3319 004016 010146
3320 004020 013746 002400
3321 004024 012701 002302
3322 004030 012737 000010 002400
3323 004036 004737 003672
3324 004042 113721 002364
3325 004046 105021
3326 004050 005237 002400
3327 004054 023727 002400 000020

READLU:

```
MOV R1,-(SP) ;SAVE R1
MOV REGNUM,R1 ;GET LINE UNIT REG NUMBER
ASL R1 ;SHIFT INTO SOURCE BITS 4-7
ASL R1
ASL R1
ASL R1
BIS #4,R1 ;SET DESTINATION = BSEL4
BIS #MVIOX,R1 ;SET REST OF MOVE INSTRUCTION
MOV R1,2$ ;SET INSTRUCTION AS SUBROUTINE ARGUMENT
JSR PC,STPCLK ;EXECUTE MOVE INSTRUCTION
2$: .WORD 0 ;INSTRUCTION GOES HERE
MOVB @BSEL4,REDBYT ;GET LU REG CONTENTS INTO REDBYT
CLRB REDBYT+1 ;CLR HI BYTE OF STORAGE
MOV (SP)+,R1 ;RESTORE R1
RTS PC ;RETURN
```

```
*****
;* WRITLU - THIS SUBROUTINE FORCES THE DMC11 OR KMC11 MICROPROCESSOR TO
;* EXECUTE AN INSTRUCTION WHICH LOADS THE BYTE CONTAINED IN WRIBYT
;* INTO THE LU REG WHOSE NUMBER IS PASSED IN REGNUM,
*****
```

WRITLU:

```
MOV R1,-(SP) ;SAVE R1
MOV REGNUM,R1 ;GET LINE UNIT REG NUMBER
BIS #100,R1 ;SET SOURCE = BSEL4
BIS #MVIXO,R1 ;SET REST OF MOVE INSTRUCTION
MOV R1,2$ ;SET INSTRUCTION AS SUBROUTINE ARGUMENT
CLRB WRIBYT+1 ;CLR HI BYTE OF STORAGE
MOVB WRIBYT,@BSEL4 ;LOAD BYTE INTO BSEL4
JSR PC,STPCLK ;EXECUTE MOVE INSTRUCTION
2$: .WORD 0
MOV (SP)+,R1 ;RESTORE R1
RTS PC ;RETURN
```

```
*****
;* GETREG - THIS SUBROUTINE READS THE LINE UNIT REGISTERS 10-17 INTO THE
;* REGISTER STORAGE TABLE (LUREG:).
*****
```

GETREG:

```
MOV R1,-(SP) ;SAVE R1
MOV REGNUM,-(SP) ;SAVE CURRENT REG NO.
MOV #LUR10,R1 ;INIT POINTER TO REG STORAGE TABLE
MOV #10,REGNUM ;INIT LU REG NO. TO 10
3$: JSR PC,READLU ;READ A LINE UNIT REG
MOVB REDBYT,(R1)+ ;PUT BYTE READ INTO TABLE
CLRB (R1)+ ;CLEAR UPPER BYTE OF TABLE ENTRY
INC REGNUM ;INCREMENT REG NO.
CMP REGNUM,#20 ;SEE IF ALL REGS READ YET
```


3328 004062 002765
3329 004064 012637 002400
3330 004070 012601
3331 004072 000207

BLT 3\$;BR IF NOT
MOV (SP)+,REGNUM ;RESTORE CURRENT REG NO.
MOV (SP)+,R1 ;RESTORE R1
RTS PC ;RETURN

3332
3333
3334
3335
3336
3337

* LOOPIN - THIS SUBROUTINE PLACES THE MICROPROCESSOR IN A LOOP ON AN
* INSTRUCTION, BY MOVING THE INSTRUCTION FROM THE WORD FOLLOWING THE CALL
* INTO SEL6, AND SETTING RUN AND ROMI IN BSEL1. THE SUBROUTINE RETURNS
* WITH THE MICROPROCESSOR STUCK IN THE LOOP, AND IF IT IS DESIRED TO
* TERMINATE THE LOOP, THE PDP-11 PROGRAM MUST CLEAR THE RUN BIT IN
* BSEL1, OR CALL SUBROUTINE MSTCLR TO DO THIS.

3345 074
3346 00074 152777 000006 176346
3347 00102 017677 000000 176346
3348 004110 152777 000206 176332
3349 004116 062716 000002
3350 004122 000207

LOOPIN:
BISB #ROMO!ROMI,@BSEL1 ;SET ROMO, ROMI BITS IN BSEL1
MOV @ (SP),@SEL6 ;PUT MICROINSTRUCTION INTO SEL6
BISB #RUN!ROMO!ROMI,@BSEL1 ;SET RUN, ROMO, ROMI IN BSEL1
ADD #2,(SP) ;FIX UP RETURN PC
RTS PC ;RETURN WITH MICROPROCESSOR STUCK IN SINGLE
; INSTRUCTION LOOP

3351
3352
3353
3354
3355
3356
3357

* READAX - THIS SUBROUTINE READS THE USYRT REG PAIR WHOSE NUMBER (0-3)
* IS PASSED IN BITS 1,2 OF AXNUM ON ENTRY, AND RETURNS THE BYTES READ IN
* RAX15 AND RAX16. IF THE LINE UNIT DOES NOT RESPOND WITH READY IN REG 14,
* RRDYTO BIT IS SET IN ERROR1 ON RETURN.

3363 004124 010146
3364 004126 013746 002400
3365 004132 042737 000001 002420
3366 004140 012737 000014 002400
3367 004146 113737 002402 002366
3368 004154 006237 002366
3369 004160 152737 000024 002366
3370 004166 053737 002426 002366
3371 004174 004737 003750
3372 004200 005001
3373 004202 004737 003672 6\$:
3374 004206 132737 000200 002364
3375 004214 001006
3376 004216 005201
3377 004220 001370
3378 004222 052737 000001 002420
3379 004230 000424
3380 004232 012737 000015 002400 9\$:
3381 004240 004737 003672
3382 004244 113737 002364 002370
3383 004252 105037 002371

READAX: MOV R1,-(SP) ;SAVE R1
MOV REGNUM,-(SP) ;STORE CURRENT REG NO.
BIC #RRDYTO,ERROR1 ;CLEAR ERROR BIT
MOV #14,REGNUM ;SET LU REG NO. = 14
MOVB AXNUM,WRIBYT ;SET UP AX REG NO. BITS
ASR WRIBYT
BISB #RDAX!ENAX,WRIBYT ;SET UP BITS TO LOAD INTO REG 14
BIS DISILO,WRIBYT ;SET PROPER STATE OF DISSI BIT
JSR PC,WRITLU ;SET RDAX AND ENAX IN REG 14
CLR R1 ;INIT TIMER
JSR PC,READLU ;READ REG 14
BITB #READY,REDBYT ;SEE IF READY BIT SET IN REG 14 YET
BNE 9\$;BR IF READY SET
INC R1 ;INCR TIMER
BNE 6\$;BR IF TIMER DIDN'T TIME OUT YET
BIS #RRDYTO,ERROR1 ;SET ERROR FLAG FOR TIME OUT ON READ RDY
BR 12\$;BR TO RETURN
9\$: MOV #15,REGNUM ;SET REG NO. = 15
JSR PC,READLU ;READ REG 15
MOVB REDBYT,RAX15 ;STORE REG AX-15
CLRB RAX15+1 ;CLR HI BYTE OF STORAGE

3384 004256 012737 000016 002400
3385 004264 004737 003672
3386 004270 113737 002364 002372
3387 004276 105037 002373
3388 004302 012637 002400
3389 004306 012601
3390 004310 000207
3391
3392
3393
3394
3395
3396
3397
3398
3399
3400
3401 004312 010146
3402 004314 013746 002400
3403 004320 042737 000002 002420
3404 004326 012737 000014 002400
3405 004334 113737 002402 002366
3406 004342 006237 002366
3407 004346 053737 002426 002366
3408 004354 004737 003750
3409 004360 012737 000015 002400
3410 004366 105037 002375
3411 004372 113737 002374 002366
3412 004400 004737 003750
3413 004404 005237 002400
3414 004410 105037 002377
3415 004414 113737 002376 002366
3416 004422 004737 003750
3417 004426 012737 000014 002400
3418 004434 113737 002402 002366
3419 004442 006237 002366
3420 004446 152737 000014 002366
3421 004454 053737 002426 002366
3422 004462 004737 003750
3423 004466 005001
3424 004470 004737 003672
3425 004474 132737 000200 002364
3426 004502 001005
3427 004504 005201
3428 004506 001370
3429 004510 052737 000002 002420
3430 004516 012637 002400
3431 004522 012601
3432 004524 000207
3433
3434
3435
3436
3437
3438
3439

```
MOV #16,REGNUM ;SET REG NO. = 16
JSR PC,READLU ;READ REG 16
MOVB REDBYT,RAX16 ;STORE REG AX-16
CLRB RAX16+1 ;CLR HI BYTE OF STORAGE
12$: MOV (SP)+,REGNUM ;RESTORE CURRENT REG NO.
MOV (SP)+,R1 ;RESTORE R1
RTS PC ;RETURN

:*****
;* WRITAX - THIS SUBROUTINE WRITES THE USYRT REG PAIR WHOSE NUMBER (0-3) IS
;* PASSED IN BITS 1,2 OF AXNUM ON ENTRY, WITH THE DATA FROM WAX15 AND
;* WAX16. IF LINE UNIT DOES NOT RESPOND WITH READY IN REG 14, WRDYTO BIT
;* IS SET IN ERROR1 ON RETURN.
:*****
WRITAX: MOV R1,-(SP) ;SAVE R1
MOV REGNUM,-(SP) ;SAVE CURRENT REG NO.
BIC #WRDYTO,ERROR1 ;CLEAR ERROR BIT
MOV #14,REGNUM ;SET LU REG NO. = 14
MOVB AXNUM,WRIBYT ;SET AX REG NO. BITS
ASR WRIBYT
BIS DISILO,WRIBYT ;SET PROPER STATE OF DISSI BIT
JSR PC,WRITLU ;SET AX NO. BITS IN REG 14
MOV #15,REGNUM ;SET REG NO. = 15
CLRB WAX15+1 ;CLR HI BYTE OF STORAGE
MOVB WAX15,WRIBYT ;SET UP BYTE TO WRITE INTO REG 15
JSR PC,WRITLU ;WRITE BYTE INTO REG 15
INC REGNUM ;SET REG NO. = 16
CLRB WAX16+1 ;CLR HI BYTE OF STORAGE
MOVB WAX16,WRIBYT ;SET UP BYTE TO WRITE INTO REG 16
JSR PC,WRITLU ;WRITE BYTE INTO REG 16
MOV #14,REGNUM ;SET REG NO. = 14
MOVB AXNUM,WRIBYT ;SET AX REG NO. BITS
ASR WRIBYT
BISB #ENAX!WAX,WRIBYT ;SET UP BITS TO LOAD INTO REG 14
BIS DISILO,WRIBYT ;SET PROPER STATE OF DISSI BIT
JSR PC,WRITLU ;SET ENAX AND WAX IN REG 14
CLR R1 ;INIT PROGRAM TIMER
6$: JSR PC,READLU ;READ REG 14
BITB #READY,REDBYT ;SEE IF READY BIT SET IN REG 14 YET
BNE 9$ ;BR IF READY SET
INC R1 ;INCR TIMER
BNE 6$ ;BR IF TIMER DIDN'T TIME OUT YET
BIS #WRDYTO,ERROR1 ;SET ERROR FLAG BIT FOR TIME OUT ON WRITE RDY
9$: MOV (SP)+,REGNUM ;RESTORE CURRENT REG NO.
MOV (SP)+,R1 ;RESTORE R1
RTS PC ;RETURN
```

```
:*****
;* GETALL - THIS SUBROUTINE READS THE LINE UNIT REGS 10-17 AND THE EXTENDED
```

3440
3441
3442 004526 010146
3443 004530 013746 002402
3444 004534 012737 014437 002530
3445 004542 032737 000001 002402
3446 004550 001403
3447 004552 012737 014442 002530
3448 004560 004737 004016
3449 004564 142777 000010 175656
3450 004572 012701 002322
3451 004576 005037 002402
3452 004602 004737 004124
3453 004606 113721 002370
3454 004612 105021
3455 004614 113721 002372
3456 004620 105021
3457 004622 062737 000002 002402
3458 004630 023727 002402 000010
3459 004636 002761
3460 004640 012637 002402
3461 004644 012601
3462 004646 013737 002402 002532
3463 004654 006237 002532
3464 004660 000207
3465
3466
3467
3468
3469
3470
3471
3472
3473
3474
3475
3476
3477
3478 004662 013746 002400
3479 004666 013746 002352
3480 004672 005737 002352
3481 004676 001006
3482 004700 016637 000004 002352
3483 004706 162737 000004 002352
3484 004714 012737 000011 002400
3485 004722 004737 003672
3486 004726 032776 000001 000004
3487 004734 001413
3488 004736 132737 000020 002364
3489 004744 001022
3490 004746 004737 004526
3491
3492 004752
(4) 004752 104455
(5) 004754 000007
(5) 004756 012376

```
;*   REGISTERS AX0-AX3 INTO REGISTER STORAGE TABLE (LUREG:).  
:*****  
GETALL: MOV   R1,-(SP)           ;SAVE R1  
        MOV   AXNUM,-(SP)       ;SAVE CURRENT AX REG BYTE NO.  
        MOV   #DH5,TMPO         ;SET AX LO BYTE NO.  
        BIT   #BIT0,AXNUM       ;SEE IF LO OR HI BYTE  
        BEQ   1$                ;BR IF LO BYTE  
        MOV   #DH6,TMPO         ;SET AX HI BYTE NO.  
1$:     JSR   PC,GETREG         ;READ AND STORE REGS 10-17  
        BICB  #LULOOP,@BSEL1    ;CLEAR LULOOP  
        MOV   #AX0.15,R1        ;INIT POINTER TO REG STORAGE TABLE  
        CLR   AXNUM             ;INIT AX REG BYTE NO. TO 0  
3$:     JSR   PC,READAX         ;READ 2 AX REG BYTES  
        MOVB  RAX15,(R1)+        ;PUT LO BYTE READ INTO TABLE  
        CLRB  (R1)+             ;CLEAR UPPER BYTE OF TABLE ENTRY  
        MOVB  RAX16,(R1)+        ;PUT HI BYTE READ INTO TABLE  
        CLRB  (R1)+             ;CLEAR UPPER BYTE OF TABLE ENTRY  
        ADD   #2,AXNUM          ;INCR AX REG BYTE NO.  
        CMP   AXNUM,#10         ;SEE IF ALL REGS READ YET  
        BLT   3$                ;BR IF NOT  
        MOV   (SP)+,AXNUM        ;RESTORE CURRENT AX REG BYTE NO.  
        MOV   (SP)+,R1          ;RESTORE R1  
        MOV   AXNUM,TMP1  
        ASR   TMP1              ;GET EXTENDED REG NO. FOR PRINTOUT  
        RTS   PC                ;RETURN
```

```
*****  
;* OSIRDY - THIS SUBROUTINE CHECKS FOR THE PROPER STATES OF ORDY (REG 11)  
;* AND OCOR (REG 17) AND REPORTS AN ERROR IF EITHER IS NOT PROPERLY SET  
;* AS PASSED IN BIT 0 (ORDY) AND BIT 1 (OCOR) OF THE WORD FOLLOWING THE  
;* CALL.  
;* IF AN ERROR OCCURS, A RETURN IS MADE TO THE TEST, AT THE ADDRESS IN  
;* RETADR.  
*****  
OSIRDY: MOV   REGNUM,-(SP)       ;SAVE LU REG NO.  
        MOV   SUBRPC,-(SP)  
        TST   SUBRPC  
        BNE  1$                ;SEE IF THIS IS A NESTED CALL  
        MOV   4(SP),SUBRPC       ;BR IF YES  
1$:     SUB   #4,SUBRPC          ;GET PC OF SUBROUTINE CALL  
        MOV   #11,REGNUM        ;SET REG NO. TO 11  
        JSR   PC,READLU         ;READ REG 11  
        BIT   #BIT0,@4(SP)      ;GET EXPECTED STATE OF ORDY  
        BEQ   3$                ;BR IF EXPECTED ORDY = 0  
        BITB  #ORDY,REDBYT      ;SEE IF ORDY = 1  
        BNE  9$                ;BR IF ORDY = 1  
        JSR   PC,GETALL         ;GET REGS FOR PRINTOUT  
:REPORT ORDY NOT SET  
ERRDF  7,EM7,ERR4
```

IRAP C\$ERDF
.WORD ?
.WORD EM7

```

(5) 004760 015624                                     .WORD  ERR4
3493 004762 000451
3494 004764 132737 000020 002364 3$:  BR      16$      ;TAKE ERROR RETURN
;SEE IF ORDY = 0
3495 004772 001407                                     ;BR IF ORDY = 0
3496 004774 004737 004526                                     ;GET REGS FOR PRINTOUT
3497                                     ;REPORT ORDY NOT CLEARED
ERRDF 8,EM8,ERR4
3498 005000
(4) 005000 104455                                     TRAP  C$ERDF
(5) 005002 000010                                     .WORD  8
(5) 005004 012413                                     .WORD  EM8
(5) 005006 015624                                     .WORD  ERR4
3499 005010 000436
3500 005012 012737 000017 002400 9$:  BR      16$      ;TAKE ERROR RETURN
;SET REG NO. = 17
3501 005020 004737 003672                                     ;READ LU REG 17
3502 005024 132776 000002 000004  JSR     PC,READLU
;GET EXPECTED STATE OF OCOR
3503 005032 001413                                     ;BR IF EXPECTED OCOR = 0
3504 005034 132737 000020 002364  BITB   #OCOR,REDBYT
;SEE IF OCOR = 1
3505 005042 001031                                     ;BR IF OCOR = 1
3506 005044 004737 004526                                     ;GET REGS FOR PRINTOUT
3507                                     ;REPORT OCOR NOT SET
ERRDF 9,EM9,ERR4
3508 005050
(4) 005050 104455                                     TRAP  C$ERDF
(5) 005052 000011                                     .WORD  9
(5) 005054 012434                                     .WORD  EM9
(5) 005056 015624                                     .WORD  ERR4
3509 005060 000412
3510 005062 132737 000020 002364 12$:  BR      16$      ;TAKE ERROR RETURN
;SEE IF OCOR = 0
3511 005070 001416                                     ;BR IF OCOR = 0
3512 005072 004737 004526                                     ;GET REGS FOR PRINTOUT
3513                                     ;REPORT OCOR NOT CLEARED
ERRDF 10,EM10,ERR4
3514 005076
(4) 005076 104455                                     TRAP  C$ERDF
(5) 005100 000012                                     .WORD  10
(5) 005102 012451                                     .WORD  EM10
(5) 005104 015624                                     .WORD  ERR4
3515 005106 016637 000002 002400 16$:  MOV     2(SP),REGNUM
;RESTORE LU REG NO.
3516 005114 013706 002346                                     ;RESTORE STACK POINTER TO BASE LEVEL
3517 005120 013746 002362                                     ;FIX ERROR RETURN PC
3518 005124 000407
3519 005126 062766 000002 000004 20$:  BR      23$
;FIX UP ERROR-FREE RETURN PC
3520 005134 012637 002352
3521 005140 012637 002400
3522 005144 000207 23$:  MOV     (SP)+,SUBRPC
;RESTORE LU REG NO.
RTS     PC
;RETURN
3523
3524
3525
3526
3527
3528
3529
3530
3531 005146 010146
3532 005150 012701 000310
3533 005154 005301
3534 005156 001376
3535 005160 012601
;*****
;* WAIT50 - THIS SUBROUTINE STALLS FOR AT LEAST 50 MICRO-SEC, AND THEN RETURNS.
;*****
WAIT50: MOV     R1,-(SP)
;SAVE R1
MOV     #200.,R1
;INIT COUNTER
3$:  DEC     R1
;DECREMENT COUNTER
BNE     3$
;BR IF NOT DONE YET
MOV     (SP)+,R1
;RESTORE R1

```

3536 005162 000207

RTS PC ;RETURN

3537
3538
3539
3540
3541

3542

3543

3544

3545 005164 000240

3546 005166 000240

3547 005170 000240

3548 005172 000207

3549

3550

3551

3552

3553

3554

3555

3556

3557

3558 005174 013746 002400

3559 005200 042737 170000 002422

3560 005206 012737 000011 002400

3561 005214 113737 002423 002366

3562 005222 004737 003750

3563 005226 012737 000010 002400

3564 005234 113737 002422 002366

3565 005242 004737 003750

3566 005246 012637 002400

3567

3568

3569

3570

3571

3572

3573

3574

3575

3576

3577

3578

3579 005254 010146

3580 005256 017601 000002

3581 005262 001426

3582 005264 100006

3583 005266 042701 100000

3584 005272 005737 002430

3585 005276 001401

3586 005300 005301

3587 005302 152777 000010 175140 2\$:

3588 005310 152777 000020 175132 3\$:

3589 005316 004737 005164

3590 005322 142777 000020 175120

3591 005330 004737 005164

;* STALL - THIS SUBROUTINE STALLS FOR ABOUT A MICRO-SEC.

STALL: NOP
NOP
NOP
RTS PC

;* LDTXSI - THIS SUBROUTINE LOADS THE TX SILO (REGS 10,11) WITH THE DATA PASSED
;* IN BITS 0-11 OF TXWORD.

LDTXSI: MOV REGNUM, -(SP) ;SAVE LU REG NO.
BIC #170000, TXWORD ;CLEAR UNUSED BITS
MOV #11, REGNUM ;SET REG NO. = 11
MOVB TXWORD+1, WRIBYT ;SET DATA TO BE WRITTEN INTO REG 11
JSR PC, WRITLU ;LOAD DATA INTO REG 11
MOV #10, REGNUM ;SET REG NO. = 10
MOVB TXWORD, WRIBYT ;SET DATA TO BE WRITTEN INTO REG 10
JSR PC, WRITLU ;LOAD DATA INTO REG 10
MOV (SP)+, REGNUM ;RESTORE LU REG NO.
RTS PC ;RETURN

;* STPLU - THIS SUBROUTINE CLOCKS THE LINE UNIT FOR THE NO. OF CYCLES PASSED
;* IN BITS 0-14 OF THE WORD FOLLOWING THE CALL.
;* IF BIT 15 = 1, A CHECK IS MADE TO DETERMINE IF THE USYRT CHIP TYPE
;* REQUIRES DECREMENTING THE NO. OF CYCLES BY 1.

STPLU: MOV R1, -(SP) ;SAVE R1
MOV @2(SP), R1 ;GET DESIRED NO. OF CYCLES
BEQ 6\$;IF DESIRED CYCLES = 0, RETURN
BPL 2\$;BR IF CHIP TYPE CHECK NOT NECESSARY
BIC #BIT15, R1 ;CLEAR FLAG BIT
TST CHPTYP ;SEE IF SIG USYRT
BEQ 2\$;BR IF YES
DEC R1 ;DECREMENT CYCLE COUNT
BISB #LULOOP, @BSEL1 ;SET LU LOOP BIT
BISB #STEPLU, @BSEL1 ;SET THE STEPLU BIT (CLOCK THE TRANSMITTER)
JSR PC, STALL ;STALL
BICB #STEPLU, @BSEL1 ;CLEAR THE STEPLU BIT (CLOCK THE RECEIVER)
JSR PC, STALL ;STALL

3592 005334 005301 DEC R1 ;DECREMENT CYCLE COUNTER
3593 005336 001364 BNE 3\$;BR IF NOT DONE YET
3594 005340 062766 000002 000002 6\$: ADD #2,2(SP) ;FIX UP RETURN PC
3595 005346 012601 MOV (SP)+,R1 ;RESTORE R1
3596 005350 000207 RTS PC ;RETURN

3597
3598
3599
3600
3601

3602 :*****
3603 :* OACTIV - THIS SUBROUTINE CHECKS FOR THE PROPER STATE OF OACT (REG 11) AND
3604 :* REPORTS AN ERROR IF IT IS NOT PROPERLY SET TO THE STATE OF BIT 0 IN THE
3605 :* WORD FOLLOWING THE CALL.
3606 :*****

3607 005352 013746 002400 OACTIV: MOV REGNUM,-(SP) ;SAVE LU REG NO.
3608 005356 013746 002352 MOV SUBRPC,-(SP)
3609 005362 005737 002352 TST SUBRPC ;SEE IF THIS IS A NESTED CALL
3610 005366 001006 BNE 1\$;BR IF YES
3611 005370 016637 000004 002352 MOV 4(SP),SUBRPC
3612 005376 162737 000004 002352 SUB #4,SUBRPC ;GET PC OF SUBROUTINE CALL
3613 005404 012737 000011 002400 1\$: MOV #11,REGNUM ;SET REG NO. = 11
3614 005412 004737 003672 JSR PC,READLU ;READ REG 11
3615 005416 032776 000001 000004 BIT #BIT0,#4(SP) ;GET EXPECTED STATE OF OACT
3616 005424 001413 BEQ 3\$;BR IF EXPECTED OACT = 0
3617 005426 132737 000100 002364 BITB #OACT,REDBYT ;SEE IF OACT = 1
3618 005434 001031 BNE 9\$;BR IF OACT = 1
3619 005436 004737 004526 JSR PC,GETALL ;GET REGS FOR PRINTOUT

3620 ;REPORT OACT NOT SET
3621 005442 ERRDF 11,EM11,ERR4
(4) 005442 104455 TRAP C\$ERDF
(5) 005444 000013 .WORD 11
(5) 005446 012472 .WORD EM11
(5) 005450 015624 .WORD ERR4

3622 005452 000412 BR 6\$;TAKE ERROR RETURN
3623 005454 132737 000100 002364 3\$: BITB #OACT,REDBYT ;SEE IF OACT = 0
3624 005462 001416 BEQ 9\$;BR IF OACT = 0
3625 005464 004737 004526 JSR PC,GETALL ;GET REGS FOR PRINTOUT
3626 ;REPORT OACT NOT CLEARED
ERRDF 12,EM12,ERR4

3627 005470 TRAP C\$ERDF
(4) 005470 104455 .WORD 12
(5) 005472 000014 .WORD EM12
(5) 005474 012507 .WORD ERR4
(5) 005476 015624 .WORD ERR4

3628 005500 016637 000002 002400 6\$: MOV 2(SP),REGNUM ;RESTORE LU REG NO.
3629 005506 013706 J2346 MOV PSTACK,SP ;RESTORE PROGRAM STACK TO BASE LEVEL
3630 005512 013746 002362 MOV RETADR,-(SP) ;FIX UP ERROR RETURN PC
3631 005516 000407 BR 12\$
3632 005520 062766 000002 000004 9\$: ADD #2,4(SP) ;FIX UP ERROR-FREE RETURN PC
3633 005526 012637 002352 MOV (SP)+,SUBRPC
3634 005532 012637 002400 MOV (SP)+,REGNUM ;RESTORE LU REG NO.
3635 005536 000207 12\$: RTS PC ;RETURN

3636
3637
3638
3639

3640
 3641
 3642
 3643
 3644
 3645
 3646
 3647
 3648
 3649
 3650
 3651
 3652
 3653
 3654
 3655
 3656
 3657
 3658
 3659
 3660
 3661
 3662
 3663
 3664
 3665
 3666
 3667
 3668
 3669
 3670
 3671
 3672
 3673
 3674
 3675
 3676
 3677
 3678
 3679
 3680
 3681
 3682
 3683
 3684
 3685
 3686
 3687
 3688
 3689
 3690
 3691
 3692
 3693
 3694
 3695

005540 010146
 005542 013746 002400
 005546 013746 002402
 005552 016637 000006 002352
 005560 162737 000004 002352
 005566 004737 003576
 005572 004737 004662
 005576 000001
 005600 004737 005352
 005604 000000
 005606 012737 000004 002402
 005614 117637 000006 002374
 005622 012737 000400 002422
 005630 113737 002374 002422
 005636 005037 002376
 005642 004737 004312
 005646 012737 000017 002400
 005654 062766 000002 000006
 005662 117637 000006 002366
 005670 004737 003750
 005674 004737 005174
 005700 004737 005174
 005704 004737 005146
 005710 004737 004662
 005714 000003
 005716 004737 005352
 005722 000000
 005724 005001
 005726 012737 000011 002400
 005734 152777 000010 174506 6\$:
 005742 152777 000020 174500
 005750 004737 005164
 005754 004737 003672
 005760 132737 000100 002364
 005766 001014
 005770 142777 000020 174452
 005776 004737 005164
 006002 005201
 006004 020127 000003
 006010 002751
 006012 004737 005352
 006016 000001
 006020 012737 000017 002400 9\$:
 006026 005037 002430
 006032 004737 003672

```

*****
* INITRN - THIS SUBROUTINE INITIATES TRANSMISSION OF A MESSAGE, BY DOING A
* MASTER CLEAR, LOADING AX2-15 AND REG 17 WITH THE DATA PASSED IN THE 2
* WORDS FOLLOWING THE CALL, LOADING 2 SOM CHARS INTO THE TX SILO, AND
* CLOCKING THE LINE UNIT UNTIL THE FIRST SYNCH OR FLAG HAS BEEN SERIALIZED
* IN THE USYRT. THE PROGRAM MONITORS ORDY,OCOR, AND OACT FOR VALID STATES,
* THROUGHOUT THE PROCESS.
* IF THE SUBROUTINE DETECTS AN ERROR, A RETURN IS MADE TO THE TEST, AT THE
* ADDRESS CONTAINED IN RETADR.
*****
INITRN: MOV R1,-(SP) ;SAVE R1
MOV REGNUM,-(SP) ;SAVE LU REG NO.
MOV AXNUM,-(SP) ;SAVE AX BYTE NO.
MOV 6(SP),SUBRPC
SUB #4,SUBRPC ;GET PC OF SUBR CALL
JSR PC,MSTCLR ;ISSUE A MASTER CLEAR
JSR PC,OSIRDY ;CHECK ORDY=1, OCOR=0
1
JSR PC,OACTIV ;CHK OACT=0
0
MOV #4,AXNUM ;SET AX BYTE NO. = 4 FOR AX2
MOVB @6(SP),WAX15 ;SET DATA BYTE TO LOAD INTO AX2-15
MOV #TXSOM,TXWORD ;SET TSOM BIT
MOVB WAX15,TXWORD ;SET SYNCH CHAR
CLR WAX16
JSR PC,WRITAX ;LOAD AX2
MOV #17,REGNUM ;SET REG NO. = 17
ADD #2,6(SP) ;INCR POINTER TO NEXT DATA BYTE
MOVB @6(SP),WRIBYT ;SET DATA BYTE TO LOAD INTO REG 17
JSR PC,WRITLU ;LOAD REG 17
JSR PC,LDTXSI ;LOAD THE SILO WITH SOM CHAR
JSR PC,LDTXSI ;LOAD ANOTHER SOM INTO SILO
JSR PC,WAIT50 ;WAIT FOR DATA TO RIPPLE
JSR PC,OSIRDY ;CHK ORDY=1, OCOR=1
3
JSR PC,OACTIV ;CHK FOR OACT = 0
0
CLR R1 ;INIT CYCLE COUNTER
MOV #11,REGNUM ;SET LU REG NO. = 11
BISB #LULOOP,@BSEL1 ;SET LINE UNIT LOOP BIT
BISB #STEPLU,@BSEL1 ;SET CLOCK BIT
JSR PC,STALL ;STALL FOR MICRO-SEC
JSR PC,READLU ;READ REG 11
BITB #OACT,REDBYT ;SEE IF OACT = 1 YET
BNE 9$ ;BR IF OACT = 1
BICB #STEPLU,@BSEL1 ;CLEAR CLOCK BIT
JSR PC,STALL ;STALL FOR A MICRO-SEC
INC R1 ;INCR CYCLE COUNT
CMP R1,#3 ;SEE IF 3 CYCLES DONE YET
BLT 6$ ;BR IF NOT
JSR PC,OACTIV ;CHK FOR OACT = 1
1
MOV #17,REGNUM ;SET REG NO. = 17
CLR CHPTYP ;CLEAR USYRT CHIP INDICATOR
JSR PC,READLU ;READ REG 17

```

```
3696 006036 132737 000020 002364 BITB #OCOR,REDBYT ;CHK FOR OCOR CLEARED YET
3697 006044 001403 BEQ 12$ ;BR IF YES - IT IS SIG CHIP
3698 006046 012737 000001 002430 MOV #1,CHPTYP ;SET INDICATOR FOR OTHER CHIP TYPE
3699 006054 142777 000020 174366 12$: BICB #STEPLU,@BSEL1 ;CLEAR CLOCK BIT
3700 006062 004737 005164 JSR PC,STALL ;STALL FOR MICRO-SEC
3701 006066 004737 004662 JSR PC,OSIRDY ;CHK FOR ORDY = 1, OCOR = 0
3702 006072 000001 1 ADD #2,6(SP) ;FIX UP RETURN PC
3703 006074 062766 000002 000006 MOV (SP)+,AXNUM ;RESTORE AX BYTE NO.
3704 006102 012637 002402 MOV (SP)+,REGNUM ;RESTORE LU REG NO.
3705 006106 012637 002400 MOV (SP)+,R1 ;RESTORE R1
3706 006112 012601 CLR SUBRPC ;CLEAR SUBR CALL PC
3707 006114 005037 002352 RTS PC ;RETURN
3708 006120 000207
```

3709
3710
3711
3712
3713
3714

```
*****
* TXCHAR - THIS SUBROUTINE INITIATES TRANSMISSION OF A CHARACTER, BY LOADING
* THE TX SILO WITH DATA PASSED IN BITS 0-11 OF THE WORD FOLLOWING THE CALL
* AND CLOCKS THE LINE UNIT WITH THE NUMBER OF CYCLES PASSED IN BITS 0-14
* OF THE SECOND WORD FOLLOWING THE CALL. IF BIT 15 = 1, A CHK IS MADE TO
* DETERMINE IF THE USYRT CHIP TYPE REQUIRES DECREMENTING THE NO. OF CYCLES
* BY 1. THE PROGRAM CHECKS FOR VALID STATES OF ORDY,
* OCOR, AND OACT THROUGHOUT THE PROCESS.
* IF AN ERROR IS DETECTED, A RETURN IS MADE TO THE TEST, AT THE ADDRESS
* CONTAINED IN RETADR.
*****
```

```
3724
3725 006122 010146 TXCHAR: MOV R1,-(SP) ;SAVE R1
3726 006124 010246 MOV R2,-(SP) ;SAVE R2
3727 006126 016637 000004 002352 MOV 4(SP),SUBRPC
3728 006134 162737 000004 002352 SUB #4,SUBRPC ;GET PC OF SUBR CALL
3729 006142 017637 000004 002422 MOV @4(SP),TXWORD ;GET DATA TO BE TRANSMITTED
3730 006150 004737 005174 JSR PC,LDTXSI ;LOAD THE TX SILO WITH THE DATA
3731 006154 004737 005146 JSR PC,WAIT50 ;WAIT FOR DATA TO RIPPLE DOWN SILO
3732 006160 062766 000002 000004 ADD #2,4(SP) ;INCR POINTER
3733 006166 005001 CLR R1 ;INIT CYCLE COUNT
3734 006170 017602 000004 MOV @4(SP),R2 ;GET DESIRED NO. OF CYCLES
3735 006174 005702 TST R2 ;SEE IF CHIP TYPE CHK SHOULD BE MADE
3736 006176 100006 BPL 9$ ;BR IF NOT
3737 006200 042702 100000 BIC #BIT15,R2 ;CLEAR FLAG BIT
3738 006204 005737 002430 TST CHPTYP ;SEE IF SIG USYRT
3739 006210 001401 BEQ 9$ ;BR IF YES
3740 006212 005302 DEC R2 ;DECREMENT NO. OF CYCLES
3741 006214 004737 005352 9$: JSR PC,OACTIV ;CHK OACT = 1
3742 006220 000001 1
3743 006222 020102 CMP R1,R2 ;SEE IF REQUIRED CYCLES DONE YET
3744 006224 001410 BEQ 12$ ;BR IF YES
3745 006226 004737 004662 JSR PC,OSIRDY ;CHK ORDY=1, OCOR=1
3746 006232 000003 3
3747 006234 004737 005254 JSR PC,STPLU ;STEP LU ONE CYCLE
3748 006240 000001 1
3749 006242 005201 INC R1 ;INCR CYCLE COUNT
3750 006244 000763 BR 9$
3751 006246 004737 004662 12$: JSR PC,OSIRDY ;CHK ORDY=1, OCOR=0
```



```
3752 006252 000001          1  
3753 006254 062766 000002 000004    ADD    #2,4(SP)      ;FIX UP RETURN PC  
3754 006262 005037 002352          CLR    SUBRPC       ;CLEAR SUBR CALL PC  
3755 006266 012602          MOV    (SP)+,R2     ;RESTORE R2  
3756 006270 012601          MOV    (SP)+,R1     ;RESTORE R1  
3757 006272 000207          RTS    PC           ;RETURN  
3758  
3759  
3760  
3761  
3762  
3763
```

```
*****  
* ENDTRN - THIS SUBROUTINE CLEARS THE TRANSMITTER BY SETTING OC. THE PROGRAM  
*   WAITS FOR 50 US, AND CHECKS FOR ORDY=1, OCOR=0, OACT=0, RTS=0.  
*****
```

```
3764  
3765  
3766  
3767 006274 010146          ENDTRN: MOV    R1,-(SP)      ;SAVE R1  
3768 006276 013746 002400    MOV    REGNUM,-(SP)     ;SAVE LU REG NO.  
3769 006302 016637 000004 002352    MOV    4(SP),SUBRPC    ;  
3770 006310 162737 000004 002352    SUB    #4,SUBRPC      ;GET PC OF SUBROUTINE CALL  
3771 006316 012737 000011 002400    MOV    #11,REGNUM     ;SET LU REG NO. = 11  
3772 006324 012737 000200 002366    MOV    #OC,WRIBYT     ;SET OC IN DATA  
3773 006332 004737 003750          JSR    PC,WRITLU      ;SET OC IN REG 11  
3774 006336 004737 005146          JSR    PC,WAIT50     ;STALL FOR >50 US.  
3775 006342 004737 004662          JSR    PC,OSIRDY     ;CHK ORDY=1, OCOR=0  
3776 006346 000001          1  
3777 006350 004737 005352          JSR    PC,OACTIV     ;CHK OACT = 0  
3778 006354 000000          0  
3779 006356 012737 000013 002400    MOV    #13,REGNUM     ;SET REG NO. = 13  
3780 006364 004737 003672          JSR    PC,READLU     ;READ REG 13  
3781 006370 032737 000040 002364    BIT    #RTS,REDBYT   ;CHK FOR RTS = 0  
3782 006376 001406          BEQ    3$           ;BR IF RTS = 0  
3783 006400 004737 004526          JSR    PC,GETALL     ;GET REGS FOR PRINTOUT  
3784  
;REPORT RTS NOT CLEARED  
3785 006404          ERRDF 65,EM65,ERR4
```

```
TRAP  CSERDF  
.WORD 65  
.WORD EM65  
.WORD ERR4
```

```
(4) 006404 104455  
(5) 006406 000101  
(5) 006410 014246  
(5) 006412 015624  
3786 006414 005037 002352  
3787 006420 012637 002400  
3788 006424 012601  
3789 006426 000207  
3790  
3791  
3792  
3793  
3794  
3795
```

```
3$: CLR    SUBRPC       ;CLEAR SUBR CALL PC  
    MOV    (SP)+,REGNUM ;RESTORE LU REG NO.  
    MOV    (SP)+,R1     ;RESTORE R1  
    RTS    PC           ;RETURN
```

```
3796  
3797  
3798  
3799  
3800  
3801  
3802  
3803 006430 013746 002400
```

```
*****  
* ISIRDY - THIS SUBROUTINE CHECKS FOR THE PROPER STATES OF ICIR (REG 17)  
*   AND IRDY (REG 12) AND REPORTS AN ERROR IF EITHER IS NOT PROPERLY SET  
*   AS PASSED IN BIT 0 (ICIR) AND BIT 1 (IRDY) OF THE WORD FOLLOWING THE  
*   CALL.  
*   IF AN ERROR OCCURS, A RETURN IS MADE TO THE TEST AT THE ADDRESS  
*   IN RETADR.  
*****  
ISIRDY: MOV    REGNUM,-(SP) ;SAVE LU REG NO.
```

3804	006434	013746	002352			MOV	SUBRPC,-(SP)		
3805	006440	005737	002352			TST	SUBRPC	:SEE IF THIS IS A NESTED CALL	
3806	006444	001006				BNE	1\$:BR IF YES	
3807	006446	016637	000004	002352		MOV	4(SP),SUBRPC		
3808	006454	162737	000004	002352		SUB	#4,SUBRPC	:GET PC OF SUBR CALL	
3809	006462	012737	000012	002400	1\$:	MOV	#12,REGNUM	:SET REG NO. TO 12	
3810	006470	004737	003672			JSR	PC,READLU	:READ REG 12	
3811	006474	032776	000002	C00004		BIT	#BIT1,@4(SP)	:GET EXPECTED STATE OF IRDY	
3812	006502	001413				BEQ	3\$:BR IF EXPECTED IRDY = 0	
3813	006504	132737	000020	002364		BITB	#IRDY,REDBYT	:SEE IF IRDY = 1	
3814	006512	001022				BNE	9\$:BR IF IRDY = 1	
3815	006514	004737	004526			JSR	PC,GETALL	:GET REGS FOR PRINTOUT	
3816									
3817	006520					:REPORT	IRDY NOT SET		
(4)	006520	104455				ERRDF	17,EM17,ERR4		
(5)	006522	000021						TRAP	C\$ERDF
(5)	006524	012645						.WORD	17
(5)	006526	015624						.WORD	EM17
3818	006530	000451				BR	16\$:TAKE ERROR EXIT	
3819	006532	132737	000020	002364	3\$:	BITB	#IRDY,REDBYT	:SEE IF IRDY = 0	
3820	006540	001407				BEQ	9\$:BR IF IRDY = 0	
3821	006542	004737	004526			JSR	PC,GETALL	:GET REGS FOR PRINTOUT	
3822						:REPORT	IRDY NOT CLEARED		
3823	006546					ERRDF	18,EM18,ERR4		
(4)	006546	104455						TRAP	C\$ERDF
(5)	006550	000022						.WORD	18
(5)	006552	012662						.WORD	EM18
(5)	006554	015624						.WORD	ERR4
3824	006556	000436				BR	16\$:TAKE ERROR RETURN	
3825	006560	012737	000017	002400	9\$:	MOV	#17,REGNUM	:SET REG NO. = 17	
3826	006566	004737	003672			JSR	PC,READLU	:READ REG 17	
3827	006572	132776	000001	000004		BITB	#BIT0,@4(SP)	:GET EXPECTED STATE OF ICIR	
3828	006600	001413				BEQ	12\$:BR IF EXPECTED ICIR = 0	
3829	006602	132737	000010	002364		BITB	#ICIR,REDBYT	:SEE IF ICIR = 1	
3830	006610	001031				BNE	20\$:BR IF ICIR = 1	
3831	006612	004737	004526			JSR	PC,GETALL	:GET REGS FOR PRINTOUT	
3832						:REPORT	ICIR NOT SET		
3833	006616					ERRDF	19,EM19,ERR4		
(4)	006616	104455						TRAP	C\$ERDF
(5)	006620	000023						.WORD	19
(5)	006622	012703						.WORD	EM19
(5)	006624	015624						.WORD	ERR4
3834	006626	000412				BR	16\$:TAKE ERROR RETURN	
3835	006630	132737	000010	002364	12\$:	BITB	#ICIR,REDBYT	:SEE IF ICIR = 0	
3836	006636	001416				BEQ	20\$:BR IF ICIR = 0	
3837	006640	004737	004526			JSR	PC,GETALL	:GET REGS FOR PRINTOUT	
3838						:REPORT	ICIR NOT CLEARED		
3839	006644					ERRDF	20,EM20,ERR4		
(4)	006644	104455						TRAP	C\$ERDF
(5)	006646	000024						.WORD	20
(5)	006650	012720						.WORD	EM20
(5)	006652	015624						.WORD	ERR4
3840	006654	016637	000002	002400	16\$:	MOV	2(SP),REGNUM	:RESTORE LU REG NO.	
3841	006662	013706	002346			MOV	PSTACK,SP	:RESTORE STACK POINTER TO BASE LEVEL	
3842	006666	013746	002362			MOV	RETADR,-(SP)	:FIX ERROR RETURN PC	
3843	006672	000407				BR	23\$		

3844 006674 062766 000002 000004 20\$: ADD #2,4(SP) ;FIX UP ERROR-FREE RETURN PC
3845 006702 012637 002352 MOV (SP)+,SUBRPC
3846 006706 012637 002400 MOV (SP)+,REGNUM ;RESTORE LU REG NO.
3847 006712 000207 23\$: RTS PC ;RETURN

3848
3849
3850
3851
3852

3853 :*****
3854 :* IACTIV - THIS SUBROUTINE CHECKS FOR THE PROPER STATE OF IACT (REG 12) AND
3855 :* REPORTS AN ERROR IF IT IS NOT PROPERLY SET TO THE STATE OF BIT 0 IN THE
3856 :* WORD FOLLOWING THE CALL.
3857 :* IF AN ERROR OCCURS, A RETURN IS MADE TO THE TEST AT THE ADDRESS IN
3858 :* RETADR.
3859 :*****

3860 006714 013746 002400 IACTIV: MOV REGNUM,-(SP) ;SAVE LU REG NO.
3861 006720 013746 002352 MOV SUBRPC,-(SP)
3862 006724 005737 002352 TST SUBRPC ;SEE IF THIS IS A NESTED CALL
3863 006730 001006 BNE 1\$;BR IF YES
3864 006732 016637 000004 002352 MOV 4(SP),SUBRPC
3865 006740 162737 000004 002352 SUB #4,SUBRPC ;GET PC OF SUBR CALL
3866 006746 012737 000012 002400 1\$: MOV #12,REGNUM ;SET REG NO. = 12
3867 006754 004737 003672 JSR PC,READLU ;READ REG 12
3868 006760 032776 000001 000004 BIT #BIT0,4(SP) ;GET EXPECTED STATE OF IACT
3869 006766 001413 BEQ 3\$;BR IF EXPECTED IACT = 0
3870 006770 132737 000100 002364 BITB #IACT,REDBYT ;SEE IF IACT = 1
3871 006776 001031 BNE 9\$;BR IF IACT = 1
3872 007000 004737 004526 JSR PC,GETALL ;GET REGS FOR PRINTOUT
3873 :REPORT IACT NOT SET
3874 007004 ERRDF 21,EM21,ERR4

(4) 007004 104455 TRAP C\$ERDF
(5) 007006 000025 .WORD 21
(5) 007010 012741 .WORD EM21
(5) 007012 015624 .WORD ERR4

3875 007014 000412 BR 6\$;TAKE ERROR EXIT
3876 007016 132737 000100 002364 3\$: BITB #IACT,REDBYT ;SEE IF IACT = 0
3877 007024 001416 BEQ 9\$;BR IF IACT = 0
3878 007026 004737 004526 JSR PC,GETALL ;GET REGS FOR PRINTOUT
3879 :REPORT IACT NOT CLEARED
3880 ERRDF 22,EM22,ERR4

(4) 007032 104455 TRAP C\$ERDF
(5) 007034 000026 .WORD 22
(5) 007036 012756 .WORD EM22
(5) 007040 015624 .WORD ERR4

3881 007042 016637 000002 002400 6\$: MOV 2(SP),REGNUM ;RESTORE LU REG NO.
3882 007050 013706 002346 MOV PSTACK,SP ;RESTORE PROGRAM STACK TO BASE LEVEL
3883 007054 013746 002362 MOV RETADR,-(SP) ;FIX UP ERROR RETURN PC
3884 007060 000407 BR 12\$
3885 007062 062766 000002 000004 9\$: ADD #2,4(SP) ;FIX UP ERROR-FREE RETURN PC
3886 007070 012637 002352 MOV (SP)+,SUBRPC
3887 007074 012637 002400 MOV (SP)+,REGNUM ;RESTORE LU REG NO.
3888 007100 000207 12\$: RTS PC ;RETURN

3889
3890
3891

```
3892
3893
3894
3895
3896
3897
3898
3899
3900 007102 013746 002402
3901 007106 013746 002352
3902 007112 005737 002352
3903 007116 001006
3904 007120 016637 000004 002352
3905 007126 162737 000004 002352
3906 007134 012737 000001 002402 1$:
3907 007142 004737 004124
3908 007146 032776 000001 000004
3909 007154 001413
3910 007156 132737 000001 002372
3911 007164 001022
3912 007166 004737 004526
3913
3914 007172
(4) 007172 104455
(5) 007174 000035
(5) 007176 013175
(5) 007200 017014
3915 007202 000444
3916 007204 132737 000001 002372 3$:
3917 007212 001407
3918 007214 004737 004526
3919
3920 007220
(4) 007220 104455
(5) 007222 000034
(5) 007224 013154
(5) 007226 017014
3921 007230 000431
3922 007232 132776 000002 000004 9$:
3923 007240 001413
3924 007242 132737 000002 002372
3925 007250 001031
3926 007252 004737 004526
3927
3928 007256
(4) 007256 104455
(5) 007260 000037
(5) 007262 013233
(5) 007264 017014
3929 007266 000412
3930 007270 132737 000002 002372 12$:
3931 007276 001416
3932 007300 004737 004526
3933
3934 007304
(4) 007304 104455
```

: * RSEOM - THIS SUBROUTINE CHECKS FOR THE PROPER STATES OF RSOM AND REOM IN
: * AX0-16, AND REPORTS AN ERROR IF EITHER IS NOT SET TO THE STATE PASSED IN BITS
: * 0,1, RESPECTIVELY, OF THE WORD FOLLOWING THE CALL.
: * IF AN ERROR OCCURS, A RETURN IS MADE TO THE TEST AT THE ADDRESS IN RETADR.
: *****

RSEOM: MOV AXNUM,-(SP) ;SAVE AX BYTE NO.
MOV SUBRPC,-(SP)
TST SUBRPC ;SEE IF THIS IS A NESTED CALL
BNE 1\$;BR IF YES
MOV 4(SP),SUBRPC
SUB #4,SUBRPC ;GET PC OF SUBR CALL
1\$: MOV #1,AXNUM ;SET AX BYTE NO. FOR AX0-16
JSR PC,READAX ;READ AX0
BIT #BIT0,@4(SP) ;GET EXPECTED STATE OF RSOM
BEQ 3\$;BR IF EXPECTED RSOM = 0
BITB #RSOM,RAX16 ;SEE IF RSOM = 1
BNE 9\$;BR IF RSOM = 1
JSR PC,GETALL ;GET REGS FOR PRINTOUT
:REPORT RSOM NOT SET
ERRDF 29,EM29,ERR6

TRAP C\$ERDF
.WORD 29
.WORD EM29
.WORD ERR6

BR 16\$;TAKE ERROR EXIT
3\$: BITB #RSOM,RAX16 ;SEE IF RSOM = 0
BEQ 9\$;BR IF RSOM = 0
JSR PC,GETALL ;GET REGS FOR PRINTOUT
:REPORT RSOM NOT CLEARED
ERRDF 28,EM28,ERR6

TRAP C\$ERDF
.WORD 28
.WORD EM28
.WORD ERR6

BR 16\$;TAKE ERROR RETURN
9\$: BITB #BIT1,@4(SP) ;GET EXPECTED STATE OF REOM
BEQ 12\$;BR IF EXPECTED REOM = 0
BITB #REOM,RAX16 ;SEE IF REOM = 1
BNE 20\$;BR IF REOM = 1
JSR PC,GETALL ;GET REGS FOR PRINTOUT
:REPORT REOM NOT SET
ERRDF 31,EM31,ERR6

TRAP C\$ERDF
.WORD 31
.WORD EM31
.WORD ERR6

BR 16\$;TAKE ERROR RETURN
12\$: BITB #REOM,RAX16 ;SEE IF REOM = 0
BEQ 20\$;BR IF REOM = 0
JSR PC,GETALL ;GET REGS FOR PRINTOUT
:REPORT REOM NOT CLEARED
ERRDF 30,EM30,ERR6

TRAP C\$ERDF

.WORD 30
.WORD EM30
.WORD ERR6

(5) 007306 000036
(5) 007310 013212
(5) 007312 017014
3935 007314 016637 000002 002402 16\$:
3936 007322 013706 002346
3937 007326 013746 002362
3938 007332 000407
3939 007334 062766 000002 000004 20\$:
3940 007342 012637 002352
3941 007346 012637 002402
3942 007352 000207 23\$:
3943
3944
3945
3946
3947
3948
3949
3950
3951
3952 007354 013746 002400
3953 007360 012737 000012 002400
3954 007366 004737 003672
3955 007372 113737 002364 002425
3956 007400 042737 170000 002424
3957 007406 012737 000010 002400
3958 007414 004737 003672
3959 007420 113737 002364 002424
3960 007426 012637 002400
3961 007432 000207
3962
3963
3964
3965
3966
3967
3968
3969
3970
3971
3972
3973
3974
3975
3976 007434 010146
3977 007436 010346
3978 007440 013746 002400
3979 007444 016637 000006 002352
3980 007452 162737 000004 002352
3981 007460 012737 000012 002400
3982 007466 005001
3983 007470 017603 000006
3984 007474 062703 000003
3985 007500 005776 000006
3986 007504 001414
3987 007506 004737 006714

* RDRXSI - THIS SUBROUTINE READS THE RCV SILO (REGS 10,12) AND RETURNS THE
* SILO ENTRY IN BITS 0-11 OF RXWORD.

RDRXSI: MOV REGNUM,-(SP) ;SAVE LU REG NO.
MOV #12,REGNUM ;SET REG NO. = 12
JSR PC,READLU ;READ LU REG 12
MOVB REDBYT,RXWORD+1 ;GET HI BITS OF SILO ENTRY
BIC #170000,RXWORD ;CLEAR UNUSED BITS
MOV #10,REGNUM ;SET REG NO. = 10
JSR PC,READLU ;READ REG 10
MOVB REDBYT,RXWORD ;GET LOW BITS OF SILO ENTRY
MOV (SP)+,REGNUM ;RESTORE LU REG NO.
RTS PC ;RETURN

* RCV1ST - THIS SUBROUTINE RECEIVES THE FIRST CHAR OF A MESSAGE, AND MONITORS
* STATUS OF THE RECEIVER. FIRST, A CHECK IS MADE FOR IACT = 0, IRDY = 0,
* ICIR = 1, AND RSOM = 0. THEN, THE LINE UNIT IS CLOCKED USING
* STEPLU UNTIL IRDY = 1. THE PROGRAM CHECKS FOR THIS TO OCCUR WITHIN 3
* CYCLES AFTER THE NO. OF CYCLES PASSED IN THE WORD FOLLOWING THE CALL.
* IF AN ERROR OCCURS, A RETURN IS MADE TO THE TEST, AT THE ADDRESS
* CONTAINED IN RETADR.

RCV1ST: MOV R1,-(SP) ;SAVE R1
MOV R3,-(SP) ;SAVE R3
MOV REGNUM,-(SP) ;SAVE LU REG NO.
MOV 6(SP),SUBRPC
SUB #4,SUBRPC ;GET PC OF SUBROUTINE CALL
MOV #12,REGNUM ;SET LU REG NO. = 12
CLR R1 ;INIT CYCLE COUNT TO 0
MOV @6(SP),R3 ;GET CYCLE COUNT LIMIT
ADD #3,R3
TST @6(SP) ;SEE IF DESIRED CYCLES = 0
BEQ 8\$;BR IF YES
JSR PC,IACTIV ;CHK FOR IACT = 0

```

3988 007512 000000 0
3989 007514 004737 006430 JSR PC,ISIRDY ;CHK FOR ICIR = 1, IRDY = 0
3990 007520 000001 1
3991 007522 004737 007102 JSR PC,RSEOM ;CHK RSOM = 0, REOM = 0 IN AX0-16
3992 007526 000000 0
3993 007530 004737 005254 6$: JSR PC,STPLU ;CLOCK LU FOR 1 CYCLE
3994 007534 000001 1
3995 007536 004737 005146 8$: JSR PC,WAIT50 ;ALLOW SILO DATA TO RIPPLE
3996 007542 005201 INC R1 ;INCREMENT CYCLE COUNT
3997 007544 004737 003672 JSR PC,READLU ;READ REG 12
3998 007550 132737 000020 002364 BITB #IRDY,REDBYT ;SEE IF IRDY = 1 YET
3999 007556 001005 BNE 9$ ;BR IF IRDY = 1
4000 007560 020103 CMP R1,R3 ;SEE IF LIMIT EXCEEDED
4001 007562 002762 BLT 6$ ;BR IF NOT YET
4002 007564 004737 006430 JSR PC,ISIRDY ;CHK FOR ICIR = 1, IRDY = 1
4003 007570 000003 3
4004 007572 020176 000006 9$: CMP R1,@6(SP) ;SEE IF LESS THAN REQUIRED CYCLES
4005 007576 002003 BGE 12$ ;BR IF NOT
4006 007600 004737 006430 JSR PC,ISIRDY ;CHK FOR ICIR = 1, IRDY = 0
4007 007604 000001 1
4008 007606 004737 006714 12$: JSR PC,IACTIV ;CHK FOR IACT = 1
4009 007612 000001 1
4010 007614 004737 006430 JSR PC,ISIRDY ;CHK FOR ICIR = 1, IRDY = 1
4011 007620 000003 3
4012 007622 062766 000002 000006 ADD #2,6(SP) ;FIX UP RETURN PC
4013 007630 012637 002400 MOV (SP)+,REGNUM ;RESTORE LU REG NO.
4014 007634 012603 MOV (SP)+,R3 ;RESTORE R3
4015 007636 012601 MOV (SP)+,R1 ;RESTORE R1
4016 007640 005037 002352 CLR SUBRPC ;CLEAR SUBR CALL PC
4017 007644 000207 RTS PC ;RETURN

```

4018
4019
4020
4021
4022
4023
4024
4025
4026
4027
4028
4029
4030
4031
4032
4033
4034
4035
4036
4037
4038
4039
4040
4041
4042
4043

```

:*****
:* STPERR - THIS SUBROUTINE LOADS THE CONTENTS OF THE FIRST WORD FOLLOWING THE
:* CALL INTO REG 17, AND SETS THE IERR BIT, AND CLOCKS THE LINE UNIT
:* FOR THE NO. OF CYCLES PASSED IN THE 2ND WORD FOLLOWING THE CALL. THEN,
:* IT RESTORES REG 17 TO ITS ORIGINAL CONTENTS, CLEARING THE IERR BIT.
:*****
STPERR: MOV REGNUM,-(SP) ;SAVE LU REG NO.
MOV #17,REGNUM ;SET LU REG NO. = 17
MOV @2(SP),WRIBYT
BISB #IERR,WRIBYT
JSR PC,WRITLU ;SET IERR BIT IN REG 17
ADD #2,2(SP) ;INCREMENT SUBR ARGUMENT POINTER
MOV @2(SP),3$ ;GET DESIRED NO. OF CYCLES
JSR PC,STPLU ;CLOCK LU FOR DESIRED NO. OF CYCLES
3$: .WORD 0 ;NO. OF CYCLES GOES HERE
BICB #IERR,WRIBYT
JSR PC,WRITLU ;CLEAR IERR BIT IN REG 17
ADD #2,2(SP) ;FIX UP RETURN PC
MOV (SP)+,REGNUM ;RESTORE LU REG NO.
RTS PC ;RETURN

```

4044
 4045
 4046
 4047
 4048
 4049
 4050
 4051
 4052
 4053
 4054
 4055
 4056
 4057 007750 010146
 4058 007752 013746 002400
 4059 007756 016637 000004 002352
 4060 007764 162737 000004 002352
 4061 007772 017601 000004
 4062 007776 042701 170000
 4063 010002 004737 007354
 4064 010006 023727 002432 000347
 4065 010014 001005
 4066 010016 042701 000200
 4067 010022 042737 000200 002424
 4068 010030 120137 002424
 4069 010034 001445
 4070 010036 005037 002404
 4071 010042 110137 002404
 4072 010046 005037 002406
 4073 010052 113737 002424 002406
 4074 010060 012737 000011 002400
 4075 010066 004737 003672
 4076 010072 132737 000001 002364
 4077 010100 001410
 4078 010102 004737 004526
 4079
 4080 010106
 (4) 010106 104455
 (5) 010110 000066
 (5) 010112 014210
 (5) 010114 015624
 4081 010116 000137 010600
 4082 010122 012737 000010 002400
 4083 010130 004737 004526
 4084
 4085 010134
 (4) 010134 104455
 (5) 010136 000042
 (5) 010140 013322
 (5) 010142 020124
 4086 010144 000137 010600
 4087 010150 000301
 4088 010152 012737 000012 002400
 4089 010160 120137 002425
 4090 010164 001002
 4091 010166 000137 010554

```

*****
* CKDATA - THIS SUBROUTINE READS THE RCV SILO AND COMPARES THE SILO ENTRY
* TO BITS 0-11 OF THE FIRST WORD FOLLOWING THE CALL. IF THERE IS A
* MISMATCH, THE ERROR IS REPORTED AND A RETURN IS MADE TO THE TEST AT THE
* ADDRESS CONTAINED IN RETADR. IF BIT 15 = 0 IN THE FIRST WORD
* FOLLOWING THE CALL, THE SUBROUTINE WILL NOT CHECK THE BCC BIT (SILO
* BIT 8). IF THERE ARE NO ERRORS, THE LINE UNIT IS CLOCKED FOR THE
* NUMBER OF CYCLES PASSED IN THE SECOND WORD FOLLOWING THE CALL.
*****
CKDATA: MOV R1, -(SP) ;SAVE R1
MOV REGNUM, -(SP) ;SAVE LU REG NO.
MOV 4(SP), SUBRPC
SUB #4, SUBRPC ;GET PC OF SUBR CALL
MOV @4(SP), R1 ;GET EXPECTED SILO ENTRY
BIC #170000, R1 ;CLEAR UNUSED BITS FOR COMPARE
JSR PC, RDRXSI ;READ RCV SILO
CMP SAVLEN, #TXLEN2!TXLEN1!TXLEN0!RXLEN2!RXLEN1!RXLEN0
BNE 4$ ;BR IF CHAR LENGTH NOT = 7
BIC #BIT7, R1 ;MASK OFF BIT 8TH BIT
BIC #BIT7, RXWORD
4$: CMPB R1, RXWORD ;COMPARE EXPECTED BITS 0-7 TO ACTUAL
BEQ 6$ ;BR IF MATCH
CLR GOODAT
MOV B, R1, GOODAT ;GET EXPECTED DATA
CLR B, BADDAT
MOV B, RXWORD, BADDAT ;GET ACTUAL DATA
MOV #11, REGNUM ;SET REG NO. = 11
JSR PC, READLU ;READ REG 11
BITB #UNRR, REDBYT ;SEE IF TX UNDERRUN ERROR
BEQ 5$ ;BR IF NOT
JSR PC, GETALL ;GET REGS FOR PRINTOUT
;REPORT TX UNDERRUN ERROR
ERRDF 54, EM54, ERR4

TRAP C$ERDF
.WORD 54
.WORD EM54
.WORD ERR4

5$: JMP 36$ ;TAKE ERROR EXIT
MOV #10, REGNUM ;SET REG NO. = 10
JSR PC, GETALL ;GET REGS FOR PRINTOUT
;REPORT RCV'D DATA MISCOMPARE
ERRDF 34, EM34, ERR8

TRAP C$ERDF
.WORD 34
.WORD EM34
.WORD ERR8

6$: JMP 36$ ;TAKE ERROR EXIT
SWAB R1
MOV #12, REGNUM ;SET LU REG NO. FOR ERROR REPORTS
CMPB R1, RXWORD+1 ;COMPARE EXPECTED SILO BITS 8-11 TO ACTUAL
BNE 7$ ;BR IF MISMATCH
JMP 22$ ;CONTINUE

```

4092	010172	005037	002404		7\$:	CLR	GOODAT					
4093	010176	110137	002404			MOVB	R1,GOODAT		;SET EXPECTED DATA			
4094	010202	005037	002406			CLR	BADDAT					
4095	010206	113737	002425	002406		MOVB	RXWORD+1,BADDAT		;SET ACTUAL DATA			
4096	010214	032776	100000	000004		BIT	#BCCCHK,24(SP)		;SEE IF BCC SHOULD BE IGNORED			
4097	010222	001433				BEQ	10\$;BR IF YES			
4098	010224	132701	000001			BITB	#BCC,R1		;SEE IF EXPECTED BIT = 1			
4099	010230	001014				BNE	8\$;BR IF YES			
4100	010232	132737	000001	002425		BITB	#BCC,RXWORD+1		;SEE IF ACTUAL BIT = 0			
4101	010240	001424				BEQ	10\$;BR IF YES			
4102	010242	004737	004526			JSR	PC,GETALL		;GET REGS FOR PRINTOUT			
4103						:REPORT	BCC NOT CLEARED					
4104	010246					ERRDF	35,EM35,ERR8					
(4)	010246	104455								TRAP	C\$ERDF	
(5)	010250	000043								.WORD	35	
(5)	010252	013350								.WORD	EM35	
(5)	010254	020124								.WORD	ERR8	
4105	010256	000137	010600			JMP	36\$;TAKE ERROR EXIT			
4106	010262	132737	000001	002425	8\$:	BITB	#BCC,RXWORD+1		;SEE IF ACTUAL BIT = 1			
4107	010270	001010				BNE	10\$;BR IF YES			
4108	010272	004737	004526			JSR	PC,GETALL		;GET REGS FOR PRINTOUT			
4109						:REPORT	BCC NOT SET					
4110	010276					ERRDF	36,EM36,ERR8					
(4)	010276	104455								TRAP	C\$ERDF	
(5)	010300	000044								.WORD	36	
(5)	010302	013370								.WORD	EM36	
(5)	010304	020124								.WORD	ERR8	
4111	010306	000137	010600			JMP	36\$;TAKE ERROR EXIT			
4112	010312				10\$:							
4113	010312	132701	000002			BITB	#EBLK,R1		;SEE IF EXPECTED BIT = 1			
4114	010316	001014				BNE	12\$;BR IF YES			
4115	010320	132737	000002	002425		BITB	#EBLK,RXWORD+1		;SEE IF ACTUAL BIT = 0			
4116	010326	001424				BEQ	14\$;BR IF YES			
4117	010330	004737	004526			JSR	PC,GETALL		;GET REGS FOR PRINTOUT			
4118						:REPORT	EBLK NOT CLEARED					
4119	010334					ERRDF	37,EM37,ERR8					
(4)	010334	104455								TRAP	C\$ERDF	
(5)	010336	000045								.WORD	37	
(5)	010340	013404								.WORD	EM37	
(5)	010342	020124								.WORD	ERR8	
4120	010344	000137	010600			JMP	36\$;TAKE ERROR EXIT			
4121	010350	132737	000002	002425	12\$:	BITB	#EBLK,RXWORD+1		;SEE IF ACTUAL BIT = 1			
4122	010356	001010				BNE	14\$;BR IF YES			
4123	010360	004737	004526			JSR	PC,GETALL		;GET REGS FOR PRINTOUT			
4124						:REPORT	EBLK NOT SET					
4125	010364					ERRDF	38,EM38,ERR8					
(4)	010364	104455								TRAP	C\$ERDF	
(5)	010366	000046								.WORD	38	
(5)	010370	013425								.WORD	EM38	
(5)	010372	020124								.WORD	ERR8	
4126	010374	000137	010600			JMP	36\$;TAKE ERROR EXIT			
4127	010400				14\$:							
4128												
4129	010400	132701	000004			BITB	#RAB,R1		;SEE IF EXPECTED BIT = 1			
4130	010404	001014				BNE	16\$;BR IF YES			
4131	010406	132737	000004	002425		BITB	#RAB,RXWORD+1		;SEE IF ACTUAL BIT = 0			

4132	010414	001424				BEQ	18\$:BR IF YES		
4133	010416	004737	004526			JSR	PC,GETALL		:GET REGS FOR PRINTOUT		
4134						:REPORT	RAB NOT	CLEARED			
4135	010422					ERRDF	39,EM39,ERR8				
(4)	010422	104455								TRAP	C\$ERDF
(5)	010424	000047								.WORD	39
(5)	010426	013442								.WORD	EM39
(5)	010430	020124								.WORD	ERR8
4136	010432	000137	010600			JMP	36\$:TAKE ERROR EXIT		
4137	010436	132737	000004	002425	16\$:	BITB	#RAB,RXWORD+1		:SEE IF ACTUAL BIT = 1		
4138	010444	001010				BNE	18\$:BR IF YES		
4139	010446	004737	004526			JSR	PC,GETALL		:GET REGS FOR PRINTOUT		
4140						:REPORT	RAB NOT	SET			
4141	010452					ERRDF	40,EM40,ERR8				
(4)	010452	104455								TRAP	C\$ERDF
(5)	010454	000050								.WORD	40
(5)	010456	013462								.WORD	EM40
(5)	010460	020124								.WORD	ERR8
4142	010462	000137	010600			JMP	36\$:TAKE ERROR EXIT		
4143	010466				18\$:						
4144	010466	132701	000010			BITB	#OVRR,R1		:SEE IF EXPECTED BIT = 1		
4145	010472	001014				BNE	20\$:BR IF YES		
4146	010474	132737	000010	002425		BITB	#OVRR,RXWORD+1		:SEE IF ACTUAL BIT = 0		
4147	010502	001424				BEQ	22\$:BR IF YES		
4148	010504	004737	004526			JSR	PC,GETALL		:GET REGS FOR PRINTOUT		
4149						:REPORT	OVRR NOT	CLEARED			
4150	010510					ERRDF	41,EM41,ERR8				
(4)	010510	104455								TRAP	C\$ERDF
(5)	010512	000051								.WORD	41
(5)	010514	013476								.WORD	EM41
(5)	010516	020124								.WORD	ERR8
4151	010520	000137	010600			JMP	36\$:TAKE ERROR EXIT		
4152	010524	132737	000010	002425	20\$:	BITB	#OVRR,RXWORD+1		:SEE IF ACTUAL BIT = 1		
4153	010532	001010				BNE	22\$:BR IF YES		
4154	010534	004737	004526			JSR	PC,GETALL		:GET REGS FOR PRINTOUT		
4155						:REPORT	OVRR NOT	SET			
4156	010540					ERRDF	42,EM42,ERR8				
(4)	010540	104455								TRAP	C\$ERDF
(5)	010542	000052								.WORD	42
(5)	010544	013517								.WORD	EM42
(5)	010546	020124								.WORD	ERR8
4157	010550	000137	010600			JMP	36\$:TAKE ERROR EXIT		
4158	010554				22\$:						
4159	010554	062766	000002	000004		ADD	#2,4(SP)		:INCR SUBROUTINE ARGUMENT POINTER		
4160	010562	017637	000004	010574		MOV	@4(SP),24\$:GET DESIRED CYCLE COUNT		
4161	010570	004737	005254			JSR	PC,STPLU		:CLOCK LU FOR DESIRED CYCLES		
4162	010574	000000			24\$:	.WORD	0				
4163	010576	000407				BR	38\$:TAKE ERROR-FREE EXIT		
4164	010600	011637	002400		36\$:	MOV	(SP),REGNUM		:RESTORE LU REG NO.		
4165	010604	013706	002346			MOV	PSTACK,SP		:RESTORE PROGRAM STACK TO BASE LEVEL		
4166	010610	013746	002362			MOV	RETADR,-(SP)		:FIX UP ERROR RETURN PC		
4167	010614	000406				BR	40\$				
4168	010616	062766	000002	000004	38\$:	ADD	#2,4(SP)		:FIX UP ERROR-FREE RETURN PC		
4169	010624	012637	002400			MOV	(SP)+,REGNUM		:RESTORE LU REG NO.		
4170	010630	012601				MOV	(SP)+,R1		:RESTORE R1		
4171	010632	005037	002352		40\$:	CLR	SUBRPC		:CLEAR SUBROUTINE PC		

4172 010636 000207 RTS PC ;RETURN
4173
4174
4175
4176
4177
4178

```
*****  
;* SETUP - THIS SUBROUTINE LOADS THE FIRST WORD AFTER THE CALL INTO AX2-15  
;* (SYNCH CHAR), LOADS THE SECOND WORD AFTER THE CALL INTO REG 17  
;* LOADS THE THIRD WORD INTO AX3-15, AND LOADS THE FOURTH INTO AX3-16.  
*****
```

4183 010640 013746 002402
4184 010644 013746 002400
4185 010650 012737 000004 002402
4186 010656 017637 000004 002374
4187 010664 005037 002376
4188 010670 004737 004312
4189 010674 012737 000017 002400
4190 010702 062766 000002 000004
4191 010710 017637 000004 002366

```
SETUP:  MOV     AXNUM,-(SP)         ;SAVE AX BYTE NO.  
        MOV     REGNUM,-(SP)       ;SAVE LU REG NO.  
        MOV     #4,AXNUM           ;SET AX BYTE NO. FOR AX2  
        MOV     @4(SP),WAX15  
        CLR     WAX16  
        JSR     PC,WRITAX          ;SET SYNCH CHAR IN AX2-15, CLEAR AX2-16  
        MOV     #17,REGNUM         ;SET LU REG NO. = 17  
        ADD     #2,4(SP)           ;INCREMENT ARGUMENT POINTER  
        MOV     @4(SP),WRIBYT
```

4193	010716	004737	003750		JSR	PC,WRITLU	:LOAD REG 17
4194	010722	012737	000006	002402	MOV	#6,AXNUM	:SET AX BYTE NO. FOR AX3
4195	010730	062766	000002	000004	ADD	#2,4(SP)	:INCREMENT ARGUMENT POINTER
4196	010736	017637	000004	002374	MOV	@4(SP),WAX15	
4197	010744	062766	000002	000004	ADD	#2,4(SP)	:INCR ARGUMENT POINTER
4198	010752	017637	000004	002376	MOV	@4(SP),WAX16	
4199	010760	013737	002376	002432	MOV	WAX16,SAVLEN	:STORE TX AND RCV CHAR LENGTH
4200	010766	004737	004312		JSR	PC,WRITAX	:LOAD AX3-15, AX3-16
4201	010772	062766	000002	000004	ADD	#2,4(SP)	:FIX RETURN PC
4202	011000	012637	002400		MOV	(SP)+,REGNUM	:RESTORE LU REG NO.
4203	011004	012637	002402		MOV	(SP)+,AXNUM	:RESTORE AX BYTE NO.
4204	011010	005037	002352		CLR	SUBRPC	:CLEAR SUBROUTINE PC STORAGE
4205	011014	000207			RTS	PC	:RETURN

4206
 4207
 4208
 4209
 4210
 4211
 4212
 4213
 4214
 4215
 4216
 4217
 4218
 4219
 4220
 4221
 4222
 4223
 4224
 4225
 4226
 4227
 4228
 4229

 : * LODMSG - THIS SUBROUTINE LOADS THE NO. OF WORDS PASSED IN THE SECOND WORD
 : * FOLLOWING THE CALL FROM THE MSG BUFFER WHOSE ADDRESS IS IN THE FIRST
 : * WORD FOLLOWING THE CALL, INTO THE TRANSMITTER SILO.

LODMSG:	MOV	R1,-(SP)	:SAVE R1
	MOV	R2,-(SP)	:SAVE R2
	MOV	@4(SP),R1	:GET MSG POINTER INTO R1
	ADD	#2,4(SP)	:INCR ARG POINTER
	MOV	@4(SP),R2	:GET WORD COUNT INTO R2
	ADD	#2,4(SP)	:FIX UP RETURN PC
6\$:	MOV	(R1)+,TXWORD	:GET NEXT MSG WORD
	JSR	PC,LDTXSI	:LOAD A WORD INTO TX SILO
	DEC	R2	:DECR COUNT
	BNE	6\$:BR IF NOT DONE YET
	JSR	PC,WAIT50	:WAIT FOR SILO TO RIPPLE
	MOV	(SP)+,R2	:RESTORE R2
	MOV	(SP)+,R1	:RESTORE R1
	RTS	PC	:RETURN

4230
 4231
 4232
 4233
 4234
 4235
 4236
 4237
 4238
 4239
 4240
 4241

 : * RXCHAR - THIS SUBROUTINE READS THE RCV SILO AND CLOCKS THE LINE UNIT.
 : * FIRST, IT READS THE CHAR FROM THE RCV SILO, AND CHECKS FOR ICIR
 : * = 1, IRDY = 0. IT THEN CLOCKS THE LINE UNIT FOR THE NO. OF CYCLES
 : * PASSED IN THE WORD FOLLOWING THE CALL, AND THEN CHECKS FOR ICIR
 : * = 1, IRDY = 1.

4242	011074	010146			RXCHAR:	MOV	R1,-(SP)	:SAVE R1
4243	011076	016637	000002	002352		MOV	2(SP),SUBRPC	
4244	011104	162737	000004	002352		SUB	#4,SUBRPC	:GET PC OF SUBR CALL
4245	011112	004737	007354			JSR	PC,RDRXSI	:READ RCV SILO
4246	011116	004737	005146			JSR	PC,WAIT50	:ALLOW SILO TO RIPPLE
4247	011122	005001				CLR	R1	:INIT CYCLE COUNT
4248	011124	020176	000002		9\$:	CMP	R1,@2(SP)	:SEE IF REQUIRED CYCLES DONE YET

```
4249 011130 001410          BEQ    12$          ;BR IF YES
4250 011132 004737 006430    JSR    PC,ISIRDY   ;CHK ICIR = 1, IRDY = 0
4251 011136 000001          1
4252 011140 004737 005254    JSR    PC,STPLU    ;CLK LU 1 CYCLE
4253 011144 000001          1
4254 011146 005201          INC    R1           ;INCR CYCLE COUNT
4255 011150 000765          BR     9$
4256 011152 004737 006430    12$: JSR    PC,ISIRDY   ;CHK ICIR = 1 IRDY = 1
4257 011156 000003          3
4258 011160 062766 000002 000002 ADD    #2,2(SP)     ;FIX RETURN PC
4259 011166 005037 002352    CLR    SUBRPC      ;CLEAR SUBR CALL PC
4260 011172 012601          MOV    (SP)+,R1    ;RESTORE R1
4261 011174 000207          RTS    PC           ;RETURN
4262
4263
4264
4265
4266
4267
4268 ;*****
4269 ;* CKTBIT - THIS SUBROUTINE CLOCKS THE LINE UNIT FOR 8 CYCLES, AND AFTER EACH
4270 ;* CYCLE, THE TXDATA BIT IN REG 17 IS EXAMINED, AND COMPARED TO A BIT OF
4271 ;* THE CHAR PASSED IN THE WORD FOLLOWING THE CALL.
4272 ;* IF AN ERROR OCCURS, A RETURN IS MADE TO THE TEST AT THE ADDRESS IN
4273 ;* RETADR.
4274 ;*****
4274 011176 013746 002400    CKTBIT: MOV    REGNUM,-(SP) ;SAVE LU REG NO.
4275 011202 010146          MOV    R1,-(SP)    ;SAVE R1
4276 011204 016637 000004 002352    MOV    4(SP),SUBRPC
4277 011212 162737 000004 002352    SUB    #4,SUBRPC   ;GET PC OF SUBROUTINE CALL
4278 011220 012701 000001          MOV    #BIT0,R1    ;INIT BIT POINTER
4279 011224 012737 000017 002400    MOV    #17,REGNUM  ;SET REG NO. = 17
4280 011232 004737 005254    4$: JSR    PC,STPLU   ;CLOCK LINE UNIT 1 CYCLE
4281 011236 000001          1
4282 011240 004737 003672    JSR    PC,READLU   ;READ REG 17
4283 011244 030176 000004          BIT    R1,#4(SP)   ;SEE IF EXPECTED BIT = 1
4284 011250 001013          BNE    9$          ;BR IF YES
4285 011252 132737 000040 002364    BITB   #TXDATA,REDBYT ;SEE IF ACTUAL BIT = 0
4286 011260 001422          BEQ    12$         ;BR IF YES
4287 011262 004737 004526    JSR    PC,GETALL   ;GET REGS FOR PRINTOUT
4288 ;REPORT TXDATA NOT CLEARED
4289 011266          ERRDF 32,EM32,ERR4
(4) 011266 104455          TRAP   C$ERDF
(5) 011270 000040          .WORD 32
(5) 011272 013250          .WORD EM32
(5) 011274 015624          .WORD ERR4
4290 011276 000420          BR     20$         ;TAKE ERROR RETURN
4291 011300 132737 000040 002364    9$: BITB   #TXDATA,REDBYT ;SEE IF ACTUAL BIT = 1
4292 011306 001007          BNE    12$         ;BR IF YES
4293 011310 004737 004526    JSR    PC,GETALL   ;GET REGS FOR PRINTOUT
4294 ;REPORT TXDATA BIT NOT SET
4295 011314          ERRDF 33,EM33,ERR4
(4) 011314 104455          TRAP   C$ERDF
(5) 011316 000041          .WORD 33
(5) 011320 013277          .WORD EM33
(5) 011322 015624          .WORD ERR4
4296 011324 000405          BR     20$         ;TAKE ERROR EXIT
```

```
4297 011326 006301      12$: ASL    R1          ;SHIFT BIT POINTER
4298 011330 020127 000400  CMP    R1,#400       ;SEE IF 8 BITS SCANNED YET
4299 011334 001336      BNE    4$            ;BR IF NO
4300 011336 000405      BR     22$          ;
4301 011340 013706 002346 20$: MOV    PSTACK,SP    ;RESTORE PROGRAM STACK POINTER TO BASE LEVEL
4302 011344 013746 002362  MOV    RETADR,-(SP)  ;FIX UP RETURN PC
4303 011350 000406      BR     40$          ;
4304 011352 062766 000002 000004 22$: ADD    #2,4(SP)      ;FIX UP ERROR-FREE RETURN PC
4305 011360 012601      MOV    (SP)+,R1      ;RESTORE R1
4306 011362 012637 002400  MOV    (SP)+,REGNUM  ;RESTORE REG NO.
4307 011366 005037 002352 40$: CLR    SUBRPC      ;CLEAR SUBR CALL PC
4308 011372 000207      RTS    PC           ;RETURN
```

4309
4310
4311
4312
4313
4314

```
*****  
* LDMSG1 - THIS SUBROUTINE LOADS THE TRANSMITTER SILO WITH MSG1, AND LOADS  
* THE DATA CHARS INTO THE RCV MSG BUFFER (RCVBUF:), AS EXPECTED DATA  
* FOR LATER COMPARISON.  
*****
```

```
4315  
4316  
4317  
4318  
4319 011374 010 46  
4320 011376 010246  
4321 011400 004737 011016  
4322 011404 003220  
4323 011406 000011  
4324 011410 012701 003224  
4325 011414 012702 003262  
4326 011420 012122  
4327 011422 020127 003236  
4328 011426 103774  
4329 011430 052762 100400 177776  
4330 011436 012726 000160  
4331 011442 012726 000034  
4332 011446 012602  
4333 011450 012601  
4334 011452 000207  
4335  
4336  
4337  
4338  
4339
```

```
LDMSG1: MOV    R1,-(SP)    ;SAVE R1
        MOV    R2,-(SP)    ;SAVE R2
        JSR    PC,LODMSG   ;LOAD MSG1 INTO TX SILO
        MSG1
        9.
        MOV    #MSG1+4,R1  ;GET POINTER TO MSG1
        MOV    #RCVBUF,R2  ;GET POINTER TO MSG BUF
3$:     MOV    (R1)+,(R2)+  ;LOAD A CHAR INTO MSG BUF
        CMP    R1,#MSG1+14. ;SEE IF DID LAST DATA CHAR YET
        BLO   3$          ;BR IF NOT
        BIS    #CRCCHK!RXBCC,-2(R2) ;SET EXPECTED BCC
        MOV    #160,(SP)+  ;LOAD HI CRC BYTE
        MOV    #034,(SP)+  ;LOAD LO CRC BYTE
        MOV    (SP)+,R2    ;RESTORE R2
        MOV    (SP)+,R1    ;RESTORE R1
        RTS    PC         ;RETURN
```

4341
4342
4343
4344
4345
4346
4347
4348

.SBTTL GLOBAL ERROR REPORT SECTION

:/
:/ THE GLOBAL ERROR REPORT SECTION CONTAINS ERROR MESSAGES
:/ THAT ARE USED IN MORE THAN ONE TEST.
:/

4349	011454	052045	047445	022466	FMT1:	.ASCIZ	/XT%06%N/
	011462	000116					
4350	011464	047045	040445	040506	FMT2:	.ASCIZ	/N%AFAILING REG: /
	011472	046111	047111	020107			
	011500	042522	035107	000040			
4351	011506	040445	054105	042520	FMT3:	.ASCIZ	/AEXPECTED: %03%S5%AACTUAL: %03%N/
	011514	052103	042105	020072			
	011522	047445	022463	032523			
	011530	040445	041501	052524			
	011536	046101	020072	047445			
	011544	022463	000116				
4352	011550	047045	052045	047045	FMT4:	.ASCIZ	/N%T%N%T%N/
	011556	052045	047045	000			
4353	011563	045	031517	051445	FMT5:	.ASCIZ	/03%S5%03%S5%03%S5%03%N/
	011570	022465	031517	051445			
	011576	022465	031517	051445			
	011604	022465	031517	047045			
	011612	000					
4354	011613	045	032123	047445	FMT6:	.ASCIZ	/S4%03%S5%03%S5%03%S5%03%N/
	011620	022463	032523	047445			
	011626	022463	032523	047445			
	011634	022463	032523	047445			
	011642	022463	000116				
4355	011646	052045	047445	022462	FMT7:	.ASCIZ	/T%02%N/
	011654	000116					
4356	011656	040445	054105	042524	FMT8:	.ASCIZ	/AEXTENDED REG AX%01%A-T%N/
	011664	042116	042105	051040			
	011672	043505	040440	022530			
	011700	030517	040445	022455			
	011706	022524	000116				
4357	011712	052045	047045	000	FMT9:	.ASCIZ	/T%N/
4358	011717	045	050101	020103	FMT10:	.ASCIZ	/APC OF SUBR CALL: %06%N/
	011724	043117	051440	041125			
	011732	020122	040503	046114			
	011740	020072	047445	022466			
	011746	000116					
4359	011750	040445	042522	020107	FMT11:	.ASCIZ	/AREG %02%A LOADED WITH: %03%N/
	011756	047445	022462	020101			
	011764	047514	042101	042105			
	011772	053440	052111	035110			
	012000	022440	031517	047045			
	012006	000					
4360	012007	045	022516	052101	FMT19:	.ASCIZ	/N%ATEST %D2%A NOT RUN%N/
	012014	051505	020124	042045			
	012022	022462	020101	047516			
	012030	020124	052522	022516			
	012036	000116					
4361	012040	047045	040445	046120	FMT24:	.ASCIZ	/N%PLEASE INSURE RUN SWITCH ON MICROPROCESSOR IS ON%N/

	012046	040505	042523	044440		
	012054	051516	051125	020105		
	012062	052522	020116	053523		
	012070	052111	044103	047440		
	012076	020116	044515	051103		
	012104	050117	047522	042503		
	012112	051523	051117	044440		
	012120	020123	047117	047045		
	012126	000				
4362						
4363						
4364						
4365	012127	103	051123	040440	EM1:	.ASCIZ /CSR ADDRESS TIME-OUT (SELO)/
	012134	042104	042522	051523		
	012142	052040	046511	026505		
	012150	052517	020124	051450		
	012156	046105	024460	000		
4366	012163	122	043505	047040	EM2:	.ASCIZ /REG NOT INITIALIZED BY MST CLR/
	012170	052117	044440	044516		
	012176	044524	046101	055111		
	012204	042105	041040	020131		
	012212	051515	020124	046103		
	012220	000122				
4367	012222	042522	020107	044515	EM3:	.ASCIZ /REG MISCOMPARE/
	012230	041523	046517	040520		
	012236	042522	000			
4368	012241	122	043505	047040	EM4:	.ASCIZ /REG NOT INITIALIZED BY UNIBUS RESET (INIT)/
	012246	052117	044440	044516		
	012254	044524	046101	055111		
	012262	042105	041040	020131		
	012270	047125	041111	051525		
	012276	051040	051505	052105		
	012304	024040	047111	052111		
	012312	000051				
4369	012314	040515	047111	020124	EM5:	.ASCIZ /MAINT CLK BIT STUCK AT 0/
	012322	046103	020113	044502		
	012330	020124	052123	041525		
	012336	020113	052101	030040		
	012344	000				
4370	012345	115	044501	052116	EM6:	.ASCIZ /MAINT CLK BIT STUCK AT 1/
	012352	041440	045514	041040		
	012360	052111	051440	052524		
	012366	045503	040440	020124		
	012374	000061				
4371	012376	051117	054504	047040	EM7:	.ASCIZ /ORDY NOT SET/
	012404	052117	051440	052105		
	012412	000				
4372	012413	117	042122	020131	EM8:	.ASCIZ /ORDY NOT CLEARED/
	012420	047516	020124	046103		
	012426	040505	042522	000104		
4373	012434	041517	051117	047040	EM9:	.ASCIZ /OCOR NOT SET/
	012442	052117	051440	052105		
	012450	000				
4374	012451	117	047503	020122	EM10:	.ASCIZ /OCOR NOT CLEARED/
	012456	047516	020124	046103		
	012464	040505	042522	000104		

4375	012472	040517	052103	047040	EM11:	.ASCIZ	/OACT NOT SET/
	012500	052117	051440	052105			
	012506	000					
4376	012507	117	041501	020124	EM12:	.ASCIZ	/OACT NOT CLEARED/
	012514	047516	020124	046103			
	012522	040505	042522	000104			
4377	012530	047125	051122	047040	EM13:	.ASCIZ	/UNRR NOT CLEARED BY SOM/
	012536	052117	041440	042514			
	012544	051101	042105	041040			
	012552	020131	047523	000115			
4378	012560	047125	051122	047040	EM14:	.ASCIZ	/UNRR NOT SET/
	012566	052117	051440	052105			
	012574	000					
4379	012575	125	051116	020122	EM15:	.ASCIZ	/UNRR NOT CLEARED BY OC/
	012602	047516	020124	046103			
	012610	040505	042522	020104			
	012616	054502	047440	000103			
4380	012624	047125	051122	047040	EM16:	.ASCIZ	/UNRR NOT CLEARED/
	012632	052117	041440	042514			
	012640	051101	042105	000			
4381	012645	111	042122	020131	EM17:	.ASCIZ	/IRDY NOT SET/
	012652	047516	020124	042523			
	012660	000124					
4382	012662	051111	054504	047040	EM18:	.ASCIZ	/IRDY NOT CLEARED/
	012670	052117	041440	042514			
	012676	051101	042105	000			
4383	012703	111	044503	020122	EM19:	.ASCIZ	/ICIR NOT SET/
	012710	047516	020124	042523			
	012716	000124					
4384	012720	041511	051111	047040	EM20:	.ASCIZ	/ICIR NOT CLEARED/
	012726	052117	041440	042514			
	012734	051101	042105	000			
4385	012741	111	041501	020124	EM21:	.ASCIZ	/IACT NOT SET/
	012746	047516	020124	042523			
	012754	000124					
4386	012756	040511	052103	047040	EM22:	.ASCIZ	/IACT NOT CLEARED/
	012764	052117	041440	042514			
	012772	051101	042105	000			
4387	012777	104	051523	020111	EM23:	.ASCIZ	/DSSI NOT CLEARED/
	013004	047516	020124	046103			
	013012	040505	042522	000104			
4388	013020	051504	044523	047040	EM24:	.ASCIZ	/DSSI NOT SET/
	013026	052117	051440	052105			
	013034	000					
4389	013035	104	051523	020111	EM25:	.ASCIZ	/DSSI NOT CLEARED BY MST CLR/
	013042	047516	020124	046103			
	013050	040505	042522	020104			
	013056	054502	046440	052123			
	013064	041440	051114	000			
4390	013071	111	041516	051117	EM26:	.ASCIZ	/INCORRECT DATA CHAR RCV'D/
	013076	042522	052103	042040			
	013104	052101	020101	044103			
	013112	051101	051040	053103			
	013120	042047	000				
4391	013123	111	041516	051117	EM27:	.ASCIZ	/INCORRECT CRC BYTE RCV'D/
	013130	042522	052103	041440			

	013136	041522	041040	052131		
	013144	020105	041522	023526		
	013152	000104				
4392	013154	051522	046517	047040	EM28:	.ASCIZ /RSOM NOT CLEARED/
	013162	052117	041440	042514		
	013170	051101	042105	000		
4393	013175	122	047523	020115	EM29:	.ASCIZ /RSOM NOT SET/
	013202	047516	020124	042523		
	013210	000124				
4394	013212	042522	046517	047040	EM30:	.ASCIZ /REOM NOT CLEARED/
	013220	052117	041440	042514		
	013226	051101	042105	000		
4395	013233	122	047505	020115	EM31:	.ASCIZ /REOM NOT SET/
	013240	047516	020124	042523		
	013246	000124				
4396	013250	054124	040504	040524	EM32:	.ASCIZ /TXDATA BIT NOT CLEARED/
	013256	041040	052111	047040		
	013264	052117	041440	042514		
	013272	051101	042105	000		
4397	013277	124	042130	052101	EM33:	.ASCIZ /TXDATA BIT NOT SET/
	013304	020101	044502	020124		
	013312	047516	020124	042523		
	013320	000124				
4398	013322	041522	023526	020104	EM34:	.ASCIZ /RCV'D DATA MISCOMPARE/
	013330	040504	040524	046440		
	013336	051511	047503	050115		
	013344	051101	000105			
4399	013350	041502	020103	047516	EM35:	.ASCIZ /BCC NOT CLEARED/
	013356	020124	046103	040505		
	013364	042522	000104			
4400	013370	041502	020103	047516	EM36:	.ASCIZ /BCC NOT SET/
	013376	020124	042523	000124		
4401	013404	041105	045514	047040	EM37:	.ASCIZ /EBLK NOT CLEARED/
	013412	052117	041440	042514		
	013420	051101	042105	000		
4402	013425	105	046102	020113	EM38:	.ASCIZ /EBLK NOT SET/
	013432	047516	020124	042523		
	013440	000124				
4403	013442	040522	020102	047516	EM39:	.ASCIZ /RAB NOT CLEARED/
	013450	020124	046103	040505		
	013456	042522	000104			
4404	013462	040522	020102	047516	EM40:	.ASCIZ /RAB NOT SET/
	013470	020124	042523	000124		
4405	013476	053117	051122	047040	EM41:	.ASCIZ /OVRR NOT CLEARED/
	013504	052117	041440	042514		
	013512	051101	042105	000		
4406	013517	117	051126	020122	EM42:	.ASCIZ /OVRR NOT SET/
	013524	047516	020124	042523		
	013532	000124				
4407	013534	053523	050040	041501	EM43:	.ASCIZ /SW PACK #1 INCORRECT/
	013542	020113	030443	044440		
	013550	041516	051117	042522		
	013556	052103	000			
4408	013561	123	020127	040520	EM44:	.ASCIZ /SW PACK #2 INCORRECT/
	013566	045503	021440	020062		
	013574	047111	047503	051122		

4409	013602	041505	000124			
	013606	053523	050040	041501	EM45:	.ASCIZ /SW PACK #3 INCORRECT/
	013614	020113	031443	044440		
	013622	041516	051117	042522		
	013630	052103	000			
4410	013633	122	053103	051440	EM46:	.ASCIZ /RCV SILO NOT CLEARED BY IC/
	013640	046111	020117	047516		
	013646	020124	046103	040505		
	013654	042522	020104	054502		
	013662	044440	000103			
4411	013666	051501	042523	041115	EM47:	.ASCIZ /ASSEMB BIT COUNT INCORRECT/
	013674	041040	052111	041440		
	013702	052517	052116	044440		
	013710	041516	051117	042522		
	013716	052103	000			
4412	013721	117	042104	053040	EM48:	.ASCIZ /ODD VRC PARITY BIT NOT SET/
	013726	041522	050040	051101		
	013734	052111	020131	044502		
	013742	020124	047516	020124		
	013750	042523	000124			
4413	013754	042117	020104	051126	EM49:	.ASCIZ /ODD VRC PARITY BIT NOT CLEARED/
	013762	020103	040520	044522		
	013770	054524	041040	052111		
	013776	047040	052117	041440		
	014004	042514	051101	042105		
	014012	000				
4414	014013	105	042526	020116	EM50:	.ASCIZ /EVEN VRC PARITY BIT NOT SET/
	014020	051126	020103	040520		
	014026	044522	054524	041040		
	014034	052111	047040	052117		
	014042	051440	052105	000		
4415	014047	105	042526	020116	EM51:	.ASCIZ /EVEN VRC PARITY BIT NOT CLEARED/
	014054	051126	020103	040520		
	014062	044522	054524	041040		
	014070	052111	047040	052117		
	014076	041440	042514	051101		
	014104	042105	000			
4416	014107	122	040505	054504	EM52:	.ASCIZ /READY NOT SET AFTER AX REG WRITE/
	014114	047040	052117	051440		
	014122	052105	040440	052106		
	014130	051105	040440	020130		
	014136	042522	020107	051127		
	014144	052111	000105			
4417	014150	042522	042101	020131	EM53:	.ASCIZ /READY NOT SET AFTER AX REG READ/
	014156	047516	020124	042523		
	014164	020124	043101	042524		
	014172	020122	054101	051040		
	014200	043505	051040	040505		
	014206	000104				
4418	014210	054124	052440	042116	EM54:	.ASCIZ /TX UNDERRUN ERROR/
	014216	051105	052522	020116		
	014224	051105	047522	000122		
4419	014232	052122	020123	047516	EM60:	.ASCIZ /RTS NOT SET/
	014240	020124	042523	000124		
4420	014245	052122	020123	047516	EM65:	.ASCIZ /RTS NOT CLEARED/
	014254	020124	046103	040505		

4421 014262 042522 000104
4422
4423
4424 014266 047111 052502 027523 DH1: .ASCIZ &INBUS/OUTBUS REG &
014274 052517 041124 051525
014302 051040 043505 000040
4425 014310 044514 042516 052440 DH2: .ASCIZ /LINE UNIT INBUS REGS :/
014316 044516 020124 047111
014324 052502 020123 042522
014332 051507 035040 000
4426 014337 122 043505 030061 DH3: .ASCIZ /REG10 REG11 REG12 REG13/
014344 020040 051040 043505
014352 030461 020040 051040
014360 043505 031061 020040
014366 051040 043505 031461
014374 000
4427 014375 040 020040 051040 DH4: .ASCIZ / REG14 REG15 REG16 REG17/
014402 043505 032061 020040
014410 051040 043505 032461
014416 020040 051040 043505
014424 033061 020040 051040
014432 043505 033461 000
4428 014437 061 000065 DH5: .ASCIZ /15/
4429 014442 033061 000 DH6: .ASCIZ /16/
4430 014445 114 047111 020105 DH7: .ASCIZ /LINE UNIT EXTENDED REGS :/
014452 047125 052111 042440
014460 052130 047105 042504
014466 020104 042522 051507
014474 035040 000
4431 014477 101 030130 030455 DH8: .ASCIZ /AX0-15 AX0-16 AX1-15 AX1-16/
014504 020065 040440 030130
014512 030455 020066 040440
014520 030530 030455 020065
014526 040440 030530 030455
014534 000066
4432 014536 020040 020040 054101 DH9: .ASCIZ / AX2-15 AX2-16 AX3-15 AX3-16/
014544 026462 032461 020040
014552 054101 026462 033061
014560 020040 054101 026463
014566 032461 020040 054101
014574 026463 033061 000

4433
4434 014602 .EVEN
4435
4436
4437
4438
4439

4440 014602 BGNMSG ERR1
(3) 014602
4441 014602 PRINTB #FMT1,#ADDRES,MPCSR
(9) 014602 013746 002446
(8) 014606 012746 035622
(7) 014612 012746 011454
(6) 014616 012746 000003

ERR1::

MOV MPCSR,-(SP)
MOV #ADDRES,-(SP)
MCV #FMT1,-(SP)
MOV #3,-(SP)

(3)	015044	010600				MOV	SP,RO
(4)	015046	104415				TRAP	C\$PNTX
(4)	015050	062706	000014			ADD	#14,SP
4453	015054			PRINTX	#FMT9,#DH4		
(8)	015054	012746	014375			MOV	#DH4,-(SP)
(7)	015060	012746	011712			MOV	#FMT9,-(SP)
(6)	015064	012746	000002			MOV	#2,-(SP)
(3)	015070	010600				MOV	SP,RO
(4)	015072	104415				TRAP	C\$PNTX
(4)	015074	062706	000006			ADD	#6,SP
4454	015100			PRINTX	#FMT6,LUR14,LUR15,LUR16,LUR17		
(11)	015100	013746	002320			MOV	LUR17,-(SP)
(10)	015104	013746	002316			MOV	LUR16,-(SP)
(9)	015110	013746	002314			MOV	LUR15,-(SP)
(8)	015114	013746	002312			MOV	LUR14,-(SP)
(7)	015120	012746	011613			MOV	#FMT6,-(SP)
(6)	015124	012746	000005			MOV	#5,-(SP)
(3)	015130	010600				MOV	SP,RO
(4)	015132	104415				TRAP	C\$PNTX
(4)	015134	062706	000014			ADD	#14,SP
4455	015140			ENDMSG			
(3)	015140						
(3)	015140	104423				L10003:	TRAP C\$MSG
4456							
4457							
4458							
4459							
4460							
4461	015142			BGNMSG	ERR3		
(3)	015142						
4462	015142			PRINTB	#FMT1,#ADDRES,MPCSR	ERR3::	
(9)	015142	013746	002446			MOV	MPCSR,-(SP)
(8)	015146	012746	035622			MOV	#ADDRES,-(SP)
(7)	015152	012746	011454			MOV	#FMT1,-(SP)
(6)	015156	012746	000003			MOV	#3,-(SP)
(3)	015162	010600				MOV	SP,RO
(4)	015164	104414				TRAP	C\$PNTB
(4)	015166	062706	000010			ADD	#10,SP
4463	015172			PRINTB	#FMT2		
(7)	015172	012746	011464			MOV	#FMT2,-(SP)
(6)	015176	012746	000001			MOV	#1,-(SP)
(3)	015202	010600				MOV	SP,RO
(4)	015204	104414				TRAP	C\$PNTB
(4)	015206	062706	000004			ADD	#4,SP
4464	015212			PRINTB	#FMT8,TMP1,TMP0		
(9)	015212	013746	002530			MOV	TMP0,-(SP)
(8)	015216	013746	002532			MOV	TMP1,-(SP)
(7)	015222	012746	011656			MOV	#FMT8,-(SP)
(6)	015226	012746	000003			MOV	#3,-(SP)
(3)	015232	010600				MOV	SP,RO
(4)	015234	104414				TRAP	C\$PNTB
(4)	015236	062706	000010			ADD	#10,SP
4465	015242			PRINTB	#FMT3,GOODAT,BADDAT		
(9)	015242	013746	002406			MOV	BADDAT,-(SP)
(8)	015246	013746	002404			MOV	GOODAT,-(SP)
(7)	015252	012746	011506			MOV	#FMT3,-(SP)

(6) 015256 012746 000003
(3) 015262 010600
(4) 015264 104414
(4) 015266 062706 000010
4466 015272
(9) 015272 012746 014337
(8) 015276 012746 014310
(7) 015302 012746 011550
(6) 015306 012746 000003
(3) 015312 010600
(4) 015314 104415
(4) 015316 062706 000010
4467 015322
(11) 015322 013746 002310
(10) 015326 013746 002306
(9) 015332 013746 002304
(8) 015336 013746 002302
(7) 015342 012746 011563
(6) 015346 012746 000005
(3) 015352 010600
(4) 015354 104415
(4) 015356 062706 000014
4468 015362
(8) 015362 012746 014375
(7) 015366 012746 011712
(6) 015372 012746 000002
(3) 015376 010600
(4) 015400 104415
(4) 015402 062706 000006
4469 015406
(11) 015406 013746 002320
(10) 015412 013746 002316
(9) 015416 013746 002314
(8) 015422 013746 002312
(7) 015426 012746 011613
(6) 015432 012746 000005
(3) 015436 010600
(4) 015440 104415
(4) 015442 062706 000014
4470 015446
(9) 015446 012746 014477
(8) 015452 012746 014445
(7) 015456 012746 011550
(6) 015462 012746 000003
(3) 015466 010600
(4) 015470 104415
(4) 015472 062706 000010
4471 015476
(11) 015476 013746 002330
(10) 015502 013746 002326
(9) 015506 013746 002324
(8) 015512 013746 002322
(7) 015516 012746 011563
(6) 015522 012746 000005
(3) 015526 010600
(4) 015530 104415

PRINTX #FMT4,#DH2,#DH3

PRINTX #FMT5,LUR10,LUR11,LUR12,LUR13

PRINTX #FMT9,#DH4

PRINTX #FMT6,LUR14,LUR15,LUR16,LUR17

PRINTX #FMT4,#DH7,#DH8

PRINTX #FMT5,AX0.15,AX0.16,AX1.15,AX1.16

MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #10,SP

MOV #DH3,-(SP)
MOV #DH2,-(SP)
MOV #FMT4,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #10,SP

MOV LUR13,-(SP)
MOV LUR12,-(SP)
MOV LUR11,-(SP)
MOV LUR10,-(SP)
MOV #FMT5,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #14,SP

MOV #DH4,-(SP)
MOV #FMT9,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #6,SP

MOV LUR17,-(SP)
MOV LUR16,-(SP)
MOV LUR15,-(SP)
MOV LUR14,-(SP)
MOV #FMT6,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #14,SP

MOV #DH8,-(SP)
MOV #DH7,-(SP)
MOV #FMT4,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #10,SP

MOV AX1.16,-(SP)
MOV AX1.15,-(SP)
MOV AX0.16,-(SP)
MOV AX0.15,-(SP)
MOV #FMT5,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX

4472	(4)	015532	062706	000014			ADD	#14,SP
		015536			PRINTX	#FMT9,#DH9		
	(8)	015536	012746	014536			MOV	#DH9,-(SP)
	(7)	015542	012746	011712			MOV	#FMT9,-(SP)
	(6)	015546	012746	000002			MOV	#2,-(SP)
	(3)	015552	010600				MOV	SP,R0
	(4)	015554	104415				TRAP	C\$PNTX
	(4)	015556	062706	000006			ADD	#6,SP
4473		015562			PRINTX	#FMT6,AX2.15,AX2.16,AX3.15,AX3.16		
	(11)	015562	013746	002340			MOV	AX3.16,-(SP)
	(10)	015566	013746	002336			MOV	AX3.15,-(SP)
	(9)	015572	013746	002334			MOV	AX2.16,-(SP)
	(8)	015576	013746	002332			MOV	AX2.15,-(SP)
	(7)	015602	012746	011613			MOV	#FMT6,-(SP)
	(6)	015606	012746	000005			MOV	#5,-(SP)
	(3)	015612	010600				MOV	SP,R0
	(4)	015614	104415				TRAP	C\$PNTX
	(4)	015616	062706	000014			ADD	#14,SP
4474		015622			ENDMSG			
	(3)	015622						
	(3)	015622	104423				L10004:	TRAP C\$MSG
4475								
4476								
4477								
4478								
4479								
4480		015624			BGNMSG	ERR4		
	(3)	015624						
4481		015624			PRINTB	#FMT10,SUBRPC		ERR4::
	(8)	015624	013746	002352			MOV	SUBRPC,-(SP)
	(7)	015630	012746	011717			MOV	#FMT10,-(SP)
	(6)	015634	012746	000002			MOV	#2,-(SP)
	(3)	015640	010600				MOV	SP,R0
	(4)	015642	104414				TRAP	C\$PNTB
	(4)	015644	062706	000006			ADD	#6,SP
4482		015650			PRINTB	#FMT1,#ADDRES,MPCSR		
	(9)	015650	013746	002446			MOV	MPCSR,-(SP)
	(8)	015654	012746	035622			MOV	#ADDRES,-(SP)
	(7)	015660	012746	011454			MOV	#FMT1,-(SP)
	(6)	015664	012746	000003			MOV	#3,-(SP)
	(3)	015670	010600				MOV	SP,R0
	(4)	015672	104414				TRAP	C\$PNTB
	(4)	015674	062706	000010			ADD	#10,SP
4483		015700			PRINTB	#FMT2		
	(7)	015700	012746	011464			MOV	#FMT2,-(SP)
	(6)	015704	012746	000001			MOV	#1,-(SP)
	(3)	015710	010600				MOV	SP,R0
	(4)	015712	104414				TRAP	C\$PNTB
	(4)	015714	062706	000004			ADD	#4,SP
4484		015720			PRINTB	#FMT7,#DH1,REGNUM		
	(9)	015720	013746	002400			MOV	REGNUM,-(SP)
	(8)	015724	012746	014266			MOV	#DH1,-(SP)
	(7)	015730	012746	011646			MOV	#FMT7,-(SP)
	(6)	015734	012746	000003			MOV	#3,-(SP)
	(3)	015740	010600				MOV	SP,R0
	(4)	015742	104414				TRAP	C\$PNTB

(4)	015744	062706	000010				
4485	015750			PRINTX	#FMT4,#DH2,#DH3	ADD	#10,SP
(9)	015750	012746	014337			MOV	#DH3,-(SP)
(8)	015754	012746	014310			MOV	#DH2,-(SP)
(7)	015760	012746	011550			MOV	#FMT4,-(SP)
(6)	015764	012746	000003			MOV	#3,-(SP)
(3)	015770	010600				MOV	SP,R0
(4)	015772	104415				TRAP	C\$PNTX
(4)	015774	062706	000010			ADD	#10,SP
4486	016000			PRINTX	#FMT5,LUR10,LUR11,LUR12,LUR13		
(11)	016000	013746	002310			MOV	LUR13,-(SP)
(10)	016004	013746	002306			MOV	LUR12,-(SP)
(9)	016010	013746	002304			MOV	LUR11,-(SP)
(8)	016014	013746	002302			MOV	LUR10,-(SP)
(7)	016020	012746	011563			MOV	#FMT5,-(SP)
(6)	016024	012746	000005			MOV	#5,-(SP)
(3)	016030	010600				MOV	SP,R0
(4)	016032	104415				TRAP	C\$PNTX
(4)	016034	062706	000014			ADD	#14,SP
4487	016040			PRINTX	#FMT9,#DH4		
(8)	016040	012746	014375			MOV	#DH4,-(SP)
(7)	016044	012746	011712			MOV	#FMT9,-(SP)
(6)	016050	012746	000002			MOV	#2,-(SP)
(3)	016054	010600				MOV	SP,R0
(4)	016056	104415				TRAP	C\$PNTX
(4)	015060	062706	000006			ADD	#6,SP
4488	016064			PRINTX	#FMT6,LUR14,LUR15,LUR16,LUR17		
(11)	016064	013746	002320			MOV	LUR17,-(SP)
(10)	016070	013746	002316			MOV	LUR16,-(SP)
(9)	016074	013746	002314			MOV	LUR15,-(SP)
(8)	016100	013746	002312			MOV	LUR14,-(SP)
(7)	016104	012746	011613			MOV	#FMT6,-(SP)
(6)	016110	012746	000005			MOV	#5,-(SP)
(3)	016114	010600				MOV	SP,R0
(4)	016116	104415				TRAP	C\$PNTX
(4)	016120	062706	000014			ADD	#14,SP
4489	016124			PRINTX	#FMT4,#DH7,#DH8		
(9)	016124	012746	014477			MOV	#DH8,-(SP)
(8)	016130	012746	014445			MOV	#DH7,-(SP)
(7)	016134	012746	011550			MOV	#FMT4,-(SP)
(6)	016140	012746	000003			MOV	#3,-(SP)
(3)	016144	010600				MOV	SP,R0
(4)	016146	104415				TRAP	C\$PNTX
(4)	016150	062706	000010			ADD	#10,SP
4490	016154			PRINTX	#FMT5,AX0.15,AX0.16,AX1.15,AX1.16		
(11)	016154	013746	002330			MOV	AX1.16,-(SP)
(10)	016160	013746	002326			MOV	AX1.15,-(SP)
(9)	016164	013746	002324			MOV	AX0.16,-(SP)
(8)	016170	013746	002322			MOV	AX0.15,-(SP)
(7)	016174	012746	011563			MOV	#FMT5,-(SP)
(6)	016200	012746	000005			MOV	#5,-(SP)
(3)	016204	010600				MOV	SP,R0
(4)	016206	104415				TRAP	C\$PNTX
(4)	016210	062706	000014			ADD	#14,SP
4491	016214			PRINTX	#FMT9,#DH9		
(8)	016214	012746	014536			MOV	#DH9,-(SP)

(7) 016220 012746 011712
(6) 016224 012746 000002
(3) 016230 010600
(4) 016232 104415
(4) 016234 062706 000006
4492 016240
(11) 016240 013746 002340
(10) 016244 013746 002336
(9) 016250 013746 002334
(8) 016254 013746 002332
(7) 016260 012746 011613
(6) 016264 012746 000005
(3) 016270 010600
(4) 016272 104415
(4) 016274 062706 000014
4493 016300
(3) 016300
(3) 016300 104423
4494
4495
4496
4497
4498
4499 016302
(3) 016302
4500 016302
(9) 016302 013746 002446
(8) 016306 012746 035622
(7) 016312 012746 011454
(6) 016316 012746 000003
(3) 016322 010600
(4) 016324 104414
(4) 016326 062706 000010
4501 016332
(9) 016332 013746 002410
(8) 016336 013746 002400
(7) 016342 012746 011750
(6) 016346 012746 000003
(3) 016352 010600
(4) 016354 104414
(4) 016356 062706 000010
4502 016362
(7) 016362 012746 011464
(6) 016366 012746 000001
(3) 016372 010600
(4) 016374 104414
(4) 016376 062706 000004
4503 016402
(9) 016402 013746 002530
(8) 016406 013746 002532
(7) 016412 012746 011656
(6) 016416 012746 000003
(3) 016422 010600
(4) 016424 104414
(4) 016426 062706 000010
4504 016432

PRINTX #FMT6,AX2.15,AX2.16,AX3.15,AX3.16

ENDMSG

BGNMSG ERR5

PRINTB #FMT1,#ADDRES,MPCSR

PRINTB #FMT11,REGNUM,LOADAT

PRINTB #FMT2

PRINTB #FMT8,TMP1,TMP0

PRINTB #FMT3,GOODAT,BADDAT

MOV #FMT9,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #6,SP

MOV AX3.16,-(SP)
MOV AX3.15,-(SP)
MOV AX2.16,-(SP)
MOV AX2.15,-(SP)
MOV #FMT6,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #14,SP

L10005:

TRAP C\$MSG

ERR5::

MOV MPCSR,-(SP)
MOV #ADDRES,-(SP)
MOV #FMT1,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #10,SP

MOV LOADAT,-(SP)
MOV REGNUM,-(SP)
MOV #FMT11,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #10,SP

MOV #FMT2,-(SP)
MOV #1,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #4,SP

MOV TMP0,-(SP)
MOV TMP1,-(SP)
MOV #FMT8,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #10,SP

(9) 016432 013746 002406
(8) 016436 013746 002404
(7) 016442 012746 011506
(6) 016446 012746 000003
(3) 016452 010600
(4) 016454 104414
(4) 016456 062706 000010
4505 016462
(9) 016462 012746 014337
(8) 016466 012746 014310
(7) 016472 012746 011550
(6) 016476 012746 000003
(3) 016502 010600
(4) 016504 104415
(4) 016506 062706 000010
4506 016512
(11) 016512 013746 002310
(10) 016516 013746 002306
(9) 016522 013746 002304
(8) 016526 013746 002302
(7) 016532 012746 011563
(6) 016536 012746 000005
(3) 016542 010600
(4) 016544 104415
(4) 016546 062706 000014
4507 016552
(8) 016552 012746 014375
(7) 016556 012746 011712
(6) 016562 012746 000002
(3) 016566 010600
(4) 016570 104415
(4) 016572 062706 000006
4508 016576
(11) 016576 013746 002320
(10) 016602 013746 002316
(9) 016606 013746 002314
(8) 016612 013746 002312
(7) 016616 012746 011613
(6) 016622 012746 000005
(3) 016626 010600
(4) 016630 104415
(4) 016632 062706 000014
4509 016636
(9) 016636 012746 014477
(8) 016642 012746 014445
(7) 016646 012746 011550
(6) 016652 012746 000003
(3) 016656 010600
(4) 016660 104415
(4) 016662 062706 000010
4510 016666
(11) 016666 013746 002330
(10) 016672 013746 002326
(9) 016676 013746 002324
(8) 016702 013746 002322
(7) 016706 012746 011563

PRINTX #FMT4,#DH2,#DH3

PRINTX #FMT5,LUR10,LUR11,LUR12,LUR13

PRINTX #FMT9,#DH4

PRINTX #FMT6,LUR14,LUR15,LUR16,LUR17

PRINTX #FMT4,#DH7,#DH8

PRINTX #FMT5,AX0.15,AX0.16,AX1.15,AX1.16

MOV BADDAT,-(SP)
MOV GOODAT,-(SP)
MOV #FMT3,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #10,SP

MOV #DH3,-(SP)
MOV #DH2,-(SP)
MOV #FMT4,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #10,SP

MOV LUR13,-(SP)
MOV LUR12,-(SP)
MOV LUR11,-(SP)
MOV LUR10,-(SP)
MOV #FMT5,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #14,SP

MOV #DH4,-(SP)
MOV #FMT9,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #6,SP

MOV LUR17,-(SP)
MOV LUR16,-(SP)
MOV LUR15,-(SP)
MOV LUR14,-(SP)
MOV #FMT6,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #14,SP

MOV #DH8,-(SP)
MOV #DH7,-(SP)
MOV #FMT4,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #10,SP

MOV AX1.16,-(SP)
MOV AX1.15,-(SP)
MOV AX0.16,-(SP)
MOV AX0.15,-(SP)
MOV #FMT5,-(SP)

(6) 016712 012746 000005
(3) 016716 010600
(4) 016720 104415
(4) 016722 062706 000014
4511 016726
(8) 016726 012746 014536
(7) 016732 012746 011712
(6) 016736 012746 000002
(3) 016742 010600
(4) 016744 104415
(4) 016746 062706 000006
4512 016752
(11) 016752 013746 002340
(10) 016756 013746 002336
(9) 016762 013746 002334
(8) 016766 013746 002332
(7) 016772 012746 011613
(6) 016776 012746 000005
(3) 017002 010600
(4) 017004 104415
(4) 017006 062706 000014
4513 017012
(3) 017012
(3) 017012 104423
4514
4515
4516
4517
4518
4519 017014
(3) 017014
4520 017014
(8) 017014 013746 002352
(7) 017020 012746 011717
(6) 017024 012746 000002
(3) 017030 010600
(4) 017032 104414
(4) 017034 062706 000006
4521 017040
(9) 017040 013746 002446
(8) 017044 012746 035622
(7) 017050 012746 011454
(6) 017054 012746 000003
(3) 017060 010600
(4) 017062 104414
(4) 017064 062706 000010
4522 017070
(7) 017070 012746 011464
(6) 017074 012746 000001
(3) 017100 010600
(4) 017102 104414
(4) 017104 062706 000004
4523 017110
(9) 017110 013746 002530
(8) 017114 013746 002532
(7) 017120 012746 011656

PRINTX #FMT9,#DH9

PRINTX #FMT6,AX2.15,AX2.16,AX3.15,AX3.16

ENDMSG

BGNMSG ERR6

PRINTB #FMT10,SUBPPC

PRINTB #FMT1,#ADDRES,MPCSR

PRINTB #FMT2

PRINTB #FMT8,TMP1,TMP0

MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #14,SP

MOV #DH9,-(SP)
MOV #FMT9,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #6,SP

MOV AX3.16,-(SP)
MOV AX3.15,-(SP)
MOV AX2.16,-(SP)
MOV AX2.15,-(SP)
MOV #FMT6,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #14,SP

L10006: TRAP C\$MSG

ERR6::

MOV SUBRPC,-(SP)
MOV #FMT10,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #6,SP

MOV MPCSR,-(SP)
MOV #ADDRES,-(SP)
MOV #FMT1,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #10,SP

MOV #FMT2,-(SP)
MOV #1,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #4,SP

MOV TMP0,-(SP)
MOV TMP1,-(SP)
MOV #FMT8,-(SP)

(6) 017124 012746 000003
(3) 017130 010600
(4) 017132 104414
(4) 017134 062706 000010
4524 017140
(9) 017140 012746 014337
(8) 017144 012746 014310
(7) 017150 012746 011550
(6) 017154 012746 000003
(3) 017160 010600
(4) 017162 104415
(4) 017164 062706 000010
4525 017170
(11) 017170 013746 002310
(10) 017174 013746 002306
(9) 017200 013746 002304
(8) 017204 013746 002302
(7) 017210 012746 011563
(6) 017214 012746 000005
(3) 017220 010600
(4) 017222 104415
(4) 017224 062706 000014
4526 017230
(8) 017230 012746 014375
(7) 017234 012746 011712
(6) 017240 012746 000002
(3) 017244 010600
(4) 017246 104415
(4) 017250 062706 000006
4527 017254
(11) 017254 013746 002320
(10) 017260 013746 002316
(9) 017264 013746 002314
(8) 017270 013746 002312
(7) 017274 012746 011613
(6) 017300 012746 000005
(3) 017304 010600
(4) 017306 104415
(4) 017310 062706 000014
4528 017314
(9) 017314 012746 014477
(8) 017320 012746 014445
(7) 017324 012746 011550
(6) 017330 012746 000003
(3) 017334 010600
(4) 017336 104415
(4) 017340 062706 000010
4529 017344
(11) 017344 013746 002330
(10) 017350 013746 002326
(9) 017354 013746 002324
(8) 017360 013746 002322
(7) 017364 012746 011563
(6) 017370 012746 000005
(3) 017374 010600
(4) 017376 104415

PRINTX #FMT4,#DH2,#DH3

PRINTX #FMT5,LUR10,LUR11,LUR12,LUR13

PRINTX #FMT9,#DH4

PRINTX #FMT6,LUR14,LUR15,LUR16,LUR17

PRINTX #FMT4,#DH7,#DH8

PRINTX #FMT5,AX0.15,AX0.16,AX1.15,AX1.16

MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTB
ADD #10,SP

MOV #DH3,-(SP)
MOV #DH2,-(SP)
MOV #FMT4,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #10,SP

MOV LUR13,-(SP)
MOV LUR12,-(SP)
MOV LUR11,-(SP)
MOV LUR10,-(SP)
MOV #FMT5,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #14,SP

MOV #DH4,-(SP)
MOV #FMT9,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #6,SP

MOV LUR17,-(SP)
MOV LUR16,-(SP)
MOV LUR15,-(SP)
MOV LUR14,-(SP)
MOV #FMT6,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #14,SP

MOV #DH8,-(SP)
MOV #DH7,-(SP)
MOV #FMT4,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #10,SP

MOV AX1.16,-(SP)
MOV AX1.15,-(SP)
MOV AX0.16,-(SP)
MOV AX0.15,-(SP)
MOV #FMT5,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX

(4)	017614	104415							
(4)	017616	062706	000010				TRAP	C\$PNTX	
4543	017622			PRINTX	#FMT5,LUR10,LUR11,LUR12,LUR13		ADD	#10,SP	
(11)	017622	013746	002310				MOV	LUR13,-(SP)	
(10)	017626	013746	002306				MOV	LUR12,-(SP)	
(9)	017632	013746	002304				MOV	LUR11,-(SP)	
(8)	017636	013746	002302				MOV	LUR10,-(SP)	
(7)	017642	012746	011563				MOV	#FMT5,-(SP)	
(6)	017646	012746	000005				MOV	#5,-(SP)	
(5)	017652	010600					MOV	SP,R0	
(4)	017654	104415					TRAP	C\$PNTX	
(4)	017656	062706	000014				ADD	#14,SP	
4544	017662			PRINTX	#FMT9,#DH4				
(8)	017662	012746	014375				MOV	#DH4,-(SP)	
(7)	017666	012746	011712				MOV	#FMT9,-(SP)	
(6)	017672	012746	000002				MOV	#2,-(SP)	
(3)	017676	010600					MOV	SP,R0	
(4)	017700	104415					TRAP	C\$PNTX	
(4)	017702	062706	000006				ADD	#6,SP	
4545	017706			PRINTX	#FMT6,LUR14,LUR15,LUR16,LUR17				
(11)	017706	013746	002320				MOV	LUR17,-(SP)	
(10)	017712	013746	002316				MOV	LUR16,-(SP)	
(9)	017716	013746	002314				MOV	LUR15,-(SP)	
(8)	017722	013746	002312				MOV	LUR14,-(SP)	
(7)	017726	012746	011613				MOV	#FMT6,-(SP)	
(6)	017732	012746	000005				MOV	#5,-(SP)	
(3)	017736	010600					MOV	SP,R0	
(4)	017740	104415					TRAP	C\$PNTX	
(4)	017742	062706	000014				ADD	#14,SP	
4546	017746			PRINTX	#FMT4,#DH7,#DH8				
(9)	017746	012746	014477				MOV	#DH8,-(SP)	
(8)	017752	012746	014445				MOV	#DH7,-(SP)	
(7)	017756	012746	011550				MOV	#FMT4,-(SP)	
(6)	017762	012746	000003				MOV	#3,-(SP)	
(3)	017766	010600					MOV	SP,R0	
(4)	017770	104415					TRAP	C\$PNTX	
(4)	017772	062706	000010				ADD	#10,SP	
4547	017776			PRINTX	#FMT5,AX0.15,AX0.16,AX1.15,AX1.16				
(11)	017776	013746	002330				MOV	AX1.16,-(SP)	
(10)	020007	013746	002326				MOV	AX1.15,-(SP)	
(9)	020006	013746	002324				MOV	AX0.16,-(SP)	
(8)	020012	013746	002322				MOV	AX0.15,-(SP)	
(7)	020016	012746	011563				MOV	#FMT5,-(SP)	
(6)	020022	012746	000005				MOV	#5,-(SP)	
(3)	020026	010600					MOV	SP,R0	
(4)	020030	104415					TRAP	C\$PNTX	
(4)	020032	062706	000014				ADD	#14,SP	
4548	020036			PRINTX	#FMT9,#DH9				
(8)	020036	012746	014536				MOV	#DH9,-(SP)	
(7)	020042	012746	011712				MOV	#FMT9,-(SP)	
(6)	020046	012746	000002				MOV	#2,-(SP)	
(3)	020052	010600					MOV	SP,R0	
(4)	020054	104415					TRAP	C\$PNTX	
(4)	020056	062706	000006				ADD	#6,SP	
4549	020062			PRINTX	#FMT6,AX2.15,AX2.16,AX3.15,AX3.16				
(1)	020062	013746	002340				MOV	AX3.16,-(SP)	

(10)	020066	013746	002336				MOV	AX3.15,-(SP)
(9)	020072	013746	002334				MOV	AX2.16,-(SP)
(8)	020076	013746	002332				MOV	AX2.15,-(SP)
(7)	020102	012746	011613				MOV	#FMT6,-(SP)
(6)	020106	012746	000005				MOV	#5,-(SP)
(3)	020112	010600					MOV	SP,R0
(4)	020114	104415					TRAP	C\$PNTX
(4)	020116	062706	000014				ADD	#14,SP
4550	020122			ENDMSG				
(3)	020122					L10010:		
(3)	020122	104423				TRAP		C\$MSG
4551								
4552								
4553								
4554								
4555								
4556	020124			BGNMSG	ERR8			
(3)	020124					ERR8::		
4557	020124			PRINTB	#FMT10,SUBRPC			
(8)	020124	013746	002352				MOV	SUBRPC,-(SP)
(7)	020130	012746	011717				MOV	#FMT10,-(SP)
(6)	020134	012746	000002				MOV	#2,-(SP)
(3)	020140	010600					MOV	SP,R0
(4)	020142	104414					TRAP	C\$PNTB
(4)	020144	062706	000006				ADD	#6,SP
4558	020150			PRINTB	#FMT1,#ADDRES,MPCSR			
(9)	020150	013746	002446				MOV	MPCSR,-(SP)
(8)	020154	012746	035622				MOV	#ADDRES,-(SP)
(7)	020160	012746	011454				MOV	#FMT1,-(SP)
(6)	020164	012746	000003				MOV	#3,-(SP)
(3)	020170	010600					MOV	SP,R0
(4)	020172	104414					TRAP	C\$PNTB
(4)	020174	062706	000010				ADD	#10,SP
4559	020200			PRINTB	#FMT2			
(7)	020200	012746	011464				MOV	#FMT2,-(SP)
(6)	020204	012746	000001				MOV	#1,-(SP)
(3)	020210	010600					MOV	SP,R0
(4)	020212	104414					TRAP	C\$PNTB
(4)	02 214	062706	000004				ADD	#4,SP
4560	020220			PRINTB	#FMT7,#DH1,REGNUM			
(9)	020220	013746	002400				MOV	REGNUM,-(SP)
(8)	020224	012746	014266				MOV	#DH1,-(SP)
(7)	020230	012746	011646				MOV	#FMT7,-(SP)
(6)	020234	012746	000003				MOV	#3,-(SP)
(3)	020240	010600					MOV	SP,R0
(4)	020242	104414					TRAP	C\$PNTB
(4)	020244	062706	000010				ADD	#10,SP
4561	020250			PRINTB	#FMT3,GOODAT,BADDAT			
(9)	020250	013746	002406				MOV	BADDAT,-(SP)
(8)	020254	013746	002404				MOV	GOODAT,-(SP)
(7)	020260	012746	011506				MOV	#FMT3,-(SP)
(6)	020264	012746	000003				MOV	#3,-(SP)
(3)	020270	010600					MOV	SP,R0
(4)	020272	104414					TRAP	C\$PNTB
(4)	020274	062706	000010				ADD	#10,SP
4562	020300			PRINTX	#FMT4,#DH2,#DH3			

(9) 020300 012746 014337
(8) 020304 012746 014310
(7) 020310 012746 011550
(6) 020314 012746 000003
(3) 020320 010600
(4) 020322 104415
(4) 020324 062706 000010
4563 020330
(11) 020330 013746 002310
(10) 020334 013746 002306
(9) 020340 013746 002304
(8) 020344 013746 002302
(7) 020350 012746 011563
(6) 020354 012746 000005
(3) 020360 010600
(4) 020362 104415
(4) 020364 062706 000014
4564 020370
(8) 020370 012746 014375
(7) 020374 012746 011712
(6) 020400 012746 000002
(3) 020404 010600
(4) 020406 104415
(4) 020410 062706 000006
4565 020414
(11) 020414 013746 002320
(10) 020420 013746 002316
(9) 020424 013746 002314
(8) 020430 013746 002312
(7) 020434 012746 011613
(6) 020440 012746 000005
(3) 020444 010600
(4) 020446 104415
(4) 020450 062706 000014
4566 020454
(9) 020454 012746 014477
(8) 020460 012746 014445
(7) 020464 012746 011550
(6) 020470 012746 000003
(3) 020474 010600
(4) 020476 104415
(4) 020500 062706 000010
4567 020504
(11) 020504 013746 002330
(10) 020510 013746 002326
(9) 020514 013746 002324
(8) 020520 013746 002322
(7) 020524 012746 011563
(6) 020530 012746 000005
(3) 020534 010600
(4) 020536 104415
(4) 020540 062706 000014
4568 020544
(8) 020544 012746 014536
(7) 020550 012746 011712
(6) 020554 012746 000002

PRINTX #FMT5,LUR10,LUR11,LUR12,LUR13

PRINTX #FMT9,#DH4

PRINTX #FMT6,LUR14,LUR15,LUR16,LUR17

PRINTX #FMT4,#DH7,#DH8

PRINTX #FMT5,AX0.15,AX0.16,AX1.15,AX1.16

PRINTX #FMT9,#DH9

MOV #DH3,-(SP)
MOV #DH2,-(SP)
MOV #FMT4,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP CSPNTX
ADD #10,SP

MOV LUR13,-(SP)
MOV LUR12,-(SP)
MOV LUR11,-(SP)
MOV LUR10,-(SP)
MOV #FMT5,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP CSPNTX
ADD #14,SP

MOV #DH4,-(SP)
MOV #FMT9,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP CSPNTX
ADD #6,SP

MOV LUR17,-(SP)
MOV LUR16,-(SP)
MOV LUR15,-(SP)
MOV LUR14,-(SP)
MOV #FMT6,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP CSPNTX
ADD #14,SP

MOV #DH8,-(SP)
MOV #DH7,-(SP)
MOV #FMT4,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP CSPNTX
ADD #10,SP

MOV AX1.16,-(SP)
MOV AX1.15,-(SP)
MOV AX0.16,-(SP)
MOV AX0.15,-(SP)
MOV #FMT5,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP CSPNTX
ADD #14,SP

MOV #DH9,-(SP)
MOV #FMT9,-(SP)
MOV #2,-(SP)

(3)	020560	010600							
(4)	020562	104415							
(4)	020564	062706	000006						
4569	020570			PRINTX	#FMT6,AX2.15,AX2.16,AX3.15,AX3.16				
(11)	020570	013746	002340						
(10)	020574	013746	002336						
(9)	020600	013746	002334						
(8)	020604	013746	002332						
(7)	020610	012746	011613						
(6)	020614	012746	000005						
(3)	020620	010600							
(4)	020622	104415							
(4)	020624	062706	000014						
4570	020630			ENDMSG					
(3)	020630								
(3)	020630	104423							
4571									
4572									
4573									
4574									
4575									
4576	020632			BGNMSG	ERR9				
(3)	020632								
4577	020632			PRINTB	#FMT1,#ADDRES,MPCSR				
(9)	020632	013746	002446						
(8)	020636	012746	035622						
(7)	020642	012746	011454						
(6)	020646	012746	000003						
(3)	020652	010600							
(4)	020654	104414							
(4)	020656	062706	000010						
4578	020662			PRINTB	#FMT2				
(7)	020662	012746	011464						
(6)	020666	012746	000001						
(3)	020672	010600							
(4)	020674	104414							
(4)	020676	062706	000004						
4579	020702			PRINTB	#FMT7,#DH1,REGNUM				
(9)	020702	013746	002400						
(8)	020706	012746	014266						
(7)	020712	012746	011646						
(6)	020716	012746	000003						
(3)	020722	010600							
(4)	020724	104414							
(4)	020726	062706	000010						
4580	020732			PRINTX	#FMT4,#DH2,#DH3				
(9)	020732	012746	014337						
(8)	020736	012746	014310						
(7)	020742	012746	011550						
(6)	020746	012746	000003						
(3)	020752	010600							
(4)	020754	104415							
(4)	020756	062706	000010						
4581	020762			PRINTX	#FMT5,LUR10,LUR11,LUR12,LUR13				
(11)	020762	013746	002310						
(10)	020766	013746	002306						

```

MOV SP,R0
TRAP C$PNTX
ADD #6,SP

MOV AX3.16,-(SP)
MOV AX3.15,-(SP)
MOV AX2.16,-(SP)
MOV AX2.15,-(SP)
MOV #FMT6,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP C$PNTX
ADD #14,SP

L10011:
TRAP C$MSG

ERR9::
MOV MPCSR,-(SP)
MOV #ADDRES,-(SP)
MOV #FMT1,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C$PNTB
ADD #10,SP

MOV #FMT2,-(SP)
MOV #1,-(SP)
MOV SP,R0
TRAP C$PNTB
ADD #4,SP

MOV REGNUM,-(SP)
MOV #DH1,-(SP)
MOV #FMT7,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C$PNTB
ADD #10,SP

MOV #DH3,-(SP)
MOV #DH2,-(SP)
MOV #FMT4,-(SP)
MOV #3,-(SP)
MOV SP,R0
TRAP C$PNTX
ADD #10,SP

MOV LUR13,-(SP)
MOV LUR12,-(SP)

```

(9) 020772 013746 002304
(8) 020776 013746 002302
(7) 021002 012746 011563
(6) 021006 012746 000005
(3) 021012 010600
(4) 021014 104415
(4) 021016 062706 000014
4582 021022
(8) 021022 012746 014375
(7) 021026 012746 011712
(6) 021032 012746 000002
(3) 021036 010600
(4) 021040 104415
(4) 021042 062706 000006
4583 021046
(11) 021046 013746 002320
(10) 021052 013746 002316
(9) 021056 013746 002314
(8) 021062 013746 002312
(7) 021066 012746 011613
(6) 021072 012746 000005
(3) 021076 010600
(4) 021100 104415
(4) 021102 062706 000014
4584 021106
(3) 021106
(3) 021106 104423
4585
4586
4587
4588
4589

PRINTX #FMT9,#DH4

PRINTX #FMT6,LUR14,LUR15,LUR16,LUR17

ENDMSG

MOV LUR11,-(SP)
MOV LUR10,-(SP)
MOV #FMT5,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #14,SP

MOV #DH4,-(SP)
MOV #FMT9,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #6,SP

MOV LUR17,-(SP)
MOV LUR16,-(SP)
MOV LUR15,-(SP)
MOV LUR14,-(SP)
MOV #FMT6,-(SP)
MOV #5,-(SP)
MOV SP,R0
TRAP C\$PNTX
ADD #14,SP

L10012: TRAP C\$MSG

4591
4592
4593
4594
4595
4596
4597
4598
4599
4600
4601
4602
4603
4604
4605
4606

021110
(3) 021110
021110
(3) 021110
(3) 021110 104425

.SBTTL REPORT CODING SECTION

:/ THE REPORT CODING SECTION CONTAINS THE
:/ 'PRINTS' CALLS THAT GENERATE STATISTICAL REPORTS.

BGNRPT

LSRPT::

FNRPT

L10013:

TRAP CSRPT

4608
4609
4610
4611
4612
4613
4614
4615
4616
4617
4618
4619
4620
4621
4622
4623
4624

021112
(3) 021112
021112 177777
021114 177777
021116 177777
021120

.SBTTL LOAD DEVICE PROTECTION TABLE

://
:/ THIS TABLE IDENTIFIES THE LOAD DEVICE TO THE SUPERVISOR, SO THAT IT CAN BE
:/ PROTECTED FROM TESTING, IF DESIRED.
://

BGNPROT

.WORD -1 ;DON'T CHK CSR ADRS
.WORD -1 ;DON'T CHK MASSBUS UNIT NO.
.WORD -1 ;DON'T CHK DRIVE NO.
ENDPROT

L\$PROT::

4626
 4627
 4628
 4629
 4630
 4631
 4632
 4633 021120
 (3) 021120
 4634
 4635 021120 010637 002346
 4636 021124 005037 002352
 4637 021130 005037 002426
 4638 021134 005037 002430
 4639 021140 005037 002420
 4640 021144 005037 002432
 4641 021150 005737 002412
 4642 021154 001007
 4643 021156 013737 000004 002414
 4644 021164 013737 000006 002416
 4645 021172 000406
 4646 021174 013737 002414 000004 6\$:
 4647 021202 013737 002416 000006
 4648 021210 012737 000001 002412 9\$:
 4649
 4650 021216
 (3) 021216 012700 000040
 (3) 021222 104447
 4651 021224
 (2) 021224 103415
 4652
 4653 021226
 (3) 021226 012700 000037
 (3) 021232 104447
 4654 021234
 (2) 021234 103411
 4655
 4656 021236
 (3) 021236 012700 000035
 (3) 021242 104447
 4657 021244
 (2) 021244 103411
 4658
 4659 021246
 (3) 021246 012700 000036
 (3) 021252 104447
 4660 021254
 (2) 021254 103504
 4661 021256 000414
 4662 021260
 4663 021260 005037 002444
 4664
 4665 021264 005037 002434
 4666 021270
 4667 021270 012737 177777 002344
 4668 021276 005237 002444

.SBTTL INITIALIZE SECTION

:/ THE INITIALIZE SECTION CONTAINS THE CODING THAT IS PERFORMED
 :/ AT THE BEGINNING OF THE TEST SEQUENCE ON THE NEXT UNIT.

BGNINIT

LSINIT::

```

MOV SP,PSTACK ;SAVE BASE-LEVEL STACK POINTER
CLR SUBRPC ;CLEAR SUBR CALL PC
CLR DISILO ;CLEAR CURRENT STATE OF DISSI
CLR CHPTYP ;CLEAR USYRT CHIP TYPE INDICATOR
CLR ERROR1 ;CLEAR ERROR FLAG
CLR SAVLEN ;CLEAR CHAR LENGTH FROM SETUP
TST FRSTIM ;SEE IF FIRST TIME THROUGH AFTER LOAD
BNE 6$ ;BR IF NOT
MOV @#4,SAVE4 ;SAVE ERROR TRAP VECTOR
MOV @#6,SAVE6
BR 9$
6$: MOV SAVE4,@#4 ;RESTORE ERROR TRAP VECTOR
MOV SAVE6,@#6
9$: MOV #1,FRSTIM ;MARK FLAG FOR NEXT TIME THROUGH
;SEE IF PROGRAM JUST STARTED, BR IF YES
READEF #EF.START
MOV #EF.START,R0
TRAP C$REFG
BCS STARST
;SEE IF PROGRAM JUST RESTARTED, BR IF YES
READEF #EF.RESTART
MOV #EF.RESTART,R0
TRAP C$REFG
BCS STARST
;SEE IF THIS IS A NEW PASS, BR IF YES
READEF #EF.NEW
MOV #EF.NEW,R0
TRAP C$REFG
BCS NEWST
;SEE IF PROGRAM WAS JUST CONTINUED
READEF #EF.CONTINUE
MOV #EF.CONTINUE,R0
TRAP C$REFG
BCS ENDIT
BR GETPRM
STARST: CLR STARES ;CLEAR FLAG TO SHOW JUST HAD STA OR RES
;CLEAR DEVICE MAP
CLR DEVMAP
NEWST: MOV #-1,LOGDEV ;RESET LOGICAL DEVICE TO -1
INC STARES ;INCR NO. OF PASSES SINCE STA OR RES
  
```

```
4669 021302 012737 000001 002436      MOV    #BIT0,DEVPTR      ;INIT DEVICE MAP BIT POINTER
4670                                     ; GET UNIBUS ADDRESS, VECTOR, PRIORITY LEVEL, SWITCH PACKS, TEST
4671                                     ; CONNECTOR INFORMATION FOR THIS LOGICAL DEVICE
4672 021310                                     GETPRM:
4673 021310 005237 002344      INC    LOGDEV           ;INCREMENT LOGICAL DEVICE NUMBER
4674 021314 023737 002344 002012      CMP    LOGDEV,LSUNIT    ;SEE IF MAXIMUM UNIT NO. EXCEEDED
4675 021322 002362                                     BGE   NEWST            ;BR IF YES
4676 021324                                     GPHARD LOGDEV,R1      ;GET P-TABLE POINTER INTO R1
(3) 021324 013700 002344                                     MOV    LOGDEV,RO
(3) 021330 104442                                     TRAP  CS$GPHRD
(3) 021332 010001                                     MOV    RO,R1
4677 021334                                     BCOMPLETE 10$        ;BR IF DEVICE AVAILABLE
(2) 021334 103403                                     BCS    10$
4678 021336 006337 002436      ASL    DEVPTR           ;SHIFT DEVICE POINTER
4679 021342 000762      BR     GETPRM          ;SKIP THIS DEVICE
4680 021344 053737 002436 002434 10$:  BIS    DEVPTR,DEVMAP   ;SET BIT FOR THIS DEVICE
4681 021352 006337 002436      ASL    DEVPTR           ;SHIFT BIT POINTER
4682 021356 062701 000002      ADD    #2,R1           ;INCREMENT R1 PAST MICROPROCESSOR TYPE
4683 021362 011137 002446      MOV    (R1),MPCSR      ;STORE POINTER TO MICROPROCESSOR CSR'S
4684 021366 011137 002450      MOV    (R1),BSEL1
4685 021372 005237 002450      INC    BSEL1           ;GET POINTER TO BSEL1 (MAINTENANCE REGISTER)
4686 021376 013737 002450 002452      MOV    BSEL1,BSEL2
4687 021404 005237 002452      INC    BSEL2           ;GET POINTER TO BSEL2
4688 021410 011137 002454      MOV    (R1),SEL4
4689 021414 062737 000004 002454      ADD    #4,SEL4        ;GET POINTER TO SEL4
4690 021422 012137 002456      MOV    (R1)+,SEL6
4691 021426 062737 000006 002456      ADD    #6,SEL6        ;STORE POINTER TO SEL6
4692 021434 011137 002460      MOV    (R1),MPIVEC    ;GET MICROPROCESSOR INPUT INTRPT VECTOR
4693 021440 012137 002462      MOV    (R1)+,MPOVEC
4694 021444 062737 000004 002462      ADD    #4,MPOVEC      ;GET MICROPROCESSOR OUTPUT INTRPT VECTOR
4695 021452 012137 002464      MOV    (R1)+,MPRIOR   ;GET MICROPROCESSOR DEVICE PRIORITY
4696 021456 062701 000014      ADD    #14,R1         ;POINT R1 TO RUN SWITCH INDICATOR
4697 021462 011137 002476      MOV    (R1),RUNINH    ;GET STATE OF MICROPROCESSOR RUN SWITCH
4698 021466      ENDIT:
4699 021466      ENDINIT
(3) 021466                                     L10015:
(3) 021466 104411                                     TRAP  CS$INIT
4700
4701
4702
4703
4704
4705
```

4707
4708
4709
4710
4711
4712
4713
(3)
4714
4715
(3)
(3)
4716
4717
4718
4719
4720
4721
4722
(3)
(3)
4723
4724
4725
(3)
(3)
4726
4727
4728
4729
4730

021470
021470 012700 000340
021474 104441
021476 012737 021520 000004
021504 012737 000340 000006
021512 005777 160730
021516 000405
021520 062706 000004
021524
021524 013700 002344
021530 104451
021532 013737 002414 000004
021540 013737 002416 000006
021546
021546 104461

```
.SBTTL AUTO DROP UNIT SECTION
://////
:/ THE AUTO DROP CODING DETERMINES WHETHER OR NOT THE DEVICE WHOSE P-TABLE
:/ WAS JUST OBTAINED IS READY FOR TESTING, AND IT IS DROPPED IF NOT READY.
://////
      BGNAUTO
      L$AUTO::
:ESTABLISH CPU PRIORITY = 7
      SETPRI #PRI07
      MOV #6$,@#4 ;SET UP NON-EXISTENT MEMORY ERROR TRAP VECTOR
      TRAP C$SPRI
      MOV #PRI07,@#6
      TST @MPCSR ;ADDRESS SELO
      BR 9$ ;TAKE THIS BR IF DEVICE RESPONDS
:COME HERE IF DEVICE CSR IS NON-EXISTENT
6$: ADD #4,SP ;CLEAN UP THE STACK POINTER
      DODU LOGDEV ;DROP THIS UNIT FROM TESTING
      MOV LOGDEV,RO
      TRAP C$DODU
9$: MOV SAVE4,@#4 ;RESTORE ERROR TRAP VECTOR
      MOV SAVE6,@#6
      ENDAUTO
      L10016:
      TRAP C$AUTO
```

4732
4733
4734
4735
4736
4737
4738
4739
4740
4741
4742
4743
4744
4745
4746
4747

.SBTTL CLEANUP CODING SECTION

:/
:/ THE CLEANUP CODING SECTION CONTAINS THE CODING THAT IS PERFORMED
:/ AT THE END OF THE TEST SEQUENCE ON A PARTICULAR UNIT.
:/

021550
(3) 021550

BGNCLN

L\$CLEAN::

021550
(3) 021550
(3) 021550 104412

ENDCLN

L10017: TRAP C\$CLEAN

4749
4750
4751
4752
4753
4754
4755
4756
(3)
4757
4758
(3)
4759
4760
(8)
(7)
(6)
(3)
(4)
(4)
4761
(3)
(3)
4762
4763

4764
4765
4766
4767
4768
4769

.SBTTL DROP UNIT SECTION

:/ THE DROP-UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
:/ TO NO LONGER BE TESTED.

BGNDU

LSDU::

:ISSUE UNIBUS RESET TO CLEAN JP
BRESET

TRAP C\$RESET

:PRINT 'UNIT XX DROPPED'
PRINTF #FMT27,LOGDEV

MOV LOGDEV,-(SP)
MOV #FMT27,-(SP)
MOV #2,-(SP)
MOV SP,R0
TRAP C\$PRINTF
ADD #6,SP

ENDDU

L10020:

TRAP C\$DU

021552 104433
021554 013746 002344
021560 012746 021602
021564 012746 000002
021570 010600
021572 104417
021574 062706 000006
021600 104453
021602 047045 040445 047125 FMT27: .ASCIZ /%N%UNIT %D2%A DROPPED%N/
021610 052111 022440 031104
021616 040445 042040 047522
021624 050120 042105 047045
021632 000
021634 .EVEN

4771
4772
4773
4774
4775
4776
4777
4778
4779
(3)
4780
(3)
(3)
4781
4782
4783
4784
4785
4786

021634
021634
021634
021634 104452

.SBTTL ADD UNIT SECTION

:/ THE ADD-UNIT SECTION CONTAINS THE CODING THAT CAUSES A DEVICE
:/ TO BE (A) TESTED FOR THE FIRST TIME, OR (B) RESUMED IN TESTING. IF
:/ 'EF.AUNIT' IS SET, THE UNIT WILL BE TESTED AS A NEW UNIT.

BGNAU

ENDAU

LSAU::

L10021:

TRAP CSAU

4788
4789
4790
4791
4792
4793
4794
4795
4796
4797
4798
4799
4800
4801
(3)
4802
4803
(3)
(3)
4804
4805
4806
4807
4808
4809
4810
4811
(4)
(5)
(5)
(5)
4812
4813
4814
(3)
(3)
4815
4816
4817
4818
4819
4820
4821
4822
4823
4824
4825
4826
(3)
4827
4828
4829
4830
4831
4832
4833

021636
021636
021636 012700 000340
021642 104441
021644 012737 021666 000004
021652 012737 000340 000006
021660 005777 160562
021664 000406
021666 062706 000004
021672
021672 104455
021674 000001
021676 012127
021700 014602
021702 013737 002414 000004
021710 013737 002416 000006
021716
021716 104401
021720
021720
021726 012737 000014 002400
021726 004737 003576
021732 004737 003672
021736 123737 002364 003026
021744 001416
021746 005037 002404
021752 113737 003026 002404

```
.SBTTL HARDWARE TESTS

:*****
.SBTTL TEST 1 - MICROPROCESSOR CSR ADDRESSING TEST (SELO)
:
:* THIS TEST ADDRESSES THE FIRST MICROPROCESSOR CSR (SELO), TO MAKE SURE
:* THAT A NON-EXISTENT MEMORY TIME-OUT TRAP DOES NOT OCCUR WHILE
:* ATTEMPTING TO ADDRESS THE MICROPROCESSOR.
:*****
BGNTST
                                T1::
;ESTABLISH CPU PRIORITY = 7
    SETPRI #PRI07
                                MOV #PRI07,R0
                                TRAP C$SPRI
;SET UP NON-EXISTENT MEMOR' ERROR TRAP VECTOR
    MOV #6$,@#4
    MOV #PRI07,@#6
    TST @MPCSR
    BR 9$
;ADDRESS SELO
;TAKE THIS BR IF DEVICE RESPONDS
;COME HERE IF DEVICE CSR IS NON-EXISTENT
6$: ADD #4,SP
;REPORT CSR ADDRESS TIME-OUT
    ERRDF 1,EM1,ERR1
                                TRAP C$ERDF
                                .WORD 1
                                .WORD EM1
                                .WORD ERR1
9$: MOV SAVE4,@#4
    MOV SAVE6,@#6
;RESTORE ERROR TRAP VECTOR
ENDTST
                                L10022:
                                TRAP C$SETST

:*****
.SBTTL TEST 2 - INBUS/OUTBUS REG 14 INITIALIZATION TEST
:
:* MASTER CLEAR (MCLR) IS SET IN THE MICROPROCESSOR, IBUS REG 14 IS READ
:* AND COMPARED TO 200.
:*****
BGNTST
                                T2::
    MOV #14,REGNUM
    JSR PC,MSTCLR
    JSR PC,READLU
    CMPB REDBYT,PATM+4
    BEQ 6$
    CLR GOODAT
    MOVB PATM+4,GOODAT
;SET LU REG NO. = 14
;ISSUE MASTER CLEAR
;READ REG 14
;CHK FOR INITIALIZED STATE
;BR IF YES
;SET EXPECTED REG CONTENTS = 000
;SET EXPECTED DATA
```

```
4834 021760 013737 002364 002406      MOV      REDBYT,BADDAT      ;SET ACTUAL REG CONTENTS
4835 021766 004737 004016      JSR      PC,GETREG        ;GET REGS FOR PRINTOUT
4836                                     ;REPORT REG NOT CLEARED BY MASTER CLEAR
4837 021772                                     ERRDF   2,EM2,ERR2
(4) 021772 104455                                     TRAP    C$ERDF
(5) 021774 000002                                     .WORD  2
(5) 021776 012163                                     .WORD  EM2
(5) 022000 014634                                     .WORD  ERR2
4838 022002                                     6$:
4839 022002                                     ENDTST
(3) 022002
(3) 022002 104401                                     L10023: TRAP    C$ETST
4840
4841
4842
4843
4844
4845
```

```
4846
4847
4848
4849
4850
4851
4852
4853
```

```
*****
:SBTTL      TEST 3 - INBUS/OUTBUS REG 14 READ/WRITE BIT TEST
:*
:* WRITE, READ, AND COMPARE ALL WORDS OF DATA PATTERN A INTO REG 14,
:* A BYTE AT A TIME. NON-R/W BITS ARE MASKED OFF TO 0 BEFORE WRITING AND
:* READING.
:* DATA PATTERN A = 125,252,000,377,001,002,004,010,020,040,100,200,376,
:*                   375,373,367,357,337,277,177.
*****
```

```
4854 022004                                     BGNTST
(3) 022004
4855 022004 004737 003576                                     T3::
4856 022010 012737 000014 002400      JSR      PC,MSTCLR        ;ISSUE MASTER CLEAR
4857 022016 012701 002571      MOV      #14,REGNUM      ;SET LU REG NO. = 14
4858 022022                                     MOV      #PATA,R1        ;GET POINTER TO DATA PAT IN R1
4859 022022                                     3$:
(3) 022022 104404                                     BGNSEG
4860 022024 111137 002366                                     TRAP    C$BSEG
4861 022030 143737 002560 002366      MOVB     (R1),WRIBYT      ;GET A BYTE OF PAT A
4862 022036 004737 003750      BICB     R14NRW,WRIBYT   ;MASK OFF NON-READ/WRITE BITS
4863 022042 004737 003672      JSR      PC,WRITLU       ;WRITE DATA BYTE INTO REG 14
4864 022046 143737 002560 002366      JSR      PC,READLU       ;READ DATA BYTE FROM REG 14
4865 022054 123737 002364 002366      BICB     R14NRW,REDBYT   ;MASK OFF NON-READ/WRITE BITS
4866 022062 001414      CMPB     REDBYT,WRIBYT   ;COMPARE BYTE READ TO BYTE WRITTEN
4867 022064 013737 002366 002404      BEQ      6$              ;BR IF BYTES MATCH
4868 022072 013737 002364 002406      MOV      WRIBYT,GOODAT   ;SET EXPECTED REG CONTENTS
4869 022100 004737 004016      MOV      REDBYT,BADDAT   ;SET ACTUAL REG CONTENTS
4870                                     JSR      PC,GETREG        ;GET REGS FOR PRINTOUT
4871                                     ;REPORT LINE UNIT REG MISCOMPARE
ERRDF   3,EM3,ERR2
```

```
(4) 022104 104455                                     TRAP    C$ERDF
(5) 022106 000003                                     .WORD  3
(5) 022110 012222                                     .WORD  EM3
(5) 022112 014634                                     .WORD  ERR2
4872 022114                                     6$:
4873 022114                                     ENDSEG
(3) 022114
(3) 022114 104405                                     10000$: TRAP    C$ESEG
4874 022116 005201
4875 022120 020127 002615      INC      R1
CMP     R1,#PATB      ;INCREMENT DATA PATTERN POINTER
;SEE IF ALL WORDS OF PATTERN A USED YET
```

4876 022124 103736
4877 022126
(3) 022126
(3) 022126 104401

ENDTST BLO 3\$:BR IF NOT DONE YET

L10024: TRAP C\$ETST

4878
4879
4880
4881
4882
4883
4884
4885
4886
4887

:SBTTL TEST 4 - REG 14 MASTER CLEAR TEST
:* WRITE 377 INTO REG 14, ISSUE MASTER CLEAR, READ REG 14 AND COMPARE
:* TO 200.

BGNTST

4888 022130
(3) 022130
4889 022130 004737 003576
4890 022134 012737 000014 002400
4891 022142 112737 000377 002366
4892 022150 004737 003750
4893 022154 004737 003576
4894 022160 004737 003672
4895 022164 123737 002364 003026
4896 022172 001416
4897 022174 005037 002404
4898 022200 113737 003026 002404
4899 022206 013737 002364 002406
4900 022214 004737 004016

JSR PC,MSTCLR :PERFORM MASTER CLEAR
MOV #14,REGNUM :SET LU REG NO. = 14
MOVB #377,WRIBYT :SET DATA BYTE = 377
JSR PC,WRITLU :WRITE 377 INTO REG 14
JSR PC,MSTCLR :ISSUE MASTER CLEAR
JSR PC,READLU :READ REG 14
CMPB REDBYT,PATM+4 :CHK FOR INIT'D STATE
BEQ 6\$:BR IF REG GOT CLEARED
CLR GOODAT
MOVB PATM+4,GOODAT :SET EXPECTED DATA
MOV REDBYT,BADDAT :SET ACTUAL DATA
JSR PC,GETREG :GET REGS FOR PRINTOUT
:REPORT REG NOT CLEARED BY MASTER CLEAR
ERRDF 2,EM2,ERR2

T4::

4901
4902 022220
(4) 022220 104455
(5) 022222 000002
(5) 022224 012163
(5) 022226 014634

TRAP C\$ERDF
.WORD 2
.WORD EM2
.WORD ERR2

4903 022230
4904 022230
(3) 022230
(3) 022230 104401

6\$:
ENDTST

L10025: TRAP C\$ETST

4905
4906
4907
4908
4909

4910
4911
4912
4913
4914

:SBTTL TEST 5 - REG 14 UNIBUS RESET (INIT) TEST
:* WRITE 377 INTO REG 14, ISSUE UNIBUS RESET (INIT), READ REG 14 AND COMPARE
:* TO 200.

BGNTST

4915 022232
(3) 022232
4916 022232 004737 003576
4917 022236 012737 000014 002400
4918 022244 112737 000377 002366
4919 022252 004737 003750
4920 022256
(3) 022256 104433

JSR PC,MSTCLR :PERFORM MASTER CLEAR
MOV #14,REGNUM :SET LU REG NO. = 14
MOVB #377,WRIBYT :SET DATA BYTE = 377
JSR PC,WRITLU :WRITE 377 INTO REG 14
BRESET :ISSUE UNIBUS RESET (INIT)

T5::

TRAP C\$RESET

```

4921 022260 142777 000200 160162      BICB  #RUN,@BSEL1      ;CLEAR RUN BIT
4922 022266 012701 000024      MOV   #20.,R1         ;INIT LOOP COUNTER
4923 022272 000240      2$:  NOP
4924 022274 005301      DEC   R1              ;DECR COUNTER
4925 022276 001375      BNE   2$              ;BR TO STALL
4926 022300 004737 003672      JSR   PC,READLU      ;READ REG 14
4927 022304 142737 000100 002364      BICB  #TXEN,REDBYT   ;CLEAR UNPREDICTABLE BIT
4928 022312 123737 002364 003026      CMPB  REDBYT,PATM+4  ;CHK FOR INIT'D STATE
4929 022320 001416      BEQ   6$              ;BR IF REG GOT CLEARED
4930 022322 005037 002404      CLR   GOODAT         ;SET EXPECTED DATA
4931 022326 113737 003026 002404      MOVB  PATM+4,GOODAT  ;SET ACTUAL DATA
4932 022334 013737 002364 002406      MOV   REDBYT,BADDAT ;SET ACTUAL DATA
4933 022342 004737 004016      JSR   PC,GETREG     ;GET REGS FOR PRINTOUT
4934                                     ;REPORT REG NOT CLEARED BY UNIBUS RESET (INIT)
4935                                     ERRDF 4,EM4,ERR2
(4) 022346 104455      TRAP  C$ERDF
(5) 022350 000004      .WORD 4
(5) 022352 012241      .WORD EM4
(5) 022354 014634      .WORD ERR2
4936 022356      6$:
4937 022356      ENDTST
(3) 022356
(3) 022356 104401      L10026: TRAP  C$ETST
4938
4939
4940
4941
4942
4943
4944
4945
4946
4947
4948
4949
4950
4951
4952
4953 022360
(3) 022360
4954 022360 004737 003576      JSR   PC,MSTCLR     ;ISSUE A MASTER CLEAR T6::
4955 022364 012737 000014 002400      MOV   #14,REGNUM   ;SET LU REG NO. = 14
4956 022372 013701 002400      MOV   REGNUM,R1    ;SET DESTINATION = OBUS* REG 14
4957 022376 052701 000100      BIS   #100,R1      ;SET SOURCE = BSEL4
4958 022402 052701 121000      BIS   #MVIXOX,R1   ;SET REST OF MOVE INSTRUCTION
4959 022406 010137 022424      MOV   R1,2$        ;SET INSTRUCTION AS SUBROUTINE ARGUMENT
4960 022412 112777 000041 160034      MOVB  #041,@BSEL4  ;SET DATA BYTE = 041
4961 022420 004737 003540      JSR   PC,STPCLK    ;EXECUTE MOVE INSTRUCTION
4962 022424 000000      2$: .WORD 0          ;INSTRUCTION GOES HERE
4963 022426 004737 003672      JSR   PC,READLU    ;READ LU REG 14
4964 022432 123737 002364 003026      CMPB  REDBYT,PATM+4 ;CHECK FOR LU REG 14 UNCHANGED
4965 022440 001415      BEQ   4$            ;BR IF LU REG 14 UNCHANGED
4966 022442 005037 002404      CLR   GOODAT       ;SET EXPECTED DATA
4967 022446 113737 003026 002404      MOVB  PATM+4,GOODAT ;SET ACTUAL DATA
4968 022454 013737 002364 002406      MOV   REDBYT,BADDAT ;SET ACTUAL DATA
4969 022462 004737 004016      JSR   PC,GETREG    ;GET REGS FOR PRINTOUT
    
```

```

:*****
:SBTTL      TEST 6 - LINE UNIT FALSE SELECTION TEST
:*
:* FIRST, A MASTER CLEAR IS PERFORMED. THEN, THE PROGRAM SINGLE-STEPS THE
:* MICROPROCESSOR THROUGH AN INSTRUCTION WHICH LOADS 041 (OCT) INTO THE MAR
:* REGISTER (OBUS* ADRS 14). THEN, THE LINE UNIT REGISTER 14 IS READ AND CHECKED
:* TO BE UNAFFECTED (STILL = 0). THIS TEST IS INTENDED TO DETECT A FALSE
:* SELECTION OF THE LINE UNIT REGISTERS, WHEN THE LINE UNIT IS NOT BEING
:* ACCESSED.
:*****
    
```

```

BGNTST
T6::
JSR   PC,MSTCLR     ;ISSUE A MASTER CLEAR
MOV   #14,REGNUM   ;SET LU REG NO. = 14
MOV   REGNUM,R1    ;SET DESTINATION = OBUS* REG 14
BIS   #100,R1      ;SET SOURCE = BSEL4
BIS   #MVIXOX,R1   ;SET REST OF MOVE INSTRUCTION
MOV   R1,2$        ;SET INSTRUCTION AS SUBROUTINE ARGUMENT
MOVB  #041,@BSEL4  ;SET DATA BYTE = 041
JSR   PC,STPCLK    ;EXECUTE MOVE INSTRUCTION
2$: .WORD 0          ;INSTRUCTION GOES HERE
JSR   PC,READLU    ;READ LU REG 14
CMPB  REDBYT,PATM+4 ;CHECK FOR LU REG 14 UNCHANGED
BEQ   4$            ;BR IF LU REG 14 UNCHANGED
CLR   GOODAT       ;SET EXPECTED DATA
MOVB  PATM+4,GOODAT ;SET ACTUAL DATA
MOV   REDBYT,BADDAT ;SET ACTUAL DATA
JSR   PC,GETREG    ;GET REGS FOR PRINTOUT
    
```

```
4970 ;REPORT REGISTER MISCOMPARE
4971 022466 ERRDF 3,EM3,ERR2
(4) 022466 104455 TRAP C$ERDF
(5) 022470 000003 .WORD 3
(5) 022472 012222 .WORD EM3
(5) 022474 014634 .WORD ERR2
4972 022476
4973 022476 4$:
(3) 022476 ENDTST
(3) 022476 104401 L10027: TRAP C$ETST
4974
4975
4976
4977
4978
4979
```

```
4980 ;*****
4981 .SBTTL TEST 7 - INBUS REG MASTER CLEAR TEST
4982 ;*
4983 ;* FIRST, ALL READ/WRITE BITS OF REGS 10-17 ARE SET BY LOADING A
4984 ;* DIFFERENT WORD OF PATTERN G INTO EACH REG. THEN,
4985 ;* A MASTER CLEAR IS ISSUED AND EACH REG IS READ AND COMPARED TO A WORD OF
4986 ;* PATTERN M, WHICH CONTAINS THE INITIALIZED STATES OF THE REGS. (UNPREDICTABLE
4987 ;* BITS ARE MASKED OFF TO 0 BEFORE COMPARISON).
4988 ;* PATTERN G = 000,000,240,120,177,000,000,001
4989 ;* PATTERN M = 000,020,000,000,200,000,000,051
4990 ;*****
4990 022500 BGNST
```

```
4991 (3) 022500 T7::
4991 022500 004737 003576 JSR PC,MSTCLR ;ISSUE MASTER CLEAR
4992 022504 012737 000010 002400 MOV #10,REGNUM ;INIT REG NO. TO 10
4993 022512 012701 002644 MOV #PATG,R1 ;INIT DATA PATTERN POINTER
4994 022516 112137 002366 2$: MOVB (R1)+,WRIBYT ;SET DATA PATTERN BYTE TO BE WRITTEN
4995 022522 004737 003750 JSR PC,WRITLU ;WRITE BYTE INTO REG
4996 022526 005237 002400 INC REGNUM ;INCREMENT REG NO. FOR WRITING
4997 022532 020127 002654 CMP R1,#PATH ;SEE IF ALL BYTES WRITTEN YET
4998 022536 103767 BLO 2$ ;BR IF NOT DONE WRITING YET
4999 022540 004737 003576 JSR PC,MSTCLR ;ISSUE MASTER CLEAR
5000 022544 012737 000010 002400 MOV #10,REGNUM ;INIT LU REG NO. TO 10
5001 022552 012702 003022 MOV #PATM,R2 ;INIT DATA PATTERN POINTER
5002 022556 012701 002550 MOV #LUPBITS,R1 ;INIT POINTER TO UNPREDICTABLE BITS
5003 022562 3$:
5004 022562 004737 003672 JSR PC,READLU ;READ A LINE UNIT REG
5005 022566 142137 002364 BICB (R1)+,REDBYT ;MASK OUT UNPREDICTABLE BITS FOR THIS REG
5006 022572 123712 002364 CMPB REDBYT,(R2) ;COMPARE MASKED DATA TO EXPECTED
5007 022576 001417 BEQ 6$ ;BR IF DATA READ IS OK
5008 022600 005037 002404 CLR GOODAT
5009 022604 111237 002404 MOVB (R2),GOODAT ;SET EXPECTED DATA
5010 022610 013737 002364 002406 MOV REDBYT,BADDAT ;SET ACTUAL DATA
5011 022616 004737 004016 JSR PC,GETREG ;GET REGS FOR PRINTOUT
5012 ;REPORT REG NOT CLEARED BY MASTER CLEAR
5013 022622 ERRDF 2,EM2,ERR2
(4) 022622 104455 TRAP C$ERDF
(5) 022624 000002 .WORD 2
(5) 022626 012163 .WORD EM2
(5) 022630 014634 .WORD ERR2
5014 022632 ESCAPE TST
```



```
(4) 023052 104455
(5) 023054 000003
(5) 023056 012222
(5) 023060 014634
5066 023062          ESCAPE TST
(3) 023062 104410
(3) 023064 000022
5067 023066 005203          12$: INC R3          ;INCREMENT DATA PATTERN INDEX FOR READING
5068 023070 020327 000010    CMP R3,#10        ;SEE IF ALL REGS READ YET
5069 023074 002724          BLT 9$           ;BR IF NOT
5070 023076 005201          INC R1          ;INCREMENT DATA PAT INDEX FOR WRITING
5071 023100 020127 000010    CMP R1,#10        ;SEE IF ALL REGS WRITTEN YET
5072 023104 002677          BLT 6$           ;BR IF NOT
5073 023106
(3) 023106
(3) 023106 104401          L10031: TRAP C$SETST
5074
5075
5076
5077
5078
5079
5080 :*****
5081 :SBTTL TEST 9 - REG 11 READ/WRITE BIT TEST
5082 :*
5083 :* WRITE, READ, AND COMPARE EACH WORD OF DATA PATTERN C INTO REG 11 :
5084 :* DATA PATTERN C = 020,020,020.
5085 :*****
5085 023110          BGNTST
(3) 023110
5086 023110 004737 003576          JSR PC,MSTCLR      ;ISSUE MASTER CLEAR T9::
5087 023114 012737 000011 002400    MOV #11,REGNUM     ;SET LU REG NO. = 11
5088 023122 012701 002625          MOV #PATC,R1      ;GET POINTER TO DATA PAT IN R1
5089 023126
5090 023126          3$: BGNSEG
(3) 023126 104404
5091 023130 111137 002366          MOVB (R1),WRIBYT   ;GET A BYTE OF PAT C TRAP C$BSEG
5092 023134 004737 003750          JSR PC,WRITLU     ;WRITE DATA BYTE INTO REG 11
5093 023140 004737 003672          JSR PC,READLU     ;READ DATA BYTE FROM REG 11
5094 023144 143737 002551 002364    BICB UPBITS+1,REDBYT ;MASK OUT UNPREDICTABLE BITS
5095 023152 123737 002364 002366    CMPB REDBYT,WRIBYT ;COMPARE BYTE READ TO BYTE WRITTEN
5096 023160 001414          BEQ 6$           ;BR IF BYTES MATCH
5097 023162 013737 002366 002404    MOV WRIBYT,GOODAT ;SET EXPECTED REG CONTENTS
5098 023170 013737 002364 002406    MOV REDBYT,BADDAT ;SET ACTUAL REG CONTENTS
5099 023176 004737 004016          JSR PC,GETREG     ;GET REGS FOR PRINTOUT
5100 :REPORT LINE UNIT REG MISCOMPARE
5101 023202          ERRDF 3,EM3,ERR2
(4) 023202 104455
(5) 023204 000003
(5) 023206 012222
(5) 023210 014634
5102 023212          6$:
5103 023212          ENDSEG
(3) 023212
(3) 023212 104405          10000$: TRAP C$ESEG
5104 023214 005201          INC R1          ;INCREMENT DATA PATTERN POINTER
5105 023216 020127 002630    CMP R1,#PATD     ;SEE IF ALL WORDS OF PATTERN C USED YET
```

5106 023222 103741
5107 023224
(3) 023224
(3) 023224 104401
5108
5109
5110
5111
5112
5113
5114
5115
5116
5117
5118
5119 023226
(3) 023226
5120 023226 004737 003576
5121 023232 012737 000012 002400
5122 023240 012701 002630
5123 023244
5124 023244
(3) 023244 104404
5125 023246 111137 002366
5126 023252 004737 003750
5127 023256 004737 003672
5128 023262 143737 002552 002364
5129
5130 023270 123737 002364 002366
5131 023276 001414
5132 023300 013737 002366 002404
5133 023306 013737 002364 002406
5134 023314 004737 004016
5135
5136 023320
(4) 023320 104455
(5) 023322 000003
(5) 023324 012222
(5) 023326 014634
5137 023330
5138 023330
(3) 023330
(3) 023330 104405
5139 023332 005201
5140 023334 020127 002633
5141 023340 103741
5142 023342
(3) 023342
(3) 023342 104401
5143
5144
5145
5146
5147
5148
5149

```
ENDTST BLO 3$ ;BR IF NOT DONE YET
L10032: TRAP C$ETST
*****
:SBTTL TEST 10 - REG 12 READ/WRITE BIT TEST
:
:* WRITE, READ, AND COMPARE EACH WORD OF DATA PATTERN D INTO REG 12 :
:* DATA PATTERN D = 000,040,000.
:*****
BGNTST
T10::
JSR PC,MSTCLR ;ISSUE MASTER CLEAR
MOV #12,REGNUM ;SET LU REG NO. = 12
MOV #PATD,R1 ;GET POINTER TO DATA PAT IN R1
3$:
BGNSEG TRAP C$BSEG
MOVB (R1),WRIBYT ;GET A BYTE OF PAT D
JSR PC,WRITLU ;WRITE DATA BYTE INTO REG 12
JSR PC,READLU ;READ DATA BYTE FROM REG 12
BICB UPBITS+2,REDBYT ;MASK OUT UNPREDICTABLE BITS
CMPB REDBYT,WRIBYT ;COMPARE BYTE READ TO BYTE WRITTEN
BEQ 6$ ;BR IF BYTES MATCH
MOV WRIBYT,GOODAT ;SET EXPECTED REG CONTENTS
MOV REDBYT,BADDAT ;SET ACTUAL REG CONTENTS
JSR PC,GETREG ;GET REGS FOR PRINTOUT
:REPORT LINE UNIT REG MISCOMPARE
ERRDF 3,EM3,ERR2
6$:
ENDSEG
10000$: TRAP C$ESEG
INC R1 ;INCREMENT DATA PATTERN POINTER
CMP R1,#PATE ;SEE IF ALL WORDS OF PATTERN D USED YET
BLO 3$ ;BR IF NOT DONE YET
ENDTST
L10033: TRAP C$ETST
*****
:SBTTL TEST 11 - REG 13 READ/WRITE BIT TEST
```

```
5150
5151
5152
5153
5154 023344
(3) 023344
5155 023344 004737 003576
5156 023350 012737 000013 002400
5157 023356 012701 002633
5158 023362
5159 023362
(3) 023362 104404
5160 023364 111137 002366
5161 023370 004737 003750
5162 023374 004737 003672
5163 023400 143737 002553 002364
5164 023406 123737 002364 002366
5165 023414 001414
5166 023416 013737 002366 002404
5167 023424 013737 002364 002406
5168 023432 004737 004016
5169
5170 023436
(4) 023436 104455
(5) 023440 000003
(5) 023442 012222
(5) 023444 014634
5171 023446
5172 023446
(3) 023446
(3) 023446 104405
5173 023450 005201
5174 023452 020127 002641
5175 023456 103741
5176 023460
(3) 023460
(3) 023460 104401
5177
5178
5179
5180
5181
5182
5183
5184
5185
5186
5187
5188 023462
(3) 023462
5189 023462 004737 003576
5190 023466 012737 000017 002400
5191 023474 012701 002641
```

```

:
:
: * WRITE, READ, AND COMPARE EACH WORD OF DATA PATTERN E INTO REG 13 :
: * DATA PATTERN E = 000,120,020,100,120,000.
:
:*****
BGNTST
T11::
JSR PC,MSTCLR ;ISSUE MASTER CLEAR
MOV #13,REGNUM ;SET LU REG NO. = 13
MOV #PATE,R1 ;GET POINTER TO DATA PAT IN R1
3$:
BGNSEG
TRAP CSBSEG
MOVB (R1),WRIBYT ;GET A BYTE OF PAT E
JSR PC,WRITLU ;WRITE DATA BYTE INTO REG 13
JSR PC,READLU ;READ DATA BYTE FROM REG 13
BICB UPBITS+3,REDBYT ;MASK OUT UNPREDICTABLE BITS
CMPB REDBYT,WRIBYT ;COMPARE BYTE READ TO BYTE WRITTEN
BEQ 6$ ;BR IF BYTES MATCH
MOV WRIBYT,GOODAT ;SET EXPECTED REG CONTENTS
MOV REDBYT,BADDAT ;SET ACTUAL REG CONTENTS
JSR PC,GETREG ;GET REGS FOR PRINTOUT
:REPORT LINE UNIT REG MISCOMPARE
ERRDF 3,EM3,ERR2
TRAP C$ERDF
.WORD 3
.WORD EM3
.WORD ERR2
6$:
ENDSEG
10000$:
TRAP C$ESEG
INC R1 ;INCREMENT DATA PATTERN POINTER
CMP R1,#PATF ;SEE IF ALL WORDS OF PATTERN E USED YET
BLO 3$ ;BR IF NOT DONE YET
ENDTST
L10034:
TRAP C$ETST
:
:*****
:SBTTL TEST 12 - REG 17 READ/WRITE BIT TEST
:
:
: * WRITE, READ, AND COMPARE EACH WORD OF DATA PATTERN F INTO REG 17 :
: * DATA PATTERN F = 050,051,050.
:
:*****
BGNTST
T12::
JSR PC,MSTCLR ;ISSUE MASTER CLEAR
MOV #17,REGNUM ;SET LU REG NO. = 17
MOV #PATF,R1 ;GET POINTER TO DATA PAT IN R1
```

```
5193 023500          3$:          BGNSEG
5194 023500
(3) 023500 104404
5195 023502 111137 002366      MOVB    (R1),WRIBYT      ;GET A BYTE OF PAT F          TRAP   C$BSEG
5196 023506 004737 003750      JSR     PC,WRITLU      ;WRITE DATA BYTE INTO REG 17
5197 023512 004737 003672      JSR     PC,READLU     ;READ DATA BYTE FROM REG 17
5198 023516 143737 002557 002364  BICB    UPBITS+7,REDBYT ;MASK OUT UNPREDICTABLE BITS
5199 023524 123737 002364 002366  CMPB    REDBYT,WRIBYT  ;COMPARE BYTE READ TO BYTE WRITTEN
5200 023532 001414      BEQ     6$            ;BR IF BYTES MATCH
5201 023534 013737 002366 002404  MOV     WRIBYT,GOODAT  ;SET EXPECTED REG CONTENTS
5202 023542 013737 002364 002406  MOV     REDBYT,BADDAT  ;SET ACTUAL REG CONTENTS
5203 023550 004737 004016      JSR     PC,GETREG     ;GET REGS FOR PRINTOUT
5204
5205 023554          ;REPORT LINE UNIT REG MISCOMPARE
(4) 023554 104455      ERRDF   3,EM3,ERR2
(5) 023556 000003          TRAP   C$ERDF
(5) 023560 012222          .WORD  3
(5) 023562 014634          .WORD  EM3
                                .WORD  ERR2
5206 023564          6$:          ENDSEG
5207 023564
(3) 023564
(3) 023564 104405          10000$: TRAP   C$ESEG
5208 023566 005201          INC     R1            ;INCREMENT DATA PATTERN POINTER
5209 023570 020127 002644      CMP     R1,#PATG     ;SEE IF ALL WORDS OF PATTERN F USED YET
5210 023574 103741          BLO    3$            ;BR IF NOT DONE YET
5211 023576          ENDTST
(3) 023576
(3) 023576 104401          L10035: TRAP   C$ETST
```

```
5212
5213
5214
5215
5216
5217
5218 :*****
5219 :SBTTL      TEST 13 - AINTENANCE CLOCK BIT TEST
5220 :*
5221 :* FIRST, A MASTER CLEAR IS ISSUED TO INIT ALL REGS. THEN, THE MICROPROCESSOR
5222 :* IS PLACED IN A LOOP ON AN INSTRUCTION, BY SETTING THE INSTRUCTION IN SEL6
5223 :* AND SETTING ROMI AND RUN IN BSEL1. THE INSTRUCTION IS ONE WHICH REPETITIVELY
5224 :* READS LINE UNIT REG 17 INTO BSEL2. THE PDP-11 CAN THEN SCAN BSEL2 TO MONITOR
5225 :* THE MAINTENANCE CLOCK BIT, MCLK. THE FOLLOWING SEQUENCE IS THEN PERFORMED
5226 :* TO MONITOR MCLK :
5227 :* - THE PROGRAM REPEATEDLY CHECKS THE MCLK BIT FOR THE 1 STATE, AND IF IT IS
5228 :* NOT FOUND WITHIN SEVERAL HUNDRED MILLI-SEC (DEPENDING ON THE PROCESSOR)
5229 :* AN ERROR IS REPORTED. (THE MAINTENANCE CLOCK HAS A PERIOD OF 41.6 MICRO-
5230 :* SEC).
5231 :* - THE PROGRAM NEXT REPEATEDLY CHECKS THE MCLK BIT FOR THE 0 STATE, AND IF
5232 :* IT IS NOT FOUND WITHIN SEVERAL HUNDRED MILLI-SEC AN ERROR IS REPORTED.
5233 :* - THE PROGRAM NEXT REPEATEDLY CHECKS MCLK BIT FOR THE 1 STATE AGAIN, AND
5234 :* IF IT IS NOT FOUND WITHIN SEVERAL HUNDRED MILLI-SEC, AN ERROR IS REPORTED.
5235 :*
5236 :* IF THE P-TABLE FOR THIS UNIT INDICATES THAT THE MICROPROCESSOR RUN SWITCH
5237 :* IS NOT ON, THE TEST WILL BE SKIPPED.
5238 :*****
(3) 023600      BGNSTST
(3) 023600
```

```
5239 023600 004737 003576 JSR PC,MSTCLR ;PERFORM MASTER CLEAR
5240 023604 012737 000015 002442 MOV #13,TSTNUM ;SET TEST NO.
5241 023612 005737 002476 TST RUNINH ;SEE IF RUN SWITCH IS SET
5242 023616 001020 BNE 1$ ;BR IF YES, TO RUN TEST
5243 023620 023727 002444 000001 CMP STARES,#1 ;SEE IF THIS IS FIRST PASS SINCE STA OR RES
5244 023626 001012 BNE 40$ ;BR IF NOT, TO SKIP PRINTING
5245 023630 PRINTF #FMT19,TSTNUM ;PRINT MSG TO SAY TEST NOT RUN
(8) 023630 013746 002442 MOV TSTNUM,-(SP)
(7) 023634 012746 012007 MOV #FMT19,-(SP)
(6) 023640 012746 000002 MOV #2,-(SP)
(3) 023644 010600 MOV SP,R0
(4) 023646 104417 TRAP C$PNTF
(4) 023650 062706 000006 ADD #6,SP
5246 023654 000137 024222 40$: JMP A1 ;GO TO SKIP TEST
5247 023660 012737 000017 002400 1$: MOV #17,REGNUM ;SET REG NO. = 17
5248 023666 012701 000360 MOV #360,R1 ;SET INSTRUCTION SOURCE = INBUS REG 17
5249 023672 052701 000002 BIS #2,R1 ;SET INSTRUCTION DESTINATION = BSEL2
5250 023676 052701 021000 BIS #MVIOX,R1 ;SET REST OF MOVE INSTRUCTION
5251 023702 010137 023712 MOV R1,2$ ;SET INSTRUCTION AS SUBROUTINE ARGUMENT
5252 023706 004737 004074 JSR PC,LOOPIN ;GET MICROPROCESSOR LOOPING ON MOVE INSTRUCTION
5253 023712 000000 2$: .WORD 0 ;INSTRUCTION GOES HERE
5254
5255
5256 ;-----
5257 ; WAIT FOR MCLK BIT TO BE SET TO 1
5258 023714 005037 002510 CLR REG0 ;INIT PROGRAM TIMER
5259 023720 117737 156526 002364 3$: MOVB @BSEL2,REDBYT ;GET RLG 17 INTO REDBYT
5260 023726 132737 000002 002364 BITB #MCLK,REDBYT ;SEE IF MCLK BIT = 1 YET
5261 023734 001031 BNE 6$ ;BR IF MCLK = 1
5262 023736 005237 002510 INC REG0 ;INCREMENT TIMER
5263 023742 001366 BNE 3$ ;BR IF PROGRAM TIMER DID NOT TIME OUT YET
5264 ; (TIME OUT = SEVERAL HUNDRED MILLI-SEC)
5265 023744 012737 000002 002404 MOV #MCLK,GOODAT ;SET EXPECTED REG CONTENTS
5266 023752 013737 002364 002406 MOV REDBYT,BADDAT ;SET ACTUAL REG CONTENTS
5267 023760 004737 004016 JSR PC,GETREG ;GET REGS FOR PRINTOUT
5268 ;REPORT MCLK BIT STUCK AT 0
5269 023764 ERRDF 5,EMS,ERR2
(4) 023764 104455 TRAP C$ERDF
(5) 023766 000005 .WORD 5
(5) 023770 012314 .WORD EMS
(5) 023772 014634 .WORD ERR2
5270 ;TYPE 'PLEASE INSURE MICROPROCESSOR RUN SWITCH IS ON'
5271 023774 PRINTF #FMT24
(7) 023774 012746 012040 MOV #FMT24,-(SP)
(6) 024000 012746 000001 MOV #1,-(SP)
(3) 024004 010600 MOV SP,R0
(4) 024006 104417 TRAP C$PNTF
(4) 024010 062706 000004 ADD #4,SP
5272 024014 000137 024222 JMP A1 ;ESCAPE TO END OF TEST
5273
5274 ;-----
5275 ; WAIT FOR MCLK BIT TO BE CLEARED TO 0
5276 ;-----
5277 024020 6$:
5278 024020 005037 002510 CLR REG0 ;INIT PROGRAM TIMER
5279 024024 117737 156422 002364 8$: MOVB @BSEL2,REDBYT ;GET REG 17 INTO REDBYT
```


5316
5317
5318
5319
5320
5321
5322
5323
5324
5325
5326
5327
5328
5329
5330
5331
5332
5333
5334
5335
5336
5337
5338
5339
5340
5341
5342
5343
5344
5345
5346
5347
5348
5349
5350
5351
5352
5353
5354
5355
5356
5357
5358

024230
(3) 024230
024230 004737 003576
024234 005037 002402
024240 012701 002654
024244 112137 002374
024250 112137 002376
024254 004737 004312
024260 032737 000002 002420
024266 001413
024270 012737 000014 002400
024276 004737 004016

024302
(4) 024302 104455
(5) 024304 000064
(5) 024306 014107
(5) 024310 020632
024312
(3) 024312 104410
(3) 024314 000244
024316 062737 000002 002402
024324 020127 002664
024330 103745
024332 004737 003576
024336 005037 002402
024342 012701 002664
024346 004737 004124
024352 032737 000001 002420
024360 001413
024362 012737 000014 002400
024370 004737 004016

024374
(4) 024374 104455
(5) 024376 000065
(5) 024400 014150
(5) 024402 020632
024404
(3) 024404 104410
(3) 024406 000152
024410 023727 002402 000006
024416 001003
024420 142737 000372 002370

```
*****  
:SBTTL TEST 14 - EXTENDED REGISTER MASTER CLEAR TEST  
:  
:* FIRST, ALL READ/WRITE BITS OF EXTENDED REGS AX0-AX3 ARE SET BY LOADING  
:* A DIFFERENT WORD OF PATTERN H INTO EACH REG. THEN, A MASTER CLEAR IS  
:* ISSUED AND EACH REG IS READ AND COMPARED TO A WORD OF PATTERN I, WHICH  
:* CONTAINS THE INITIALIZED STATES OF ALL THE EXTENDED REGS.  
:* PATTERN H = 000,000,377,017,377,377,375,377  
:* PATTERN I = 000,000,000,000,000,103,000,000  
:*****  
BGNTST  
  
T14::  
JSR PC,MSTCLR ;ISSUE AN INITIAL MASTER CLEAR  
CLR AXNUM ;INIT AX REG BYTE NO. TO 0  
MOV #PATH,R1 ;INIT DATA PATTERN POINTER  
1$: MOV (R1)+,WAX15 ;SET BITS TO LOAD INTO LO BYTE  
MOV (R1)+,WAX16 ;SET BITS TO LOAD INTO HI BYTE  
JSR PC,WRITAX ;WRITE EXTENDED REG  
BIT #WRDYTO,ERROR1 ;SEE IF READY FAILED TO SET  
BEQ 8$ ;BR IF NOT  
MOV #14,REGNUM ;SET LU REG NO = 14  
JSR PC,GETREG ;GET REGS FOR PRINTOUT  
;REPORT READY NOT SET AFTER AX REG WRITE  
ERRDF 52,EM52,ERR9  
  
TRAP C$ERDF  
.WORD 52  
.WORD EM52  
.WORD ERR9  
  
ESCAPE TST  
TRAP C$ESCAPE  
.WORD L10037-.  
  
8$: ADD #2,AXNUM ;INCR REG BYTE NO.  
CMP R1,#PATI ;SEE IF ALL REGS WRITTEN YET  
BLO 1$ ;BR IF NOT DONE WRITING YET  
JSR PC,MSTCLR ;ISSUE MASTER CLEAR  
CLR AXNUM ;INIT EXTENDED REG BYTE NO. TO 0  
MOV #PATI,R1 ;INIT DATA PAT POINTER FOR READS  
2$: JSR PC,READAX ;READ AN EXTENDED REG  
BIT #RRDYTO,ERROR1 ;SEE IF READY FAILED TO SET  
BEQ 10$ ;BR IF NOT  
MOV #14,REGNUM ;SET LU REG NO. = 14  
JSR PC,GETREG ;GET REGS FOR PRINTOUT  
;REPORT READY NOT SET AFTER AX REG READ  
ERRDF 53,EM53,ERR9  
  
TRAP C$ERDF  
.WORD 53  
.WORD EM53  
.WORD ERR9  
  
ESCAPE TST  
TRAP C$ESCAPE  
.WORD L10037-.  
  
10$: CMP AXNUM,#6 ;SEE IF AX3-15  
BNE 3$ ;BR IF NOT  
BICB #AX315U,RAX15 ;MASK OFF UNPREDICTABLE BITS
```

```
5359 024426 123711 002370 3$: CMPB RAX15,(R1) ;COMPARE LO BYTE TO EXPECTED VALUE
5360 024432 001417 BEQ 4$ ;BR IF DATA MATCHES
5361 024434 005037 002404 CLR GOODAT
5362 024440 111137 002404 MOV (R1),GOODAT ;GET EXPECTED DATA BYTE
5363 024444 013737 002370 002406 MOV RAX15,BADDAT ;GET ACTUAL DATA BYTE
5364 024452 004737 004526 JSR PC,GETALL ;GET REGS FOR PRINTOUT
5365 ;REPORT REG NOT INITIALIZED BY MASTER CLEAR
5366 024456 ERRDF 2,EM2,ERR3
(4) 024456 104455 TRAP C$ERDF
(5) 024460 000002 .WORD 2
(5) 024462 012163 .WORD EM2
(5) 024464 015142 .WORD ERR3
5367 024466 ESCAPE TST TRAP C$ESCAPE
(3) 024466 104410 .WORD L10037-.
(3) 024470 000070
5368 024472 005237 002402 4$: INC AXNUM ;INCREMENT AX BYTE NO.
5369 024476 005201 INC R1 ;INCREMENT PAT POINTER
5370 024500 123711 002372 CMPB RAX16,(R1) ;COMPARE HI BYTE TO EXPECTED VALUE
5371 024504 001417 BEQ 6$ ;BR IF DATA MATCHES
5372 024506 005037 002404 CLR GOODAT
5373 024512 111137 002404 MOV (R1),GOODAT ;GET EXPECTED DATA BYTE
5374 024516 013737 002372 002406 MOV RAX16,BADDAT ;GET ACTUAL DATA BYTE
5375 024524 004737 004526 JSR PC,GETALL ;GET REGS FOR PRINTOUT
5376 ;REPORT REG NOT INITIALIZED BY MASTER CLEAR
5377 024530 ERRDF 2,EM2,ERR3
(4) 024530 104455 TRAP C$ERDF
(5) 024532 000002 .WORD 2
(5) 024534 012163 .WORD EM2
(5) 024536 015142 .WORD ERR3
5378 024540 ESCAPE TST TRAP C$ESCAPE
(3) 024540 104410 .WORD L10037-.
(3) 024542 000016
5379 024544 005237 002402 6$: INC AXNUM ;INCR AX BYTE NO.
5380 024550 005201 INC R1 ;INCR PAT POINTER
5381 024552 020127 002674 CMP R1,#PATJ ;SEE IF ALL REGS READ YET
5382 024556 103673 BLO 2$ ;BR IF NOT DONE READING YET
5383 024560 ENDTST
(3) 024560 L10037: TRAP C$ETST
(3) 024560 104401
5384
5385
5386
5387
5388
5389
5390 ;*****
5391 .SBTTL TEST 15 - EXTENDED REGISTER ADDRESSING TEST
5392 ;*
5393 ;* FIRST, ISSUE A MASTER CLEAR TO PUT REGS INTO INITIALIZED STATES SHOWN IN
5394 ;* PATTERN I. THEN, WRITE A DIFFERENT WORD OF PATTERN J INTO EACH EXTENDED (AX)
5395 ;* REG, AND AFTER EACH WRITE, READ AND COMPARE ALL EXTENDED REGS TO EXPECTED
5396 ;* VALUES.
5397 ;* PATTERN I = 000,000,000,000,000,103,000,000
5398 ;* PATTERN J = 000,000,010,002,004,103,001,100
5399 ;*****
(3) 024562 BGNTST
(3) 024562
```



```

5400 024562 004737 003576      JSR    PC,MSTCLR      ;ISSUE MASTER CLEAR
5401 024566 012701 002664      MOV    #PATI,R1      ;INIT POINTER TO PAT I
5402 024572 012702 002500      MOV    #REDDAT,R2    ;INIT POINTER TO EXPECTED DATA AREA
5403 024576 112122              3$:  MOVB   (R1)+,(R2)+  ;MOVE PAT I INTO REDDAT TABLE
5404 024600 020127 002674      CMP    R1,#PATJ
5405 024604 103774              BLO   3$
5406 024606 005001              CLR   R1              ;INIT INDEX FOR WRITING
5407 024610 010137 002402      6$:  MOV    R1,AXNUM      ;GET AX BYTE NO. FOR WRITING
5408 024614 116137 002674 002374  MOVB   PATJ(R1),WAX15 ;SET LO AX BYTE TO BE WRITTEN
5409 024622 116137 002675 002376  MOVB   PATJ+1(R1),WAX16 ;SET HI AX BYTE TO BE WRITTEN
5410 024630 113761 002374 002500  MOVB   WAX15,REDDAT(R1) ;SET EXPECTED LO BYTE IN TABLE
5411 024636 113761 002376 002501  MOVB   WAX16,REDDAT+1(R1) ;SET EXPECTED HI BYTE IN TABLE
5412 024644 004737 004312      JSR    PC,WRITAX     ;WRITE DATA BYTES INTO EXTENDED REG
5413 024650 005003              CLR   R3              ;INIT INDEX FOR READING
5414 024652 010337 002402      9$:  MOV    R3,AXNUM      ;GET AX BYTE NO. FOR READING
5415 024656 004737 004124      JSR    PC,READAX     ;READ 2 AX BYTES
5416 024662 023727 002402 000006  CMP    AXNUM,#6      ;SEE IF AX3-15
5417 024670 001003              BNE   10$             ;BR IF NOT
5418 024672 142737 000372 002370  BICB   #AX315U,RAX15 ;MASK OFF UNPREDICTABLE BITS
5419 024700 123763 002370 002500  10$:  CMPB   RAX15,REDDAT(R3) ;COMPARE LO BYTE READ TO EXPECTED
5420 024706 001420              BEQ   12$             ;BR IF LO BYTE MATCHES EXPECTED
5421 024710 005037 002404      CLR   GOODAT
5422 024714 116337 002500 002404  MOVB   REDDAT(R3),GOODAT ;GET EXPECTED LO BYTE
5423 024722 013737 002370 002406  MOV    RAX15,BADDAT  ;GET ACTUAL DATA
5424 024730 004737 004526      JSR    PC,GETALL     ;GET REGS FOR PRINTOUT
5425                          ;REPORT REG MISCMPARE
5426 024734                          ERRDF  3,EM3,ERR3
(4) 024734 104455                          TRAP  C$ERDF
(5) 024736 000003                          .WORD 3
(5) 024740 012222                          .WORD EM3
(5) 024742 015142                          .WORD ERR3
5427 024744                          ESCAPE TST
(3) 024744 104410                          TRAP  C$ESCAPE
(3) 024746 000102                          .WORD L10040-
5428 024750 005237 002402      12$:  INC    AXNUM          ;INCR AX BYTE NO.
5429 024754 123763 002372 002501  CMPB   RAX16,REDDAT+1(R3) ;COMPARE HI BYTE READ TO EXPECTED
5430 024762 001420              BEQ   15$             ;BR IF HI BYTE MATCHES EXPECTED
5431 024764 005037 002404      CLR   GOODAT
5432 024770 116337 002501 002404  MOVB   REDDAT+1(R3),GOODAT ;GET EXPECTED HI BYTE
5433 024776 013737 002372 002406  MOV    RAX16,BADDAT  ;GET ACTUAL DATA
5434 025004 004737 004526      JSR    PC,GETALL     ;GET REGS FOR PRINTOUT
5435                          ;REPORT REG MISCMPARE
5436 025010                          ERRDF  3,EM3,ERR3
(4) 025010 104455                          TRAP  C$ERDF
(5) 025012 000003                          .WORD 3
(5) 025014 012222                          .WORD EM3
(5) 025016 015142                          .WORD ERR3
5437 025020                          ESCAPE TST
(3) 025020 104410                          TRAP  C$ESCAPE
(3) 025022 000026                          .WORD L10040-
5438 025024 062703 000002      15$:  ADD    #2,R3          ;INCR INDEX FOR READS
5439 025030 020327 000010  CMP    R3,#10        ;SEE IF ALL AX BYTES READ YET
5440 025034 002706              BLT   9$              ;BR IF NOT DONE READING YET
5441 025036 062701 000002  ADD    #2,R1          ;INCR INDEX FOR WRITES
5442 025042 020127 000010  CMP    R1,#10        ;SEE IF ALL AX BYTES WRITTEN YET
5443 025046 002660              BLT   6$              ;BR IF NOT DONE WRITING YET

```

5444 025050
(3) 025050
(3) 025050 104401

ENDTST

L10040: TRAP C\$ETST

5445
5446
5447
5448
5449
5450
5451
5452
5453
5454
5455
5456
5457
5458
5459
5460
5461
5462
5463
5464

```
*****  
:SBTTL TEST 16 - REGS 15,16 / AX2-15,AX2-16 READ/WRITE BIT TEST  
:  
* USING REGS 15,16, THE INDIRECT REGS AX2-15,AX2-16 (USYRT REGS 4,5) ARE  
* WRITTEN AND READ USING EACH WORD OF PATTERN K. AX2-15 IS COMPARED  
* TO THE WORD WRITTEN, AND AX2-16 IS ALWAYS COMPARED TO 103. (AX2-16 IS NOT  
* WRITEABLE).  
* PATTERN K =  
* FOR REG 15: 000,377,125,252,001,002,004,010,020,040,100,200,000,000,  
* 000,000,000,000,000,000,000,376,375,373,367,357,337,277,177,  
* 377,377,377,377,377,377,377,377  
* FOR REG 16: 000,377,125,252,000,000,000,000,000,000,000,000,001,002,  
* 004,010,020,040,100,200,377,377,377,377,377,377,377,377,  
* 376,375,373,367,357,337,277,177.  
*****
```

5465 025052
(3) 025052

BGNTST

5466 025052 004737 003576
5467 025056 012701 002704

JSR PC,MSTCLR ;ISSUE MASTER CLEAR T16::
MOV #PATK,R1 ;INIT DATA PATTERN POINTER

5468 025062
5469 025062
(3) 025062

3\$:

BGNSEG ;TRAP C\$BSEG

5470 025064 012737 000004 002402
5471 025072 111137 002374
5472 025076 116137 000001 002376
5473 025104 143737 002565 002374
5474 025112 143737 002566 002376
5475 025120 004737 004312
5476 025124 004737 004124
5477 025130 123737 002370 002374
5478 025136 001416
5479 025140 013737 002374 002404
5480 025146 013737 002370 002406
5481 025154 004737 004526

MOV #4,AXNUM ;SET BYTE NO. = 4
MOV (R1),WAX15 ;SET DATA TO WRITE INTO LO BYTE
MOV 1(R1),WAX16 ;SET DATA TO WRITE INTO HI BYTE
BICB ANBITS+4,WAX15 ;MASK OFF NON-READ/WRITE BITS IN LO BYTE
BICB ANBITS+5,WAX16 ;MASK OFF NON-READ/WRITE BITS IN HI BYTE
JSR PC,WRITAX ;LOAD DATA INTO AX2-15,AX2-16
JSR PC,READAX ;READ AX2-15 AND AX2-16
CMPB RAX15,WAX15 ;COMPARE LO BYTE DATA READ
BEQ 6\$;BR IF DATA MATCHES
MOV WAX15,GOODAT ;SET EXPECTED DATA
MOV RAX15,BADDAT ;SET ACTUAL DATA
JSR PC,GETALL ;GET REGS FOR PRINTOUT

5482
5483 025160
(4) 025160 104455
(5) 025162 000003
(5) 025164 012222
(5) 025166 015142

:REPORT REG MISCOMPARE
ERRDF 3,EM3,ERR3

TRAP C\$ERDF
.WORD 3
.WORD EM3
.WORD ERR3

5484 025170
(3) 025170 104410
(3) 025172 000052

ESCAPE SEG

TRAP C\$ESCAPE
.WORD 10000\$-

5485 025174 005237 002402
5486 025200 123737 002372 002671
5487 025206 001416
5488 025210 005037 002404
5489 025214 113737 002671 002404

6\$:

INC AXNUM ;SET AX BYTE NO. = 5
CMPB RAX16,PATI+5 ;COMPARE HI BYTE DATA READ
BEQ 9\$;BR IF DATA MATCHES
CLR GOODAT
MOV PATI+5,GOODAT ;SET EXPECTED DATA

```

5490 025222 013737 002372 002406      MOV    RAX16,BADAT    ;SET ACTUAL DATA
5491 025230 004737 004526      JSR    PC,GETALL     ;GET REGS FOR PRINTOUT
5492      ;REPORT REG MISCOMPARE
5493      ERRDF 3,EM3,ERR3
(4) 025234 104455
(5) 025236 000003      TRAP  C$ERDF
(5) 025240 012222      .WORD 3
(5) 025242 015142      .WORD  EM3
5494 025244 9$:      .WORD  ERR3
5495 025244      ENDSEG
(3) 025244
(3) 025244 104405      10000$:
5496 025246 062701 000002      ADD    #2,R1         ;INCR PATTERN POINTER      TRAP  C$ESEG
5497 025252 020127 003014      CMP    R1,#PATL     ;SEE IF ALL DATA WRITTEN YET
5498 025256 103701      BLO   3$           ;BR IF NOT DONE YET
5499 025260      ENDTST
(3) 025260
(3) 025260 104401      L10041:
5500      TRAP  C$ETST
5501
5502
5503
5504
5505

```

```

*****
:SBTTL      TEST 17 - AX0-15,AX0-16 READ/WRITE BIT TEST
:
:*
:* IN THIS TEST, A MASTER CLEAR IS DONE, AND THEN A WRITE, READ, AND COMPARE
:* ARE PERFORMED IN REGS AX0-15,AX0-16 USING EACH WORD OF PATTERN L.
:* ANY BITS IN AX0-15,AX0-16 WHICH ARE NOT READ/WRITE ARE MASKED OFF (TO 0)
:* IN THE EXPECTED VALUE BEFORE COMPARISON.
:* PATTERN L =
:*   FOR REG 15: 000,377,000
:*   FOR REG 16: 000,377,000.
*****

```

```

5506      ;*****
5507      ;SBTTL      TEST 17 - AX0-15,AX0-16 READ/WRITE BIT TEST
5508      ;
5509      ;*
5510      ;* IN THIS TEST, A MASTER CLEAR IS DONE, AND THEN A WRITE, READ, AND COMPARE
5511      ;* ARE PERFORMED IN REGS AX0-15,AX0-16 USING EACH WORD OF PATTERN L.
5512      ;* ANY BITS IN AX0-15,AX0-16 WHICH ARE NOT READ/WRITE ARE MASKED OFF (TO 0)
5513      ;* IN THE EXPECTED VALUE BEFORE COMPARISON.
5514      ;* PATTERN L =
5515      ;*   FOR REG 15: 000,377,000
5516      ;*   FOR REG 16: 000,377,000.
5517      ;*****
5518      BGNTST
5519      JSR    PC,MSTCLR    ;ISSUE MASTER CLEAR      T17::
5520      MOV    #PATL,R1    ;INIT DATA PATTERN POINTER
5521      3$:      BGNSEG
5522      MOV    #0,AXNUM     ;SET BYTE NO. = 0      TRAP  C$BSEG
5523      MOVB  (R1),WAX15    ;SET DATA TO WRITE INTO LO BYTE
5524      MOVB  1(R1),WAX16  ;SET DATA TO WRITE INTO HI BYTE
5525      BICB  ANBITS+0,WAX15 ;MASK OFF NON-READ/WRITE BITS IN LO BYTE
5526      BICB  ANBITS+1,WAX16 ;MASK OFF NON-READ/WRITE BITS IN HI BYTE
5527      JSR    PC,WRITAX   ;LOAD DATA INTO AX0-15,AX0-16
5528      JSR    PC,READAX   ;READ AX0-15 AND AX0-16
5529      CMPB  RAX15,WAX15  ;COMPARE LO BYTE DATA READ
5530      BEQ   6$           ;BR IF DATA MATCHES
5531      MOV    WAX15,GOODAT ;SET EXPECTED DATA
5532      MOV    RAX15,BADAT  ;SET ACTUAL DATA
5533      JSR    PC,GETALL     ;GET REGS FOR PRINTOUT
5534      ;REPORT REG MISCOMPARE
5535      ERRDF 3,EM3,ERR3
(4) 025370 104455      TRAP  C$ERDF

```

```
(5) 025372 000003 .WORD 3
(5) 025374 012222 .WORD EM3
(5) 025376 015142 .WORD ERR3
5535 025400          ESCAPE SEG
(3) 025400 104410          TRAP C$ESCAPE
(3) 025402 000046          .WORD 10000$-.
5536 025404 005237 002402 6$: INC AXNUM ;SET AX BYTE NO. = 1
5537 025410 123737 002372 002376 CMPB RAX16,WAX16 ;COMPARE HI BYTE DATA READ
5538 025416 001414          BEQ 9$ ;BR IF DATA MATCHES
5539 025420 013737 002376 002404 MOV WAX16,GOODAT ;SET EXPECTED DATA
5540 025426 013737 002372 002406 MOV RAX16,BADDAT ;SET ACTUAL DATA
5541 025434 004737 004526 JSR PC,GETALL ;GET REGS FOR PRINTOUT
5542          ;REPORT REG MISCOMPARE
5543 025440          ERRDF 3,EM3,ERR3
(4) 025440 104455          TRAP C$ERDF
(5) 025442 000003          .WORD 3
(5) 025444 012222          .WORD EM3
(5) 025446 015142          .WORD ERR3
5544 025450          9$:
5545 025450          ENDSEG
(3) 025450          10000$:
(3) 025450 104405          TRAP C$ESEG
5546 025452 062701 000002 ADD #2,R1 ;INCR PATTERN POINTER
5547 025456 020127 003022 CMP R1,#PATM ;SEE IF ALL DATA WRITTEN YET
5548 025462 103703          BLO 3$ ;BR IF NOT DONE YET
5549 025464          ENDTST
(3) 025464          L10042:
(3) 025464 104401          TRAP C$ETST
5550
5551
5552
5553
5554
5555
5556 :*****
5557 :SBTTL TEST 18 - AX1-15,AX1-16 READ/WRITE BIT TEST
5558 :*
5559 :* IN THIS TEST, A MASTER CLEAR IS DONE, AND THEN A WRITE, READ, AND COMPARE
5560 :* ARE PERFORMED IN REGS AX1-15,AX1-16 USING EACH WORD OF PATTERN K.
5561 :* ANY BITS IN AX1-15,AX1-16 WHICH ARE NOT READ/WRITE ARE MASKED OFF (TO 0)
5562 :* IN THE EXPECTED VALUE BEFORE COMPARISON.
5563 :*****
5563 025466          BGNST
(3) 025466          T18::
5564 025466 004737 003576 JSR PC,MSTCLR ;ISSUE MASTER CLEAR
5565 025472 012701 002704 MOV #PATK,R1 ;INIT DATA PATTERN POINTER
5566 025476          3$:
5567 025476          BGNSEG
(3) 025476 104404          TRAP C$BSEG
5568 025500 012737 000002 002402 MOV #2,AXNUM ;SET BYTE NO. = 2
5569 025506 111137 002374          MOVB (R1),WAX15 ;SET DATA TO WRITE INTO LO BYTE
5570 025512 116137 000001 002376 MOVB 1(R1),WAX16 ;SET DATA TO WRITE INTO HI BYTE
5571 025520 143737 002563 002376 BICB ANBITS+2,WAX15 ;MASK OFF NON-READ/WRITE BITS IN LO BYTE
5572 025526 143737 002564 002376 BICB ANBITS+3,WAX16 ;MASK OFF NON-READ/WRITE BITS IN HI BYTE
5573 025534 004737 004312 JSR PC,WRITAX ;LOAD DATA INTO AX1-15,AX1-16
5574 025540 004737 004124 JSR PC,READAX ;READ AX1-15 AND AX1-16
5575 025544 123737 002370 002374 CMPB RAX15,WAX15 ;COMPARE LO BYTE DATA READ
```

```
5576 025552 001416          BEQ      6$          ;BR IF DATA MATCHES
5577 025554 013737 002374 002404  MOV      WAX15,GOODAT ;SET EXPECTED DATA
5578 025562 013737 002370 002406  MOV      RAX15,BADDAT ;SET ACTUAL DATA
5579 025570 004737 004526          JSR      PC,GETALL   ;GET REGS FOR PRINTOUT
5580                                ;REPORT REG MISCMPARE
5581 025574                                ERRDF   3,EM3,ERR3
(4) 025574 104455                                TRAP   C$ERDF
(5) 025576 000003                                .WORD 3
(5) 025600 012222                                .WORD EM3
(5) 025602 015142                                .WORD ERR3
5582 025604                                ESCAPE SEG
(3) 025604 104410                                TRAP   C$ESCAPE
(3) 025606 000046                                .WORD 10000$.
5583 025610 005237 002402          6$:  INC      AXNUM      ;SET AX BYTE NO. = 3
5584 025614 123737 002372 002376  CMPB    RAX16,WAX16  ;COMPARE HI BYTE DATA READ
5585 025622 001414          BEQ      9$          ;BR IF DATA MATCHES
5586 025624 013737 002376 002404  MOV      WAX16,GOODAT ;SET EXPECTED DATA
5587 025632 013737 002372 002406  MOV      RAX16,BADDAT ;SET ACTUAL DATA
5588 025640 004737 004526          JSR      PC,GETALL   ;GET REGS FOR PRINTOUT
5589                                ;REPORT REG MISCMPARE
5590                                ERRDF   3,EM3,ERR3
(4) 025644 104455                                TRAP   C$ERDF
(5) 025646 000003                                .WORD 3
(5) 025650 012222                                .WORD EM3
(5) 025652 015142                                .WORD ERR3
5591 025654          9$:
5592 025654                                ENDSEG
(3) 025654                                TRAP   10000$.
(3) 025654 104405                                TRAP   C$ESEG
5593 025656 062701 000002          ADD     #2,R1        ;INCR PATTERN POINTER
5594 025662 020127 003014          CMP     R1,#PATL    ;SEE IF ALL DATA WRITTEN YET
5595 025666 103703          BLO     3$          ;BR IF NOT DONE YET
5596 025670          ENDTST
(3) 025670                                TRAP   L10043:
(3) 025670 104401                                TRAP   C$ETST
```

5597
5598
5599
5600
5601
5602
5603
5604
5605
5606
5607
5608
5609
5610
5611
5612
5613
5614
5615
5616
5617

```
::*****  
:SBTTL      TEST 19 - AX3-15,AX3-16 READ/WRITE BIT TEST  
:*  
:* IN THIS TEST A MASTER CLEAR IS DONE AND THEN A WRITE, READ, AND COMPARE ARE  
:* PERFORMED IN REGS AX3-15,AX3-16 USING EACH WORD OF PATTERN V FOR WRITING,  
:* AND PATTERN U FOR COMPARING.  
:* ANY BITS IN AX3-15,AX3-16 WHICH ARE NOT READ/WRITE ARE MASKED OFF (TO 0)  
:* IN THE EXPECTED VALUE BEFORE COMPARISON.  
:* PATTERN V =  
:*   FOR REG 15 : 000,333,331,323,313,233,133,000,000,000,000,  
:*               000,000,000,000,000,000,000,000  
:*   FOR REG 16 : 000,000,000,000,000,000,000,001,002,004,040,  
:*               100,200,346,345,343,307,247,147  
:* PATTERN U =  
:*   FOR REG 15 : 000,001,013,011,021,101,301,000,000,000,000,  
:*               000,000,000,000,000,000,000,000
```

```
5618          ;*      FOR REG 16 : 000,000,000,000,000,000,000,001,002,004,040,  
5619          ;*      100,200,346,345,343,307,247,147  
5620          ;*****  
5621 025672    BGNST  
5622 (3) 025672  
5623 025672 004737 003576          JSR    PC,MSTCLR          ;ISSUE MASTER CLEAR          T19::  
5624 025676 142777 000010 154544 BICB   #LULOP,@BSEL1      ;CLEAR LULOP  
5625 025704 012702 003152          MOV    #PATV,R2          ;INIT PATTERN V POINTER  
5626 025710 012701 003104          MOV    #PATU,R1          ;INIT PATTERN U POINTER  
5627 025714    3$:  
5628 (3) 025714 104404          BGNSEG  
5629 025716 012737 000006 002402    MOV    #6,AXNUM          ;SET BYTE NO. = 6          TRAP    C$BSEG  
5630 025724 111237 002374          MOV    (R2),WAX15        ;SET DATA TO WRITE INTO LO BYTE  
5631 025730 116237 000001 002376    MOV    1(R2),WAX16       ;SET DATA TO WRITE INTO HI BYTE  
5632 025736 004737 004312          JSR    PC,WRITAX         ;LOAD DATA INTO AX3-15,AX3-16  
5633 025742 004737 004124          JSR    PC,READAX        ;READ AX3-15 AND AX3-16  
5634 025746 132737 000001 002374    BITB   #TEST,WAX15      ;SEE IF AN INTERFACE IS SELECTED  
5635 025754 001003          BNE    4$               ;BR IF YES  
5636 025756 142737 000372 002370    BICB   #AX315U,RAX15    ;MASK OFF UNPREDICTABLE BITS  
5637 025764 142737 000040 002370    4$:  BICB   #C32BCC,RAX15  ;CLEAR CRC32 BCC BIT  
5638 025772 123711 002370          CMPB   RAX15,(R1)       ;COMPARE LO BYTE DATA READ  
5639 025776 001417          BEQ    6$               ;BR IF DATA MATCHES  
5640 026000 005037 002404          CLR    GOODAT          ;SET EXPECTED DATA  
5641 026004 111137 002404          MOV    (R1),GOODAT     ;SET ACTUAL DATA  
5642 026010 013737 002370 002406    MOV    RAX15,BADDAT    ;GET REGS FOR PRINTOUT  
5643 026016 004737 004526          JSR    PC,GETALL  
5644 026022    ;REPORT REG MISC COMPARE  
5645 (4) 026022 104455          ERRDF  3,EM3,ERR3  
5646 (5) 026024 000003          TRAP   C$ERDF  
5647 (5) 026026 012222          .WORD 3  
5648 (5) 026030 015142          .WORD EM3  
5649 026032    ESCAPE SEG          .WORD ERR3  
5650 (3) 026032 104410          TRAP   C$ESCAPE  
5651 (3) 026034 000052          .WORD 10000$-.  
5652 026036 005237 002402 000001 6$:  INC    AXNUM            ;SET AX BYTE NO. - 7  
5653 026042 123761 002372          CMPB   RAX16,1(R1)     ;COMPARE HI BYTE DATA READ  
5654 026050 001416          BEQ    9$               ;BR IF DATA MATCHES  
5655 026052 005037 002404          CLR    GOODAT          ;SET EXPECTED DATA  
5656 026056 116137 000001 002404    MOV    1(R1),GOODAT   ;SET ACTUAL DATA  
5657 026064 013737 002372 002406    MOV    RAX16,BADDAT   ;GET REGS FOR PRINTOUT  
5658 026072 004737 004526          JSR    PC,GETALL  
5659 026076    ;REPORT REG MISC COMPARE  
5660 (4) 026076 104455          ERRDF  3,EM3,ERR3  
5661 (5) 026100 000003          TRAP   C$ERDF  
5662 (5) 026102 012222          .WORD 3  
5663 (5) 026104 015142          .WORD EM3  
5664 026106    9$:  ENDSEG          .WORD ERR3  
5665 026106          TRAP   C$ESEG  
5666 (3) 026106 104405          10000$:  
5667 026110 062702 000002          ADD    #2,R2           ;INCR PATTERN V POINTER  
5668 026114 062701 000002          ADD    #2,R1           ;INCR PATTERN U POINTER  
5669 026120 020127 003152          CMP    R1,#PATV       ;SEE IF ALL DATA WRITTEN YET
```

5660 026124 103673
5661 026126
(3) 026126
(3) 026126 104401

BLO 3\$;BR IF NOT DONE YET
ENDTST

L10044: TRAP C\$ETST

5662
5663
5664
5665
5666
5667
5668
5669
5670
5671
5672
5673
5674
5675
5676
5677

:SBTTL TEST 20 - REG 17 - AX2-16 READ/WRITE, MASTER CLEAR TEST
: *
: * THIS TEST CONSISTS OF 2 SUBTESTS. IN THE FIRST SUBTEST, EACH BYTE OF PAT 0
: * IS WRITTEN INTO REG 17 AND AFTER EACH WRITE, AX2-16 IS READ AND COMPARED
: * TO A BYTE OF PAT P.
: * PATTERN 0 = 000,041,004,010,020,040,100,101,200,201,300,111,301,375
: * PATTERN P = 000,113,200,040,020,010,001,104,007,105,007,144,107,157
: * IN THE SECOND SUBTEST, REG 17 IS LOADED WITH 375, A MASTER CLEAR IS ISSUED,
: * AND AX2-16 IS COMPARED TO ITS INITIALIZED STATE (103).
:*****
BGNTST

5678 026130
(3) 026130
5679 026130 004737 003576
5680 026134 012701 003050
5681 026140 012702 003066
5682 026144 012737 000017 002400
5683 026152 012737 000005 002402

JSR PC,MSTCLR ;ISSUE MASTER CLEAR T20::
MOV #PATO,R1 ;INIT PAT 0 POINTER
MOV #PATP,R2 ;INIT PAT P POINTER
MOV #17,REGNUM ;SET LU REG NO. = 17
MOV #5,AXNUM ;SET AX BYTE NO. = 5 FOR AX2-16

: WRITE REG 17, READ AND COMPARE AX2-16

5684
5685
5686
5687 026160
(3) 026160
(3) 026160 104402

BGNSUB

T20.1: TRAP C\$BSUB

5688 026162
5689 026162
(3) 026162 104404
5690 026164 111137 002366
5691 026170 004737 003750
5692 026174 004737 004124
5693 026200 123712 002372
5694 026204 001421
5695 026206 005037 002410
5696 026212 111137 002410
5697 026216 005037 002404
5698 026222 111237 002404
5699 026226 013737 002372 002406
5700 026234 004737 004526

3\$: BGNSEG TRAP C\$BSEG
MOV (R1),WRIBY ;SET BYTE TO BE WRITTEN
JSR PC,WRITLU ;WRITE DATA BYTE INTO REG 17
JSR PC,READAX ;READ AX2
CMPB RAX16,(R2) ;COMPARE AX2-16 TO EXPECTED DATA
BEQ 6\$;BR IF DATA MATCHES
CLR LOADAT ;SET DATA WHICH WAS WRITTEN
MOV (R1),LOADAT
CLR GOODAT
MOV (R2),GOODAT ;SET EXPECTED DATA READ
MOV RAX16,BADDAT ;SET ACTUAL DATA READ
JSR PC,GETALL ;GET REGS FOR PRINTOUT
:REPORT REG MISCOMPARE
ERRDF 3,EM3,ERR5

5701
5702 026240
(4) 026240 104455
(5) 026242 000003
(5) 026244 012222
(5) 026246 016302
5703 026250
5704 026250
(3) 026250

6\$: ENDSEG

10000\$:

```
(3) 026250 104405                                TRAP  C$ESEG
5705 026252 005201                                ;INCR PAT O POINTER
5706 026254 005202                                ;INCR PAT P POINTER
5707 026256 020127 003066                        ;SEE IF ALL BYTES LOADED YET
5708 026262 103737                                ;BR IF NOT
5709 026264                                ENDSUB
(3) 026264                                L10046:
(3) 026264 104403                                TRAP  C$ESUB
-----
5710                                     ; LOAD REG 17, DO MASTER CLEAR, READ AND COMPARE AX2-16
5711                                     ;-----
5712                                     BGNSUB
5713 026266                                T20.2:
(3) 026266                                TRAP  C$BSUB
(3) 026266 104402
5714 026270 112737 000375 002366                MOVB  #375,WRIBYT ;SET DATA TO BE LOADED
5715 026276 004737 003750                        JSR   PC,WRITLU  ;LOAD DATA INTO REG 17
5716 026302 004737 003576                        JSR   PC,MSTCLR  ;PERFORM MASTER CLEAR
5717 026306 004737 004124                        JSR   PC,READAX  ;READ AX2-15,AX2-16
5718 026312 123737 002372 002671                CMPB  RAX16,PAT1+5 ;SEE IF AX2-16 WAS INIT'D CORRECTLY
5719 026320 001416                                BEQ   6$         ;BR IF YES
5720 026322 005037 002404                        CLR   GOODAT
5721 026326 113737 002671 002404                MOVB  PAT1+5,GOODAT ;SET EXPECTED DATA
5722 026334 013737 002372 002406                MOV   RAX16,BADDAT ;SET ACTUAL DATA
5723 026342 004737 004526                        JSR   PC,GETALL  ;GET REGS FOR PRINTOUT
5724                                     ;REPORT REG NOT INITIALIZED BY MASTER CLEAR
5725                                     ERRDF 2,EM2,ERR3
(4) 026346 104455                                TRAP  C$ERDF
(5) 026350 000002                                .WORD 2
(5) 026352 012163                                .WORD EM2
(5) 026354 015142                                .WORD ERR3
5726 026356                                6$:
5727 026356                                ENDSUB
(3) 026356                                L10047:
(3) 026356 104403                                TRAP  C$ESUB
5728 026360                                ENDTST
(3) 026360                                L10045:
(3) 026360 104401                                TRAP  C$ETST
5729
5730
5731
5732
5733
5734
5735                                     ;*****
5736                                     ;SBTTL TEST 21 - TRANSMITTER BUFFER DATA TEST
5737                                     ;* A MASTER CLEAR IS DONE FIRST, AND THEN A BYTE OF PATTERN N IS LOADED INTO
5738                                     ;* REG 11 AND THE NEXT BYTE IS LOADED TWICE INTO REG 10. THE PROGRAM THEN WAITS
5739                                     ;* AT LEAST 50 MICRO-SEC, AND THEN IT READS AND COMPARES AX1-15 TO THE BYTE
5740                                     ;* WHICH WAS LOADED INTO REG 10, AND IT READS AND COMPARES AX1-16 TO THE BYTE
5741                                     ;* WHICH WAS LOADED INTO REG 11. THIS PROCESS IS REPEATED (INCLUDING THE MASTER
5742                                     ;* CLEAR) FOR EACH PAIR OF BYTES IN PATTERN N.
5743                                     ;* PATTERN N =
5744                                     ;* FOR REG 10: 000,125,252,377,000,000,000
5745                                     ;* FOR REG 11: 000,000,000,000,005,012,017
5746 026362                                     ;*****
(3) 026362                                BGNTST
T21::
```



```
5747 026362 012701 003032      MOV    #PATN,R1      ;INIT PATTERN POINTER
5748 026366 012737 026630 002362  MOV    #A2,RETADR   ;SET SUBROUTINE ERROR RETURN ADDRESS
5749 026374      BGNSUB
(3) 026374      T21.1:
(3) 026374 104402      TRAP   C$BSUB
5750 026376 004737 003576      3$:   JSR    PC,MSTCLR   ;ISSUE MASTER CLEAR
5751 026402 004737 004662      JSR    PC,OSIRDY   ;CHECK ORDY AND OCOR FOR EXPECTED STATES
5752 026406 000001      1
5753 026410 012737 000011 002400  MOV    #11,REGNUM   ;SET LU REG NO. = 11
5754 026416 111137 002366      MOVB   (R1),WRIBYT  ;SET DATA BYTE TO BE WRITTEN
5755 026422 004737 003750      JSR    PC,WRITLU   ;WRITE BYTE INTO REG 11
5756 026426 012737 000010 002400  MOV    #10,REGNUM   ;SET LU REG NO. = 10
5757 026434 116137 000001 002366  MOVB   1(R1),WRIBYT ;SET DATA BYTE TO BE WRITTEN
5758 026442 004737 003750      JSR    PC,WRITLU   ;WRITE BYTE INTO REG 10
5759 026446 004737 003750      JSR    PC,WRITLU   ;WRITE IT AGAIN (SO 2 ENTRIES ARE IN SILO)
5760 026452 004737 005146      JSR    PC,WAIT50   ;WAIT FOR SILO DATA TO RIPPLE
5761 026456 004737 004662      JSR    PC,OSIRDY   ;CHECK ORDY AND OCOR FOR EXPECTED STATES
5762 026462 000003      3
5763 026464 012737 000002 002402  MOV    #2,AXNUM     ;SET BYTE NO. FOR AX1-15
5764 026472 004737 004124      JSR    PC,READAX   ;READ AX1-15, AX1-16
5765 026476 123761 002370 000001  CMPB   RAX15,1(R1) ;COMPARE AX1-15 TO EXPECTED
5766 026504 001420      BEQ    6$          ;BR IF MATCH
5767 026506 005037 002404      CLR    GOODAT
5768 026512 116137 000001 002404  MOVB   1(R1),GOODAT ;SET EXPECTED DATA
5769 026520 013737 002370 002406  MOV    RAX15,BADDAT ;SET ACTUAL DATA
5770 026526 004737 004526      JSR    PC,GETALL   ;GET REGS FOR PRINTOUT
5771      ;REPORT REG MISCMPARE
5772 026532      ERRDF  3,EM3,ERR3
(4) 026532 104455      TRAP   C$ERDF
(5) 026534 000003      .WORD  3
(5) 026536 012222      .WORD  EM3
(5) 026540 015142      .WORD  ERR3
5773 026542      ESCAPE SUB
(3) 026542 104410      TRAP   C$ESCAPE
(3) 026544 000064      .WORD  L10051-.
5774 026546 005237 002402      6$:   INC    AXNUM        ;INCR AX BYTE NO.
5775 026552 123711 002372      CMPB   RAX16,(R1)  ;COMPARE AX1-16 TO EXPECTED
5776 026556 001417      BEQ    9$          ;BR IF MATCH
5777 026560 005037 002404      CLR    GOODAT
5778 026564 111137 002404      MOVB   (R1),GOODAT ;SET EXPECTED DATA
5779 026570 013737 002372 002406  MOV    RAX16,BADDAT ;SET ACTUAL DATA
5780 026576 004737 004526      JSR    PC,GETALL   ;GET REGS FOR PRINTOUT
5781      ;REPORT REG MISCMPARE
5782 026602      ERRDF  3,EM3,ERR3
(4) 026602 104455      TRAP   C$ERDF
(5) 026604 000003      .WORD  3
(5) 026606 012222      .WORD  EM3
(5) 026610 015142      .WORD  ERR3
5783 026612      ESCAPE SUB
(3) 026612 104410      TRAP   C$ESCAPE
(3) 026614 000014      .WORD  L10051-.
5784 026616 062701 000002      9$:   ADD    #2,R1        ;INCR DATA PATTERN POINTER
5785 026622 020127 003050      CMP    R1,#PATO    ;SEE IF ALL DATA BYTES WRITTEN YET
5786 026626 103663      BLO   3$          ;BR IF NOT DONE YET
5787 026630      A2:
5788 026630      ENDSUB
```

(3) 026630
 (3) 026630 104403
 5789 026632
 (3) 026632
 (3) 026632 104401

L10051: TRAP C\$ESUB
 L10050: TRAP C\$ETST

5790
 5791
 5792
 5793
 5794
 5795
 5796
 5797
 5798
 5799
 5800
 5801
 5802
 5803
 5804
 5805
 5806
 5807
 5808
 5809

```

:*****
:SBTTL TEST 22 - TRANSMITTER BUFFER SEQUENCING TEST
:
:* FIRST, A MASTER CLEAR IS DONE, AND THE PROGRAM CHECKS FOR ORDY=1, OCOR=0.
:* THEN, 2 TSOM CHARS ARE LOADED INTO THE TX SILO, AND ALLOWED TO RIPPLE
:* DOWN TO THE OUTPUT. THE PROGRAM CHECKS FOR ORDY=1, OCOR=1.
:* NEXT, THE PROGRAM CYCLES THE STEPLU BIT UNTIL OCOR=0 AGAIN, AND CHECKS FOR
:* THIS TO OCCUR WITHIN 3 CYCLES.
:* THE SILO IS THEN FILLED WITH 64 BYTES OF A 256-BYTE BINARY COUNT PATTERN
:* (000-377) AND THE PROGRAM CHECKS FOR ORDY=0 AFTER THE 64TH CHAR IS LOADED.
:* THE PROGRAM CYCLES STEPLU FOR 8 CYCLES AND CHECKS THAT AFTER THE 8TH, ORDY=1.
:* AX1-15 IS READ AND COMPARED TO EXPECTED DATA.
:* THE REST OF THE BINARY COUNT DATA BYTES ARE LOADED, CYCLED 8 CLOCKS, READ AND
:* COMPARED, A BYTE AT A TIME. UPON COMPLETION, THE SILO IS CHECKED TO BE EMPTY
:* WITH ORDY=1, OCOR=0.
:*****
    
```

5810
 5811 026634
 (3) 026634
 5812 026634 012737 027310 002362
 5813
 5814
 5815
 5816 026642 004737 003576
 5817 026646 004737 004662
 5818 026652 000001
 5819
 5820
 5821
 5822 026654 012737 000400 002422
 5823 026662 004737 005174
 5824 026666 004737 005174
 5825 026672 004737 005146
 5826 026676 004737 004662
 5827 026702 000003
 5828
 5829
 5830
 5831 026704 005001
 5832 026706 012737 000017 002400
 5833 026714 004737 005254
 5834 026720 000001
 5835 026722 004737 003672
 5836 026726 132737 000020 002364
 5837 026734 001404
 5838 026736 005201
 5839 026740 020127 000003

```

BGNTST
:
:----- T22:-----
:MOV #A3,RETADR ;SET SUBR ERROR RETURN ADDR
:-----
:SET MASTER CLEAR, CHECK FOR ORDY=1, OCOR=0
:-----
:JSR PC,MSTCLR ;ISSUE MASTER CLEAR
:JSR PC,OSIRDY ;CHK ORDY=1, OCOR=0
:1
:-----
:LOAD 2 SOM CHARS, ALLOW SILO TO RIPPLE, CHK ORDY=1, OCOR=1
:-----
:MOV #TXSOM, TXWORD ;SET DATA TO WRITE INTO SILO
:JSR PC,LDTXSI ;LOAD THE SILO WITH SOM
:JSR PC,LDTXSI ;LOAD ANOTHER SOM
:JSR PC,WAIT50 ;WAIT FOR DATA TO RIPPLE
:JSR PC,OSIRDY ;CHK ORDY=1, OCOR=1
:3
:-----
:CLOCK LINE UNIT, CHK FOR OCOR = 0 WITHIN 3 CYCLES
:-----
:CLR R1 ;INIT CYCLE COUNTER TO 0
:MOV #17,REGNUM ;SET REG NO. = 17
:3$: JSR PC,STPLU ;STEP LU 1 CYCLE
:1
:JSR PC,READLU ;READ REG 17
:BITB #OCOR,REDBYT ;SEE IF OCOR = 0 YET
:BEQ 6$ ;BR IF OCOR = 0
:INC R1 ;INCR CYCLE COUNT
:CMP R1,#3 ;SEE IF 3 CYCLES DONE YET
    
```

```
5840 026744 002763
5841 026746 004737 004662
5842 026752 000001
5843
5844
5845
5846 026754 005003
5847 026756 010337 002422
5848 026762 004737 005174
5849 026766 004737 005146
5850 026772 020327 000077
5851 026776 001004
5852 027000 012737 000002 027022
5853 027006 000403
5854 027010 012737 000003 027022
5855 027016 004737 004662
5856 027022 000000
5857 027024 005203
5858 027026 020327 000100
5859 027032 002751
5860
5861
5862
5863 027034 012737 000011 002400
5864 027042 005001
5865 027044 004737 005254
5866 027050 000001
5867 027052 004737 003672
5868 027056 132737 000020 002364
5869 027064 001004
5870 027066 005201
5871 027070 020127 000010
5872 027074 002763
5873 027076 004737 004662
5874 027102 000003
5875
5876
5877
5878 027104 005004
5879 027106 012737 000002 002402
5880 027114 004737 004124
5881 027120 123704 002370
5882 027124 001415
5883 027126 010437 002404
5884 027132 013737 002370 002406
5885 027140 004737 004526
5886
5887 027144
(4) 027144 104455
(5) 027146 000003
(5) 027150 012222
(5) 027152 015142
5888 027154
(3) 027154 104410
(3) 027156 000132
5889 027160
```

```
6$: BLT 3$ :BR IF NO
JSR PC,OSIRDY :CHK ORDY=1, OCOR=0
1

-----
: LOAD 64 BINARY COUNT CHARS INTO SILO, CHK ORDY=0
-----
8$: CLR R3 :INIT PATTERN FOR WRITING
MOV R3,TXWORD :SET DATA TO BE WRITTEN
JSR PC,LDTXSI :LOAD DATA CHAR INTO TX SILO
JSR PC,WAIT50 :WAIT FOR DATA TO RIPPLE IN SILO
CMP R3,#63. :SEE IF 64TH CHAR JUST LOADED
BNE 9$ :BR IF NO
MOV #2,14$ :SET UP TO CHK ORDY=0,OCOR=1
BR 12$
9$: MOV #3,14$ :SET UP TO CHK ORDY=1,OCOR=1
12$: JSR PC,OSIRDY :CHK ORDY, OCOR
14$: .WORD 0
INC R3 :INCR PATTERN FOR WRITES
CMP R3,#64. :SEE IF 64 CHARS LOADED YET
BLT 8$ :BR IF NO

-----
: CLOCK LINE UNIT, CHECK ORDY = 1 WITHIN 8 CYCLES
-----
16$: MOV #11,REGNUM :SET REG NO. = 11
CLR R1 :INIT CYCLE COUNT
JSR PC,STPLU :CLOCK LU FOR 1 CYCLE
1
JSR PC,READLU :READ REG 11
BITB #ORDY,REDBYT :SEE IF ORDY = 1 YET
BNE 19$ :BR IF YES
INC R1 :INCR CYCLE COUNT
CMP R1,#8. :SEE IF 8 CYCLES YET
BLT 16$ :BR IF NOT YET
19$: JSR PC,OSIRDY :CHK ORDY = , OCOR = 1
3

-----
: READ AND COMPARE FIRST CHARACTER IN AX1-15
-----
CLR R4 :INIT PATTERN FOR READING
MOV #2,AXNUM :SET AX BYTE NO. FOR AX1-15
JSR PC,READAX :READ AX1-15
CMPB RAX15,R4 :COMPARE AX1-15 TO EXPECTED
BEQ 20$ :BR IF MATCH
MOV R4,GOODAT :SET EXPECTED DATA
MOV RAX15,BADAT :SET ACTUAL DATA
JSR PC,GETALL :GET REGS FOR PRINTOUT
:REPORT REG MISCMPARE
ERRDF 3,EM3,ERR3

TRAP C$ERDF
.WORD 3
.WORD EM3
.WORD ERR3

ESCAPE TST

TRAP C$ESCAPE
.WORD L10052-
```

5890
5891
5892
5893 027160 020327 000377
5894 027164 003010
5895 027166 010337 002422
5896 027172 004737 005174
5897 027176 005203
5898 027200 004737 004662
5899 027204 000002
5900 027206 005204
5901 027210 004737 005254
5902 027214 000010
5903 027216 004737 004124
5904 027222 123704 002370
5905 027226 001415
5906 027230 010437 002404
5907 027234 013737 002370 002406
5908 027242 004737 004526
5909
5910 027246
(4) 027246 104455
(5) 027250 000003
(5) 027252 012222
(5) 027254 015142
5911 027256
(3) 027256 104410
(3) 027260 000030
5912 027262 020427 000377
5913 027266 001004
5914 027270 004737 004662
5915 027274 000001
5916 027276 000404
5917 027300 004737 004662
5918 027304 000003
5919 027306 000724
5920 027310
5921 027310
(3) 027310
(3) 027310 104401

```
-----  
: LOAD AND COMPARE REST OF CHARS, MONITOR ORDY, OCOR  
-----  
24$:  CMP      R3,#255.      ;SEE IF ALL CHARS LOADED YET  
      BGT      26$          ;BR IF YES  
      MOV      R3, TXWORD    ;SET DATA TO BE WRITTEN  
      JSR      PC, LDTXSI    ;LOAD DATA CHAR INTO TX SILO  
      INC      R3           ;INCR DATA TO BE WRITTEN  
      JSR      PC, OSIRDY    ;CHK ORDY=0, OCOR=1  
      2  
26$:  INC      R4           ;INCR PAT FOR READING  
      JSR      PC, STPLU     ;CLOCK LINE UNIT FOR 8 CYCLES  
      8.  
      JSR      PC, READAX    ;READ AX1-15  
      CMPB     RAX15, R4     ;COMPARE AX1-15 TO EXPECTED  
      BEQ      27$          ;BR IF MATCH  
      MOV      R4, GOODAT    ;SET EXPECTED DATA  
      MOV      RAX15, BADDAT ;SET ACTUAL DATA  
      JSR      PC, GETALL    ;GET REGS FOR PRINTOUT  
:REPORT REG MISC  
      COMPARE  
      ERRDF    3, EM3, ERR3  
  
      ESCAPE  TST  
  
27$:  CMP      R4,#255.      ;SEE IF WE READ LAST CHAR YET  
      BNE     29$          ;BR IF NOT  
      JSR      PC, OSIRDY    ;CHK ORDY=1, OCOR=0  
      1  
29$:  JSR      PC, OSIRDY    ;CHK ORDY=1, OCOR=1  
      3  
      BR      24$          ;CONTINUE  
  
A3:  
ENDTST
```

TRAP C\$ERDF
.WORD 3
.WORD EM3
.WORD ERR3

TRAP C\$ESCAPE
.WORD L10052-

L10052: TRAP C\$ETST

5922
5923
5924
5925
5926
5927
5928
5929
5930
5931
5932
5933
5934
5935
5936
5937

```
*****  
: SBTTL      TEST 23 - TX MSG TIMING TEST, CHAR MODE, WITH CRC  
: *  
: * IN THIS TEST, AN ENTIRE MESSAGE IS TRANSMITTED (USING STEPLU AND LULOOP)  
: * AND THE PROGRAM MONITORS THE OCCURRENCE OF USYRT TX BUFFER EMPTY FLAGS  
: * (BY SCANNING ORDY AND OCOR) AND OACT, AT EACH STEP. THE TEST IS DONE IN  
: * CHARACTER ORIENTED PROTOCOL MODE, USING 8-BIT CHARS AND CRC-16.  
: * THE FOLLOWING STEPS ARE DONE:  
: * A MASTER CLEAR IS DONE, AND THE LINE UNIT IS PLACED IN CHAR MODE.  
: * SOM IS SET TWICE TO SEND 2 SYNCH CHARS. THEN, 2 000 CHARS ARE SENT, AND  
: * THEN 2 TERMINATING SYNCHS ARE SENT.
```

5938
5939
5940
5941
(3)
5942
5943
5944
5945
5946
5947
5948
5949
5950
5951
5952
5953
5954
5955
5956
5957
5958
5959
5960
5961
5962
5963
5964
5965
5966
5967
5968
5969
5970
(3)
(3)
5971
5972
5973
5974
5975
5976
5977
5978
5979
5980
5981
5982
5983
5984
5985
5986
5987
5988
5989
5990

027312
027312
012737 027424 002362
027320 004737 005540
027324 000226
027326 000011
027330 004737 006122
027334 000000
027336 100010
027340 004737 006122
027344 000000
027346 000010
027350 004737 006122
027354 001000
027356 000010
027360 004737 006122
027364 001000
027366 000010
027370 004737 006122
027374 001000
027376 000020
027400 004737 006122
027404 001000
027406 000010
027410 004737 006122
027414 001000
027416 000010
027420 004737 006274
027424
027424 004737 003576
027430
027430
(3) 027430 104401
027432

;* THE TRANSMITTER IS THEN DISABLED, USING OC, AND OACT IS MONITORED FOR THE
;* CLEARED STATE AFTER THE 3RD SYNCH COMPLETES.

BGNTST

T23::
MOV #A4,RETADR ;SET TEST EXIT ADDRESS FOR ERRORS
JSR PC,INITRN ;DO MASTER CLR, LOAD 2 SOM'2
SYNCH
STRIP!DDCMP
JSR PC,TXCHAR ;LOAD A 000 CHAR, TX FIRST SYNCH (226)
000
CHPCHK!8.
JSR PC,TXCHAR ;LOAD 2ND 000 CHAR, TX 2ND SYNCH
000
8.
JSR PC,TXCHAR ;LOAD EOM CHAR, TX FIRST 000 CHAR
TXEOM
8.
JSR PC,TXCHAR ;LOAD EOM CHAR, TX 2ND 000 CHAR
TXEOM
8.
JSR PC,TXCHAR ;LOAD EOM, TX CRC-16 CHAR
TXEOM
16.
JSR PC,TXCHAR ;LOAD EOM, TX FIRST TERMINATING SYNCH
TXEOM
8.
JSR PC,TXCHAR ;LOAD EOM, TX 2ND TERMINATING SYNCH
TXEOM
8.
JSR PC,ENDTRN ;SET OC, MONITOR OCOR
A4:
JSR PC,MSTCLR ;ISSUE MASTER CLEAR TO CLEAN UP
ENDTST

L10053: TRAP C\$ETST

:SBTTL TEST 24 - TX MSG TIMING TEST, BIT MODE, WITH CRC

;* IN THIS TEST, AN ENTIRE MESSAGE IS TRANSMITTED (USING STEPLU AND LULOOP)
;* AND THE PROGRAM MONITORS THE OCCURRENCE OF USYRT TX BUFFER EMPTY FLAGS
;* (BY SCANNING ORDY AND OCOR) AND OACT, AT EACH STEP. THE TEST IS DONE IN
;* BIT ORIENTED PROTOCOL MODE, USING 8-BIT CHARS AND CRC-CCITT-1.
;* THE FOLLOWING STEPS ARE DONE:
;* A MASTER CLEAR IS DONE, AND THE LINE UNIT IS PLACED IN BIT MODE.
;* SOM IS SET TWICE TO SEND 2 FLAG CHARS. THEN, 2 000 CHARS ARE SENT, AND
;* THEN 2 TERMINATING FLAGS ARE SENT.
;* THE TRANSMITTER IS THEN DISABLED, USING OC, AND OACT IS MONITORED FOR THE
;* CLEARED STATE.

BGNTST

```
(3) 027432
5991 027432 012737 027534 002362      MOV      #A5,RETADR      ;SET TFST EXIT ADDRESS FOR ERRORS
5992 027440 004737 005540                JSR      PC,INITRN      ;DO MASTER CLR, LOAD 2 SOM'2
5993 027444 000000
5994 027446 000000
5995 027450 004737 006122                JSR      PC,TXCHAR      ;LOAD A 000 CHAR, TX FIRST FLAG
5996 027454 000000
5997 027456 100010
5998 027460 004737 006122                JSR      PC,TXCHAR      ;LOAD 2ND 000 CHAR, TX 2ND FLAG
5999 027464 000000
6000 027466 000010
6001 027470 004737 006122                JSR      PC,TXCHAR      ;LOAD EOM CHAR, TX FIRST 000 CHAR
6002 027474 001000
6003 027476 000010
6004 027500 004737 006122                JSR      PC,TXCHAR      ;LOAD EOM, TX 2ND 000 CHAR AND CRC-CCITT-1 CHAR
6005 027504 001000
6006 027506 000030
6007 027510 004737 006122                JSR      PC,TXCHAR      ;LOAD EOM, TX FIRST TERMINATING FLAG
6008 027514 001000
6009 027516 000010
6010 027520 004737 006122                JSR      PC,TXCHAR      ;LOAD EOM, TX 2ND TERMINATING FLAG
6011 027524 001000
6012 027526 000010
6013 027530 004737 006274                JSR      PC,ENDTRN      ;SET OC, MONITOR OCOR
6014 027534
6015 027534 004737 003576                JSR      PC,MSTCLR      ;ISSUE MASTER CLEAR TO CLEAN UP
6016 027540
(3) 027540
(3) 027540 104401
```

A5:
ENDTST

L10054:
TRAP C\$ETST

6017
6018
6019
6020
6021
6022
6023
6024
6025
6026
6027
6028
6029
6030
6031
6032
6033
6034
6035
6036
6037

```
*****
.SBTTL      TEST 25 - TX MSG TIMING TEST, CHAR MODE, WITH NO CRC
*
* IN THIS TEST, AN ENTIRE MESSAGE IS TRANSMITTED (USING STEPLU AND LULOOP)
* AND THE PROGRAM MONITORS THE OCCURRENCE OF USYRT TX BUFFER EMPTY FLAGS
* (BY SCANNING ORDY AND OCOR) AND OACT, AT EACH STEP. THE TEST IS DONE IN
* CHARACTER ORIENTED PROTOCOL MODE, USING 8-BIT CHARS AND NO ERROR CHECKING.
* THE FOLLOWING STEPS ARE DONE:
* A MASTER CLEAR IS DONE, AND THE LINE UNIT IS PLACED IN CHAR MODE.
* SOM IS SET TWICE TO SEND 2 SYNCH CHARS. THEN, 2 000 CHARS ARE SENT, AND
* THEN 2 TERMINATING SYNCHS ARE SENT.
* THE TEST IS PERFORMED WITH TXEN (REG14, BIT6) SET, AND THE PROGRAM CHECKS
* THAT THIS HOLDS RTS HIGH PAST THE END OF THE MESSAGE.
* THE TRANSMITTER IS THEN DISABLED, USING OC, AND OACT IS MONITORED FOR THE
* CLEARED STATE.
*****
BGNTST
```

```
(3) 027542
6039 027542 012737 027730 002362      MOV      #A6,RETADR      ;SET TEST EXIT ADDRESS FOR ERRORS
6040 027550 004737 005540                JSR      PC,INITRN      ;DO MASTER CLR, LOAD 2 SOM'2
6041 027554 000226
6042 027556 000311                SYNCH      CRC2!CRC1!STRIP!DDCMP
```

T25::

6043	027560	012737	000014	002400	MOV	#14,REGNUM	:SET REG NO. = 14		
6044	027566	012737	000100	002366	MOV	#TXEN,WRIBYT	:SET TXEN BIT		
6045	027574	004737	003750		JSR	PC,WRITLU	:LOAD TXEN BIT IN REG 14		
6046	027600	004737	006122		JSR	PC,TXCHAR	:LOAD A 000 CHAR, TX FIRST SYNCH (226)		
6047	027604	000000			000				
6048	027606	100010			CHPCHK!8.				
6049	027610	004737	006122		JSR	PC,TXCHAR	:LOAD 2ND 000 CHAR, TX 2ND SYNCH		
6050	027614	000000			000				
6051	027616	000010			8.				
6052	027620	004737	006122		JSR	PC,TXCHAR	:LOAD EOM CHAR, TX FIRST 000 CHAR		
6053	027624	001000			TXEOM				
6054	027626	000010			8.				
6055	027630	004737	006122		JSR	PC,TXCHAR	:LOAD EOM CHAR, TX 2ND 000 CHAR		
6056	027634	001000			TXEOM				
6057	027636	000010			8.				
6058	027640	004737	006122		JSR	PC,TXCHAR	:LOAD EOM, TX FIRST TERMINATING SYNCH		
6059	027644	001000			TXEOM				
6060	027646	000010			8.				
6061	027650	004737	006122		JSR	PC,TXCHAR	:LOAD EOM, TX 2ND TERMINATING SYNCH		
6062	027654	001000			TXEOM				
6063	027656	000010			8.				
6064	027660	004737	005254		JSR	PC,STPLU	:CLK PAST END OF MSG		
6065	027664	000030			24.				
6066	027666	012737	000013	002400	MOV	#13,REGNUM	:SET REG NO. = 13		
6067	027674	004737	003672		JSR	PC,READLU	:READ REG 13		
6068	027700	032737	000040	002364	BIT	#RTS,REDBYT	:CHK FOR RTS STILL SET		
6069	027706	001006			BNE	3\$:BR IF RTS SET		
6070	027710	004737	004526		JSR	PC,GETALL	:GET REGS FOR PRINTOUT		
6071					:REPORT	RTS NOT SET			
6072	027714				ERRDF	60,EM60,ERR7			
(4)	027714	104455						TRAP	C\$ERDF
(5)	027716	000074						.WORD	60
(5)	027720	014232						.WORD	EM60
(5)	027722	017472						.WORD	ERR7
6073	027724	004737	006274		3\$: JSR	PC,ENDTRN	:SET OC, MONITOR OCOR		
6074	027730				A6:				
6075	027730	004737	003576		JSR	PC,MSTCLR	:ISSUE MASTER CLEAR TO CLEAN UP		
6076	027734				ENDTST				
(3)	027734							L10055:	
(3)	027734	104401						TRAP	C\$ETST

6077
6078
6079
6080
6081
6082
6083
6084
6085
6086
6087
6088
6089
6090
6091
6092

```
*****  
:SBTTL TEST 26 - TX UNDERRUN SET AND CLEAR TEST - CHAR MODE  
:*  
:* IN THIS TEST, A TX UNDERRUN ERROR IS FORCED IN EACH OF 2 SITUATIONS,  
:* AND THEN CLEARED DIFFERENTLY IN EACH.  
:* IN THE FIRST, A MESSAGE IS INITIATED, A 000 CHAR IS SENT, AND THE TX  
:* BUFFER IS NOT SERVICED IN RESPONSE TO THE USYRT TX BUFFER EMPTY FLAG,  
:* WHICH CAUSES UNRR TO SET IN REG 11. THEN, SOM IS SET TO CLEAR THE ERROR,  
:* AND THIS IS VERIFIED.  
:* IN THE SECOND SITUATION, A MSG IS INITIATED, A 000 CHAR IS SENT, AND THE TX  
:* BUFFER IS NOT SERVICED IN RESPONSE TO THE USYRT TX BUFFER EMPTY FLAG, WHICH
```

6093
6094
6095
6096 027736
(3) 027736
6097
6098
6099
6100 027736 012737 030274 002362
6101 027744 004737 005540
6102 027750 000226
6103 027752 000011
6104 027754 004737 006122
6105 027760 000000
6106 027762 100010
6107 027764 004737 005254
6108 027770 000016
6109 027772 012737 000011 002400
6110 030000 004737 003672
6111 030004 132737 000001 002364
6112 030012 001410
6113 030014 004737 004526
6114
6115 030020
(4) 030020 104455
(5) 030022 000020
(5) 030024 012624
(5) 030026 017472
6116 030030 000137 030274
6117 030034 004737 005254
6118 030040 000003
6119 030042 004737 003672
6120 030046 132737 000001 002364
6121 030054 001010
6122 030056 004737 004526
6123
6124 030062
(4) 030062 104455
(5) 030064 000016
(5) 030066 012560
(5) 030070 017472
6125 030072 000137 030274
6126 030076 013737 000001 002422
6127 030104 004737 005174
6128 030110 004737 005146
6129 030114 004737 005254
6130
6131 030120 000002
6132 030122 004737 003672
6133 030126 132737 000001 002364
6134 030134 001410
6135 030136 004737 004526
6136
6137 030142
(4) 030142 104455
(5) 030144 000015

;* AGAIN CAUSES UNRR TO SET. THEN, A MASTER CLEAR IS DONE, AND THE UNRR BIT
;* IS CHECKED TO BE CLEARED.
:*****
BGNTST

T26::

: CAUSE TX UNDERRUN, CHK UNRR = 1; SET SOM, CHK UNRR = 0

MOV #A7,RETADR ;SET TEST EXIT ADDRESS FOR ERRORS
JSR PC,INITRN ;DO MASTER CLEAR, LOAD 2 SOM'S
SYNCH
STRIP!DDCMP
JSR PC, TXCHAR ;LOAD A 000 CHAR, TX FIRST SYNCH
000
CHPCHK!8.
JSR PC,STPLU ;CLOCK THE TRANSMITTER UNTIL 7 BITS OF
14. ; THE 000 CHAR HAVE BEEN TRANSMITTED
MOV #11,REGNUM ;SET LU REG NO. = 11
JSR PC,READLU ;READ REG 11
BITB #UNRR,REDBYT ;CHK FOR UNRR = 0
BEQ 6\$;BR IF UNRR = 0
JSR PC,GETALL ;GET REGS FOR PRINTOUT
:REPORT UNRR NOT CLEARED
ERRDF 16,EM16,ERR7
TRAP C\$ERDF
.WORD 16
.WORD EM16
.WORD ERR7
6\$: JMP A7 ;SKIP TO END OF TEST
JSR PC,STPLU ;CLOCK LAST BIT OF 000 CHAR
3
JSR PC,READLU ;READ REG 11
BITB #UNRR,REDBYT ;CHK FOR UNRR = 1
BNE 9\$;BR IF UNRR = 1
JSR PC,GETALL ;GET REGS FOR PRINTOUT
:REPORT UNRR NOT SET
ERRDF 14,EM14,ERR7
TRAP C\$ERDF
.WORD 14
.WORD EM14
.WORD ERR7
9\$: JMP A7 ;SKIP TO END OF TEST
MOV ^H#TXSOM, TXWORD ;SET SOM CHAR TO BE WRITTEN
JSR PC,LDTXSI ;LOAD SOM CHAR INTO TX SILO
JSR PC, WAIT50 ;WAIT FOR SILO DATA TO RIPPLE
JSR PC,STPLU ;CLOCK LU FOR 2 CYCLES
2
JSR PC,READLU ;READ REG 11
BITB #UNRR,REDBYT ;CHK FOR UNRR = 0
BEQ 12\$;BR IF UNRR = 0
JSR PC,GETALL ;GET REGS FOR PRINTOUT
:REPORT UNRR NOT CLEARED BY SOM
ERRDF 13,EM13,ERR7
TRAP C\$ERDF
.WORD 13


```
(5) 030146 012530
(5) 030150 017472
6138 030152 000137 030274
6139 030156
6140
6141
6142
6143 030156 004737 005540
6144 030162 000226
6145 030164 000051
6146 030166 004737 006122
6147 030172 000000
6148 030174 100010
6149 030176 004737 005254
6150 030202 000021
6151 030204 004737 003672
6152 030210 132737 000001 002364
6153 030216 001010
6154 030220 004737 004526
6155
6156 030224
(4) 030224 104455
(5) 030226 000016
(5) 030230 012560
(5) 030232 017472
6157 030234 000137 030274
6158 030240 004737 003576
6159 030244 004737 003672
6160 030250 132737 000001 002364
6161 030256 001406
6162 030260 004737 004526
6163
6164 030264
(4) 030264 104455
(5) 030266 000017
(5) 030270 012575
(5) 030272 017472
6165 030274 004737 003576
6166 030300
(3) 030300
(3) 030300 104401
6167
6168
6169
6170
6171
6172
6173
6174
6175
6176
6177
6178
6179
6180
6181 030302
```

```
12$:
-----
: CAUSE TX UNDERRUN, CHK UNRR = 1; DO MASTER CLR, CHK UNRR = 0
-----
JSR PC,INITRN ;DO MASTER CLEAR, LOAD 2 SOM'S
SYNCH
IDLE!STRIP!DDCMP
JSR PC,TXCHAR ;LOAD A 000 CHAR, TX FIRST SYNCH
000
CHPCHK!8.
JSR PC,STPLU ;STEP THE LU UNTIL 000 HAS BEN TRANSMITTED
17.
JSR PC,READLU ;READ REG 11
BITB #UNRR,REDBYT ;CHK FOR UNR = 1
BNE 16$ ;BR IF UNRR = 1
JSR PC,GETALL ;GET REGS FOR PRINTOUT
:REPORT UNRR NOT SET
ERRDF 14,EM14,ERR7
TRAP C$ERDF
.WORD 14
.WORD EM14
.WORD ERR7
16$:
JMP A7 ;SKIP TO END OF TEST
JSR PC,MSTCLR ;ISSUE MASTER CLEAR
JSR PC,READLU ;READ REG 11
BITB #UNRR,REDBYT ;CHK FOR UNRR = 0
BEQ A7 ;BR IF UNRR = 0
JSR PC,GETALL ;GET REGS FOR PRINTOUT
:REPORT UNRR NOT CLEARED BY OC
ERRDF 15,EM15,ERR7
TRAP C$ERDF
.WORD 15
.WORD EM15
.WORD ERR7
A7:
JSR PC,MSTCLR ;ISSUE CLEAN-UP MASTER CLEAR
ENDTST
L10056:
TRAP C$ETST
```

```
*****
:SBTTL TEST 27 - TRANSMIT CHAR LENGTH TIMING TEST - CHAR MODE, CRC
:*
:* THE LINE UNIT IS PLACED IN CHAR MODE (DDCMP) AND A MESSAGE IS INITIATED
:* WITH AN 8-BIT SYNCH AND A 5-BIT SYNCH CHAR. NEXT, A 000 CHAR IS SENT WITH
:* EACH OF THE FOLLOWING TX CHAR LENGTHS : 5 BITS, 6 BITS, 7 BITS, 8 BITS.
:* (FOR EXAMPLE, A 5-BIT CHAR REQUIRES 5 CLOCK CYCLES TO BE TRANSMITTED). TWO
:* TERMINATING SYNCHS ARE SENT AFTER THE DATA.
:*****
BGNTST
```

CZDMRC M8203 STATIC DIAG #1
CZDMRC.P11 13-MAR-80 14:38

MACY11 30A(1052) 13-MAR-80 14:40 E 12 PAGE 5-22
TEST 27 - TRANSMIT CHAR LENGTH TIMING TEST - CHAR MODE, CRC

SEQ 0147

```
(3) 030302
6182 030302 012737 030510 002362      MOV    #A8,RETADR      ;SET TEST EXIT ADDRESS FOR ERRORS
6183 030310 004737 005540                JSR    PC,INITRN      ;DO MASTER CLR, LOAD 2 SOM'S
6184 030314 000000                        000
6185 030316 000041                        IDLE!DDCMP
6186 030320 012737 000006 002402      MOV    #6,AXNUM       ;SET BYTE NO. = 6 FOR AX3
6187 030326 012737 000000 002374      MOV    #000,WAX15     ;SET DATA FOR AX3-15 = 0
6188 030334 012737 000240 002376      MOV    #TXLEN2!TXLENO,WAX16 ;SET TX LENGTH = 5 FOR AX3-16
6189 030342 004737 004312                JSR    PC,WRITAX      ;LOAD AX3-15,AX3-16
6190 030346 004737 006122                JSR    PC,TXCHAR      ;LOAD 5-BIT 000 CHAR, TX 8-BIT SYNCH
6191 030352 000000                        000
```

```
6193 030354 100010 CHPCHK!8.  
6194 030356 004737 006122 JSR PC,TXCHAR ;LOAD 6-BIT 000 CHAR, TX 5-BIT SYNCH  
6195 030362 000000 000  
6196 030364 000005 5  
6197 030366 012737 000300 002376 MOV #TXLEN2!TXLEN1,WAX16  
6198 030374 004737 004312 JSR PC,WRITAX ;SET TX CHAR LENGTH = 6  
6199 030400 004737 006122 JSR PC,TXCHAR ;LOAD 7-BIT 000 CHAR, TX 5-BIT 000 CHAR  
6200 030404 000000 000  
6201 030406 000005 5  
6202 030410 012737 000340 002376 MOV #TXLEN2!TXLEN1!TXLENO,WAX16  
6203 030416 004737 004312 JSR PC,WRITAX ;SET TX CHAR LENGTH = 7  
6204 030422 004737 006122 JSR PC,TXCHAR ;LOAD 8-BIT 000 CHAR, TX 6-BIT 000 CHAR  
6205 030426 000000 000  
6206 030430 000006 6  
6207 030432 012737 000000 002376 MOV #000,WAX16  
6208 030440 004737 004312 JSR PC,WRITAX ;SET TX CHAR LENGTH = 8  
6209 030444 004737 006122 JSR PC,TXCHAR ;LOAD EOM, TX 7-BIT 000 CHAR  
6210 030450 001000 TXEOM  
6211 030452 000007 7  
6212 030454 004737 006122 JSR PC,TXCHAR ;LOAD EOM, TX 8-BIT 000 CHAR  
6213 030460 001000 TXEOM  
6214 030462 000010 8.  
6215 030464 004737 006122 JSR PC,TXCHAR ;LOAD EOM, TX CRC-16 CHAR  
6216 030470 001000 TXEOM  
6217 030472 000020 16.  
6218 030474 004737 006122 JSR PC,TXCHAR ;LOAD EOM, TX FIRST TERMINATING SYNCH  
6219 030500 001000 TXEOM  
6220 030502 000010 8.  
6221 030504 004737 006274 JSR PC,ENDTRN ;CLEAR TRANSMITTER
```

A8:
ENDTST

L10057:
TRAP C\$ETST

6222 030510
6223 030510
(3) 030510
(3) 030510 104401

6224
6225
6226
6227
6228
6229
6230
6231
6232
6233
6234
6235
6236
6237
6238

```
::*****  
:SBTTL TEST 28 - TRANSMIT CHAR LENGTH TIMING TEST - BIT MODE, CRC  
:*  
:* THE LINE UNIT IS PLACED IN BIT MODE AND A MESSAGE IS INITIATED  
:* WITH 2 FLAG CHARS. NEXT, 2 8-BIT 000 CHARS ARE SENT, FOLLOWED BY 000 CHARS  
:* WITH EACH OF THE FOLLOWING TRANSMITTER CHAR LENGTHS:  
:* 1 BIT, 2 BITS, 3 BITS, 4 BITS, 5 BITS, 6 BITS, 7 BITS, AND 8 BITS.  
:* (FOR EXAMPLE, A 5-BIT CHAR REQUIRES 5 CLOCK CYCLES TO BE TRANSMITTED).  
:* TWO TERMINATING FLAGS ARE SENT AFTER THE DATA.  
:*****  
:BGNTST
```

```
6239 030512  
(3) 030512  
6240 030512 012737 031040 002362 MOV #A9,RETADR ;SET TEST EXIT ADDRESS FOR ERRORS  
6241 030520 004737 005540 JSR PC,INITRN ;DO MASTER CLR, LOAD 2 SOM'S  
6242 030524 000000 000  
6243 030526 000000 000  
6244 030530 004737 006122 JSR PC,TXCHAR ;LOAD FIRST 8-BIT 000 CHAR, TX FIRST FLAG  
6245 030534 000000 000
```

T28::

6246	030536	100010			CHPCHK.8.	
6247	030540	004737	006122		JSR PC,TXCHAR	:LOAD 2ND 8-BIT 000 CHAR, TX 2ND FLAG
6248	030544	000000			000	
6249	030546	000010			8.	
6250	030550	004737	006122		JSR PC,TXCHAR	:LOAD 1-BIT 000 CHAR, TX FIRST 8-BIT 000 CHAR
6251	030554	000000			000	
6252	030556	000010			8.	
6253	030560	012737	000006	002402	MOV #6,AXNUM	:SET BYTE NO. = 6 FOR AX3
6254	030566	012737	000000	002374	MOV #000,WAX15	:SET DATA FOR AX3-15 = 0
6255	030574	012737	000040	002376	MOV #TXLENO,WAX16	:SET TX CHAR LENGTH = 1 FOR AX3-16
6256	030602	004737	004312		JSR PC,WRITAX	:LOAD AX3-15,AX3-16
6257	030606	004737	006122		JSR PC,TXCHAR	:LOAD 2-BIT 000 CHAR, TX 2ND 8-BIT 000 CHAR
6258	030612	000000			000	
6259	030614	000010			8.	
6260	030616	012737	000100	002376	MOV #TXLEN1,WAX16	
6261	030624	004737	004312		JSR PC,WRITAX	:SET TX CHAR LENGTH = 2
6262	030630	004737	006122		JSR PC,TXCHAR	:LOAD 3-BIT 000 CHAR, TX 1-BIT 000 CHAR
6263	030634	000000			000	
6264	030636	000001			1	
6265	030640	012737	000140	002376	MOV #TXLEN1!TXLENO,WAX16	
6266	030646	004737	004312		JSR PC,WRITAX	:SET TX CHAR LENGTH = 3
6267	030652	004737	006122		JSR PC,TXCHAR	:LOAD 4-BIT 000 CHAR, TX 2-BIT 000 CHAR
6268	030656	000000			000	
6269	030660	000002			2	
6270	030662	012737	000200	002376	MOV #TXLEN2,WAX16	
6271	030670	004737	004312		JSR PC,WRITAX	:SET TX CHAR LENGTH = 4
6272	030674	004737	006122		JSR PC,TXCHAR	:LOAD 5-BIT 000 CHAR, TX 3-BIT 000 CHAR
6273	030700	000000			000	
6274	030702	000003			3	
6275	030704	012737	000240	002376	MOV #TXLEN2!TXLENO,WAX16	
6276	030712	004737	004312		JSR PC,WRITAX	:SET TX CHAR LENGTH = 5
6277	030716	004737	006122		JSR PC,TXCHAR	:LOAD 6-BIT 000 CHAR, TX 4-BIT 000 CHAR
6278	030722	000000			000	
6279	030724	000004			4	
6280	030726	012737	000300	002376	MOV #TXLEN2!TXLEN1,WAX16	
6281	030734	004737	004312		JSR PC,WRITAX	:SET TX CHAR LENGTH = 6
6282	030740	004737	006122		JSR PC,TXCHAR	:LOAD 7-BIT 000 CHAR, TX 5-BIT 000 CHAR
6283	030744	000000			000	
6284	030746	000005			5	
6285	030750	012737	000340	002376	MOV #TXLEN2!TXLEN1!TXLENO,WAX16	
6286	030756	004737	004312		JSR PC,WRITAX	:SET TX CHAR LENGTH = 7
6287	030762	004737	006122		JSR PC,TXCHAR	:LOAD 8-BIT 000 CHAR, TX 6-BIT 000 CHAR
6288	030766	000000			000	
6289	030770	000006			6	
6290	030772	012737	000000	002376	MOV #000,WAX16	
6291	031000	004737	004312		JSR PC,WRITAX	:SET TX CHAR LENGTH = 8
6292	031004	004737	006122		JSR PC,TXCHAR	:LOAD EOM, TX 7-BIT 000 CHAR
6293	031010	001000			TXEOM	
6294	031012	000007			7	
6295	031014	004737	006122		JSR PC,TXCHAR	:LOAD EOM, TX 8-BIT 000 CHAR, CRC-CCITT-1 CHAR
6296	031020	001000			TXEOM	
6297	031022	000031			25.	
6298	031024	004737	006122		JSR PC,TXCHAR	:LOAD EOM, TX FIRST TERMINATING FLAG
6299	031030	001000			TXEOM	
6300	031032	000010			8.	
6301	031034	004737	006274		JSR PC,ENDTRN	:CLEAR TRANSMITTER

6302 031040
 6303 031040
 (3) 031040
 (3) 031040 104401

A9:
 ENDTST

L10060: TRAP C\$ETST

6304
 6305
 6306
 6307
 6308
 6309
 6310
 6311
 6312
 6313
 6314
 6315
 6316
 6317

```

:*****
.SBTTL TEST 29 - TXDATA BIT TEST - CHAR MODE, CRC
:*
:* THE LINE UNIT IS INITIALIZED AND A MSG IS INITIATED (USING STEPLU) WITH CRC-
:* 16 SELECTED IN CHAR MODE. TWO SYNCHS, 000,125,252,377,000, AND 2 TERMINATING
:* SYNCHS ARE THEN SENT. THE PROGRAM CHECKS EACH BIT OF THE TRANSMITTED
:* DATA CHARS, BY MONITORING TXDATA (REG 17) AS THE DATA IS CLOCKED OUT OF
:* THE USYRT TRANSMITTER.
:*****
    
```

6318 031042
 (3) 031042
 6319 031042 004737 005540
 6320 031046 000226
 6321 031050 000011
 6322 031052 012701 003224
 6323 031056 010103
 6324 031060 012137 002422
 6325 031064 004737 005174
 6326 031070 020127 003242
 6327 031074 103771
 6328 031076 004737 005146
 6329 031102 004737 005254
 6330 031106 100020
 6331 031110 011337 031120
 6332 031114 004737 011176
 6333 031120 000000
 6334 031122 062703 000002
 6335 031126 020327 003236
 6336 031132 103766
 6337 031134 004737 003576
 6338 031140
 (3) 031140
 (3) 031140 104401

```

BGNTST
:*****
T29::
JSR PC,INITRN ;DO MASTER CLR, LOAD 2 SOM'S
SYNCH
STRIP!DDCMP
MOV #MSG1+4,R1 ;GET POINTER TO DATA
MOV R1,R3
3$: MOV (R1)+,TXWORD
JSR PC,LDTXSI ;LOAD A DATA CHAR INTO TX SILO
CMP R1,#MSG1+18. ;SEE IF ALL CHARS LOADED YET
BLO 3$ ;BR IF NOT YET
JSR PC,WAIT50 ;WAIT FOR SILO TO RIPPLE
JSR PC,STPLU ;CLOCK LU UNTIL SYNCHS ARE TX'D
CHPCHK!16.
6$: MOV (R3),8$ ;GET EXPECTED DATA CHAR
JSR PC,CKTBIT ;CHECK TXDATA FOR CHAR BITS
8$: .WORD 0 ;EXPECTED CHAR GOES HERE
ADD #2,R3 ;INCR PATTERN POINTER
CMP R3,#MSG1+14. ;SEE IF ALL CHARS CHECKED YET
BLO 6$ ;BR IF NOT YET
16$: JSR PC,MSTCLR ;ISSUE MASTER CLEAR TO CLEAN UP
ENDTST
    
```

L10061: TRAP C\$ETST

6339
 6340
 6341
 6342
 6343
 6344
 6345
 6346
 6347
 6348
 6349
 6350
 6351
 6352

```

:*****
.SBTTL TEST 30 - USYRT RECEIVER MSG TEST - CHAR MODE, CRC
:*
:* THE LINE UNIT IS INITIALIZED AND A MESSAGE IS INITIATED (USING STEPLU) WITH
:* LULP (REG 12) SET TO LOOP THE DATA INTERNALLY IN THE USYRT, AND WITH CRC-16
:* SELECTED. TWO SYNCHS, 000,125,252,377,000, AND FOUR TERMINATING SYNCHS ARE
:* SENT. THE PROGRAM MONITORS IACT, AND THE RCV'D CHARS AND CRC BYTES ARE READ
:* FROM AX0-15 AND COMPARED TO EXPECTED VALUES. THE PROGRAM THEN CHECKS FOR IACT
:* STILL SET AFTER THE MESSAGE.
    
```

```
6353 ;*****
6354 031142 BGNTST
(3) 031142
6355 031142 012737 031504 002362 MOV #24$,RETADR ;SET TEST EXIT ADDRESS FOR ERRORS
6356 031150 004737 003576 JSR PC,MSTCLR ;ISSUE MASTER CLEAR
6357 031154 004737 010640 JSR PC,SETUP ;PROGRAM THE USYRT
6358 031160 000226 SYNCH
6359 031162 000013 STRIP! IERR!DDCMP
6360 031164 000000 000
6361 031166 000000 000
6362 031170 012737 000012 002400 MOV #12,REGNUM ;SET LU REG NO. = 12
6363 031176 005037 002402 CLR AXNUM ;SET AX BYTE NO. = 0 FOR AX0
6364 031202 112737 000040 002366 MOVB #LULP,WRIBYT
6365 031210 004737 003750 JSR PC,WRITLU ;SET LULP IN REG 12
6366 031214 012701 003220 MOV #MSG1,R1 ;GET POINTER TO MSG DATA TABLE
6367 031220 012137 002422 3$: MOV (R1)+,TXWORD ;GET CHAR TO BE LOADED
6368 031224 004737 005174 JSR PC,LDTXSI ;LOAD CHAR INTO TX SILO
6369 031230 020127 003246 CMP R1,#MSG1+22. ;SEE IF ALL MSG CHARS LOADED YET
6370 031234 103771 BLO 3$ ;BR IF NOT YET
6371 031236 004737 005146 JSR PC,WAIT50 ;ALLOW DATA TO RIPPLE IN SILO
6372 031242 004737 005254 JSR PC,STPLU ;CLOCK LU FOR 40 CYCLES (UNTIL FIRST
6373 031246 000050 40. ; DATA CHAR IS ABOUT TO BE RECEIVED)
6374 031250 004737 006714 JSR PC,IACTIV ;CHK IACT = 0
6375 031254 000000 0
6376 031256 004737 005254 JSR PC,STPLU ;CLOCK LU UNTIL 1ST DATA CHAR IS RCV'D
6377 031262 000006 6
6378 031264 012701 003224 MOV #MSG1+4,R1
6379 031270 004737 006714 10$: JSR PC,IACTIV ;CHK IACT = 1
6380 031274 000001 1
6381 031276 004737 004124 JSR PC,READAX ;READ AX0
6382 031302 023721 002370 CMP RAX15,(R1)+ ;COMPARE RCV'D CHAR TO EXPECTED
6383 031306 001415 BEQ 12$ ;BR IF RCV'D DATA OK
6384 031310 016137 177776 002404 MOV -2(R1),GOODAT ;GET EXPECTED DATA
6385 031316 013737 002370 002406 MOV RAX15,BADDAT ;GET ACTUAL DATA
6386 031324 004737 004526 JSR PC,GETALL ;GET REGS FOR PRINTOUT
6387 ;REPORT INCORRECT DATA CHAR RCV'D
6388 ERRDF 26,EM26,ERR3
(4) 031330 104455
(5) 031332 000032 TRAP C$ERDF
(5) 031334 013071 .WORD 26
(5) 031336 015142 .WORD EM26
6389 031340 000461 .WORD ERR3
6390 031342 004737 005254 12$: BR 24$
6391 031346 000010 JSR PC,STPLU ;CLOCK LU 8 CYCLES
6392 031350 020127 003236 8.
6393 031354 103745 CMP R1,#MSG1+14. ;SEE IF CHECKING HI CRC BYTE YET
6394 031356 004737 006714 BLO 10$ ;BR IF NOT YET
6395 031362 000001 JSR PC,IACTIV ;CHK IACT = 1
6396 031364 004737 004124 1
6397 031370 123727 002370 000160 JSR PC,READAX ;READ AX0
6398 031376 001415 CMPB RAX15,#160 ;CMP RCV'D CHAR TO EXPECTED HI CRC BYTE
6399 031400 012737 000160 002404 BEQ 16$ ;BR IF HI CRC BYTE RCV'D OK
6400 031406 013737 002370 002406 14$: MOV #160,GOODAT ;GET EXPECTED DATA
6401 031414 004737 004526 MOV RAX15,BADDAT ;SET ACTUAL DATA
6402 ;REPORT INCORRECT CRC BYTE RCV'D
6403 ERRDF 27,EM27,ERR3
```

(4)	031420	104455						TRAP	C\$ERDF
(5)	031422	000033						.WORD	27
(5)	031424	013123						.WORD	EM27
(5)	031426	015142						.WORD	ERR3
6404	031430	000425							
6405	031432	004737	005254	16\$:	BR	24\$			
6406	031436	000010			JSR	PC,STPLU			:CLOCK LU FOR 8 CYCLES
6407	031440	004737	006714		8.				
6408	031444	000001			JSR	PC,IACTIV			:CHK IACT = 1
6409	031446	012737	000034	002404	1				
6410	031454	004737	004124		MOV	#034,GOODAT			:GET EXPECTED LO CRC BYTE
6411	031460	123727	002370	000034	JSR	PC,READAX			:READ AX0
6412	031466	001347			CMPB	RAX15,#034			:CMP RCV'D CHAR TO EXPECTED LO CRC BYTE
6413	031470	004737	005254		BNE	14\$:BR IF LO CRC INCORRECT
6414	031474	000010			JSR	PC,STPLU			:CLOCK LU 8 CYCLES
6415	031476	004737	006714		8.				
6416	031502	000001			JSR	PC,IACTIV			:CHK IACT STILL = 1
6417	031504	004737	003576	24\$:	1				
6418	031510			ENDTST	JSR	PC,MSTCLR			:ISSUE CLEAN-UP MASTER CLEAR
(3)	031510								
(3)	031510	104401						L10062:	TRAP C\$ETST

```

:*****
.SBTL TEST 31 - USYRT RECEIVER MSG TEST - BIT MODE, CRC
:*
:* THE LINE UNIT IS INITIALIZED AND A MESSAGE IS INITIATED (USING STEPLU) WITH
:* LULP (REG 12) SET TO LOOP THE DATA INTERNALLY IN THE USYRT, AND WITH CRC-
:* CCITT-1. TWO FLAGS, 000,125,252,377,000, AND TWO TERMINATING FLAGS ARE THEN
:* SENT. THE PROGRAM MONITORS IACT, RSOM, AND THE RCV'D CHARS ARE READ
:* FROM AX0-15 AND COMPARED TO EXPECTED VALUES. THE PROGRAM THEN CHECKS FOR
:* IACT = 0, SETS IC TO CLEAR THE RECEIVER, AND CHECKS FOR IACT STILL = 0.
:*****
BGNTST
  
```

6434	031512								T31::
(3)	031512								
6435	031512	012737	032046	002362	MOV	#24\$,RETADR			:SET TEST EXIT ADDRESS FOR ERRORS
6436	031520	004737	003576		JSR	PC,MSTCLR			:ISSUE MASTER CLEAR
6437	031524	004737	010640		JSR	PC,SETUP			:PROGRAM THE USYRT
6438	031530	000000			000				
6439	031532	000002			IERR				
6440	031534	000000			000				
6441	031536	000000			000				
6442	031540	012737	000012	002400	MOV	#12,REGNUM			:SET LU REG NO. = 12
6443	031546	005037	002402		CLR	AXNUM			:SET AX BYTE NO. = 0 FOR AX0
6444	031552	112737	000040	002366	MOVB	#LULP,WRIBYT			
6445	031560	004737	003750		JSR	PC,WRITLU			:SET LULP IN REG 12
6446	031564	012701	003220		MOV	#MSG1,R1			:GET POINTER TO MSG DATA TABLE
6447	031570	012137	002422		3\$:	MOV	(R1)+,TXWORD		:GET CHAR TO BE LOADED
6448	031574	004737	005174		JSR	PC,LDTXSI			:LOAD CHAR INTO TX SILO
6449	031600	020127	003242		CMP	R1,#MSG1+18.			:SEE IF ALL MSG CHARS LOADED YET
6450	031604	103771			BLO	3\$:BR IF NOT YET
6451	031606	004737	005146		JSR	PC,WAIT50			:ALLOW DATA TO RIPPLE IN SILO
6452	031612	004737	005254		JSR	PC,STPLU			:CLOCK LU FOR 50 CYCLES (UNTIL FIRST

```
6453 031616 000062          50.          : DATA CHAR IS ABOUT TO BE RECEIVED)
6454 031620 004737 006714   JSR          PC,IACTIV      ;CHK IACT = 0
6455 031620 000000          0             :
6456 031626 004737 007102   JSR          PC,RSEOM      ;CHK RSOM = 0, REOM = 0
6457 031632 000000          0             :
6458 031634 004737 005254   JSR          PC,STPLU      ;CLOCK LU UNTIL 1ST DATA CHAR IS RCV'D
6459 031640 000006          6             :
6460 031642 012701 003224   MOV          #MSG1+4,R1
6461 031646 020127 003224   5$: CMP          R1,#MSG1+4  ;SEE IF 1ST CHAR RCV'D
6462 031652 001007          6$           ;BR IF NO
6463 031654 004737 006714   JSR          PC,IACTIV      ;CHK IACT = 1
6464 031660 000001          1             :
6465 031662 004737 007102   JSR          PC,RSEOM      ;CHK RSOM = 1, REOM = 0
6466 031666 000001          1             :
6467 031670 000420          9$           :
6468 031672 020127 003234   6$: CMP          R1,#MSG1+12. ;SEE IF LAST CHAR RCV'D
6469 031676 001007          8$           ;BR IF NO
6470 031700 004737 006714   JSR          PC,IACTIV      ;CHK FOR IACT = 0
6471 031704 000000          0             :
6472 031706 004737 007102   JSR          PC,RSEOM      ;CHK RSOM = 0, REOM = 0
6473 031712 000000          0             :
6474 031714 000406          9$           :
6475 031716 004737 006714   8$: JSR          PC,IACTIV      ;CHK FOR IACT = 1
6476 031722 000001          1             :
6477 031724 004737 007102   JSR          PC,RSEOM      ;CHK RSOM = 0, REOM = 0
6478 031730 000000          0             :
6479 031732 004737 004124   9$: JSR          PC,READAX     ;READ AX0
6480 031736 023721 002370   CMP          RAX15,(R1)+   ;COMPARE RCV'D CHAR TO EXPECTED
6481 031742 001415          12$          ;BR IF RCV'D DATA OK
6482 031744 016137 177776   002404      MOV          -2(R1),GOODAT ;GET EXPECTED DATA
6483 031752 013737 002370   002406      MOV          RAX15,BADDAT  ;GET ACTUAL DATA
6484 031760 004737 004526   JSR          PC,GETALL     ;GET REGS FOR PRINTOUT
6485          ;REPORT INCORRECT DATA CHAR RCV'D
6486          ERRDF 26,EM26,ERR3
(4) 031764 104455          TRAP        C$ERDF
(5) 031766 000032          .WORD      26
(5) 031770 013071          .WORD      EM26
(5) 031772 015142          .WORD      ERR3
6487 031774 000424          24$         :
6488 031776 004737 005254   12$: BR          24$
6489 032002 000010          8.          :
6490 032004 020127 003236   CMP          R1,#MSG1+14.  ;SEE IF ALL DATA CHARS CHECKED YET
6491 032010 103716          5$           ;BR IF NOT YET
6492 032012 004737 006714   JSR          PC,IACTIV      ;CHK IACT = 0
6493 032016 000000          0             :
6494 032020 004737 007102   JSR          PC,RSEOM      ;CHK RSOM = 0, REOM = 0
6495 032024 000000          0             :
6496 032026 012737 000200   002366      MOV          #IC,WRIBYT
6497 032034 004737 003750   JSR          PC,WRITLU     ;SET IC (INPUT CLEAR) IN REG 12
6498 032040 004737 006714   JSR          PC,IACTIV      ;CHK IACT = 0
6499 032044 000000          0             :
6500 032046 004737 003576   24$: JSR          PC,MSTCLR    ;ISSUE CLEAN-UP MASTER CLEAR
6501 032052          ENDTST
(3) 032052          L10063:
(3) 032052 104401          TRAP        C$ETST
6502
```


6503
6504
6505
6506
6507
6508
6509
6510
6511
6512
6513
6514
6515
6516
6517
(3)
6518
6519
6520
6521
6522
6523
6524
6525
6526
6527
6528
6529
6530
6531
6532
6533
6534
6535
6536
6537
6538
6539
6540
6541
6542
6543
6544
6545
6546
6547
6548
6549
6550
6551
(4)
(5)
(5)
(5)
6552
6553

032054
032054 012737 032316 002362
032062 004737 003576
032066 004737 010640
032072 000226
032074 000313
032076 000000
032100 000000
032102 012737 000012 002400
032110 005037 002402
032114 112737 000040 002366
032122 004737 003750
032126 012701 003220
032132 012137 002422
032136 004737 005174
032142 020127 003242
032146 103771
032150 004737 005146
032154 004737 005254
032160 000030
032162 004737 006714
032166 000000
032170 004737 005254
032174 000006
032176 012701 003224
032202 004737 006714
032206 000001
032210 004737 004124
032214 023721 002370
032220 001415
032222 016137 177776 002404
032230 013737 002370 002406
032236 004737 004526
032242
(4) 032242 104455
(5) 032244 000032
(5) 032246 013071
(5) 032250 015142
032252 000421
032254 004737 005254

```
*****  
:SBTTL TEST 32 - USYRT RECEIVER MSG TEST - CHAR MODE, NO CRC  
:  
* THE LINE UNIT IS INITIALIZED AND A MESSAGE IS INITIATED (USING STEPLU) WITH  
* LULP (REG 12) SET TO LOOP THE DATA INTERNALLY IN THE USYRT, AND WITH NO  
* ERROR DETECTION. TWO SYNCHS, 000,125,252,377,000, AND TWO SYNCHS ARE  
* THEN SENT. THE PROGRAM MONITORS IACT, AND THE RECEIVED CHARS ARE READ FROM  
* AX0-15 AND COMPARED TO EXPECTED VALUES. THE PROGRAM THEN CHECKS FOR IACT  
* STILL = 0, SETS IC TO CLEAR THE RECEIVER, AND CHECKS FOR IACT = 0.  
:*****
```

```
BGNTST  
T32::  
MOV #24$,RETADR ;SET TEST EXIT ADDRESS FOR ERRORS  
JSR PC,MSTCLR ;ISSUE MASTER CLEAR  
JSR PC,SETUP ;PROGRAM THE USYRT  
SYNCH  
CRC2!CRC1!STRIP!IERR!DDCMP  
000  
000  
MOV #12,REGNUM ;SET LU REG NO. = 12  
CLR AXNUM ;SET AX BYTE NO. = 0 FOR AX0  
MOVB #LULP,WRIBYT  
JSR PC,WRITLU ;SET LULP IN REG 12  
MOV #MSG1,R1 ;GET POINTER TO MSG DATA TABLE  
3$: MOV (R1)+,TXWORD ;GET CHAR TO BE LOADED  
JSR PC,LDTXSI ;LOAD CHAR INTO TX SILO  
CMP R1,#MSG1+18. ;SEE IF ALL MSG CHARS LOADED YET  
BLO 3$ ;BR IF NOT YET  
JSR PC,WAIT50 ;ALLOW DATA TO RIPPLE IN SILO  
JSR PC,STPLU ;CLOCK LU FOR 24 CYCLES (UNTIL IRST  
24. ; DATA CHAR IS ABOUT TO BE RECEIVED)  
JSR PC,IACTIV ;CHK IACT = 0  
0  
JSR PC,STPLU ;CLOCK LU UNTIL 1ST DATA CHAR IS RCV'  
6  
MOV #MSG1+4,R1  
10$: JSR PC,IACTIV ;CHK IACT = 1  
1  
JSR PC,READAX ;READ AX0  
CMP RAX15,(R1)+ ;COMPARE RCV'D CHAR TO EXPECTED  
BEQ 12$ ;BR IF RCV'D DATA OK  
MOV -2(R1),GOODAT ;GET EXPECTED DATA  
MOV RAX15,BADDAT ;GET ACTUAL DATA  
JSR PC,GETALL ;GET REGS FOR PRINTOUT  
;REPORT INCORRECT DATA CHAR RCV'D  
ERRDF 26,EM26,ERR3  
  
TRAP C$ERDF  
.WORD 26  
.WORD EM26  
.WORD ERR3  
  
12$: BR 24$  
JSR PC,STPLU ;CLOCK LU 8 CYCLES
```

```
6554 032260 000010      8.  
6555 032262 020127 003236  CMP      R1,#MSG1+14.  ;SEE IF ALL 5 DATA CHARS RCV'D YET  
6556 032266 103745      BLO      10$          ;BR IF NOT YET  
6557 032270 004737 006714  JSR      PC,IACTIV   ;CHK FOR IACT STILL = 1  
6558 032274 000001      1  
6559 032276 012737 000200 002366  MOV      #IC,WRIBYT  
6560 032304 004737 003750      JSR      PC,WRITLU   ;SET IC (INPUT CLEAR) IN REG 12  
6561 032310 004737 006714      JSR      PC,IACTIV   ;CHK IACT = 0  
6562 032314 000000      0  
6563 032316 004737 003576 24$: JSR      PC,MSTCLR  ;ISSUE CLEAN-UP MASTER CLEAR  
6564 032322      ENDTST  
  (3) 032322  
  (3) 032322 104401      L10064: TRAP C$ETST
```

6565
6566
6567
6568
6569
6570
6571
6572
6573
6574
6575
6576
6577
6578
6579

```
*****  
:SBTTL TEST 33 - USYRT RECEIVER MSG TEST - BIT MODE, NO CRC  
: *  
: * THE LINE UNIT IS INITIALIZED AND A MESSAGE IS INITIATED (USING STEPLU) WITH  
: * LULP (REG 12) SET TO LOOP THE DATA INTERNALLY IN THE USYRT, AND WITH ERROR  
: * DETECTION INHIBITED. TWO FLAGS, 000,125,252,377,000, AND TWO TERMINATING FLAGS  
: * ARE THEN SENT. THE PROGRAM MONITORS IACT, RSOM, AND THE RCV'D CHARS ARE  
: * READ FROM AX0-15 AND COMPARED TO EXPECTED VALUES. THE PROGRAM THEN CHECKS FOR  
: * IACT = 0, SETS IC TO CLEAR THE RECEIVER, AND CHECKS FOR IACT STILL - 0.  
:*****  
BGNTST
```

```
6580 032324      (3) 032324  
6581 032324 012737 032660 002362  MOV      #24$,RETADR  T33::  
6582 032332 004737 003576      JSR      PC,MSTCLR   ;ISSUE MASTER CLEAR  
6583 032336 004737 010640      JSR      PC,SETUP    ;PROGRAM THE USYRT  
6584 032342 000000      000  
6585 032344 000302      CRC2!CRC1!IERR  
6586 032346 000000      000  
6587 032350 000000      000  
6588 032352 012737 000012 002400  MOV      #12,REGNUM  ;SET LU REG NO. = 12  
6589 032360 005037 002402      CLR      AXNUM       ;SET AX BYTE NO. = 0 FOR AX0  
6590 032364 112737 000040 002366  MOVB     #LULP,WRIBYT  
6591 032372 004737 003750      JSR      PC,WRITLU   ;SET LULP IN REG 12  
6592 032376 012701 003220      MOV      #MSG1,R1    ;GET POINTER TO MSG DATA TABLE  
6593 032402 012137 002422 3$: MOV      (R1)+,TXWORD ;GET CHAR TO BE LOADED  
6594 032406 004737 005174      JSR      PC,LDTXSI   ;LOAD CHAR INTO TX SILO  
6595 032412 020127 003242      CMP      R1,#MSG1+18. ;SEE IF ALL MSG CHARS LOADED YET  
6596 032416 103771      BLO      3$          ;BR IF NOT YET  
6597 032420 004737 005146      JSR      PC,WAIT50   ;ALLOW DATA TO RIPPLE IN SILO  
6598 032424 004737 005254      JSR      PC,STPLU    ;CLOCK LU FOR 33 CYCLES (UNTIL FIRST  
6599 032430 000041      33.                ; DATA CHAR IS ABOUT TO BE RECEIVED)  
6600 032432 004737 006714      JSR      PC,IACTIV   ;CHK IACT = 0  
6601 032436 000000      0  
6602 032440 004737 007102      JSR      PC,RSEOM    ;CHK RSOM = 0, REOM = 0  
6603 032444 000000      0  
6604 032446 004737 005254      JSR      PC,STPLU    ;CLOCK LU UNTIL 1ST DATA CHAR IS RCV'D  
6605 032452 000006      6  
6606 032454 012701 003224      MOV      #MSG1+4,R1
```

```
6607 032460 020127 003224 5$: CMP R1,#MSG1+4 ;SEE IF 1ST CHAR RCV'D
6608 032464 001007 BNE 6$ ;BR IF NO
6609 032466 004737 006714 JSR PC,IACTIV ;CHK IACT = 1
6610 032472 000001 1
6611 032474 004737 007102 JSR PC,RSEOM ;CHK RSOM = 1, REOM = 0
6612 032500 000001 1
6613 032502 000420 BR 9$
6614 032504 020127 003234 6$: CMP R1,#MSG1+12. ;SEE IF LAST CHAR RCV'D
6615 032510 001007 BNE 8$ ;BR IF NO
6616 032512 004737 006714 JSR PC,IACTIV ;CHK FOR IACT = 0
6617 032516 000000 0
6618 032520 004737 007102 JSR PC,RSEOM ;CHK RSOM = 0, REOM = 0
6619 032524 000000 0
6620 032526 000406 BR 9$
6621 032530 004737 006714 8$: JSR PC,IACTIV ;CHK FOR IACT = 1
6622 032534 000001 1
6623 032536 004737 007102 JSR PC,RSEOM ;CHK RSOM = 0, REOM = 0
6624 032542 000000 0
6625 032544 004737 004124 9$: JSR PC,READAX ;READ AX0
6626 032550 023721 002370 CMP RAX15,(R1)+ ;COMPARE RCV'D CHAR TO EXPECTED
6627 032554 001415 BEQ 12$ ;BR IF RCV'D DATA OK
6628 032556 016137 177776 002404 MOV -2(R1),GOODAT ;GET EXPECTED DATA
6629 032564 013737 002370 002406 MOV RAX15,BADAT ;GET ACTUAL DATA
6630 032572 004737 004526 JSR PC,GETALL ;GET REGS FOR PRINTOUT
6631 ;REPORT INCORRECT DATA CHAR RCV'D
6632 032576 ERRDF 26,EM26,ERR3
(4) 032576 104455 TRAP C$ERDF
(5) 032600 000032 .WORD 26
(5) 032602 013071 .WORD EM26
(5) 032604 015142 .WORD ERR3
6633 032606 000424 BR 24$
6634 032610 004737 005254 12$: JSR PC,STPLU ;CLOCK LU 8 CYCLES
6635 032614 000010 8.
6636 032616 020127 003236 CMP R1,#MSG1+14. ;SEE IF ALL DATA CHARS CHECKED YET
6637 032622 103716 BLO 5$ ;BR IF NOT YET
6638 032624 004737 006714 JSR PC,IACTIV ;CHK IACT = 0
6639 032630 000000 0
6640 032632 004737 007102 JSR PC,RSEOM ;CHK RSOM = 0, REOM = 0
6641 032636 000000 0
6642 032640 012737 000200 002366 MOV #IC,WRIBYT ;SET IC (INPUT CLEAR) IN REG 12
6643 032646 004737 003750 JSR PC,WRITLU ;CHK IACT = 0
6644 032652 004737 006714 JSR PC,IACTIV
6645 032656 000000 0
6646 032660 004737 003576 24$: JSR PC,MSTCLR ;ISSUE CLEAN-UP MASTER CLEAR
6647 032664 ENDTST
(3) 032664 L10065: TRAP C$ETST
(3) 032664 104401
6648
6649
6650
6651
6652
6653
6654 ;*****
6655 ;SBTTL TEST 34 - SILO-DISABLED TRANSMITTER LOAD TEST
6656 ;*
```

* THIS TEST DISABLES THE SILOS, LOADS A 125 CHARACTER INTO THE TX SILO, AND

6657
6658
6659
6660 032666
(3) 032666
6661 032666 004737 003576
6662 032672 012737 000014 002400
6663 032700 012737 000040 002366
6664 032706 004737 003750
6665 032712 012737 000040 002426
6666 032720 012737 000125 002422
6667 032726 004737 005174
6668 032732 012737 000002 002402
6669 032740 004737 004124
6670 032744 123727 002370 000000
6671 032752 001414
6672 032754 012737 000000 002404
6673 032762 013737 002370 002406
6674 032770 004737 004526
6675
6676 032774
(4) 032774 104455
(5) 032776 000003
(5) 033000 012222
(5) 033002 015142
6677 033004 005037 002426
6678 033010 004737 003576
6679 033014
(3) 033014
(3) 033014 104401
6680
6681
6682
6683
6684
6685
6686
6687
6688
6689
6690
6691
6692
6693
6694
6695
6696 033016
(3) 033016
6697 033016 012737 033716 002362
6698 033024 004737 005540
6699 033030 000000
6700 033032 000000
6701 033034 004737 003576
6702 033040 004737 010640
6703 033044 000000
6704 033046 000302

```
;* READS AX1-15 AND CHECKS THAT THE DATA DID NOT GET LOADED INTO THE USYRT TX
;* BUFFER.
;*****
BGNTST
T34::
JSR PC,MSTCLR ;ISSUE MASTER CLEAR
MOV #14,REGNUM ;SET REG NO. = 14
MOV #DISSI,WRIBYT ;SET DISSI BIT
JSR PC,WRITLU
MOV #DISSI,DISILO ;SET DISABLE SILO FLAG
MOV #125,TXWORD ;LOAD 125 INTO TX SILO
JSR PC,LDTXSI
MOV #2,AXNUM ;SET REG NO. FOR AX1
JSR PC,READAX ;READ AX1-15, AX1-16
CMPB RAX15,#000 ;CHECK FOR AX1-15 UNCHANGED
BEQ 3$ ;BR IF UNCHANGED
MOV #000,GOODAT ;GET EXPECTED DATA
MOV RAX15,BADDAT ;GET ACTUAL DATA
JSR PC,GETALL ;GET REGS FOR PRINTOUT
;REPORT REG MISCOMPARE
ERRDF 3,EM3,ERR3
TRAP C$ERDF
.WORD 3
.WORD EM3
.WORD ERR3
3$: CLR DISILO ;CLEAR DISABLE SILO FLAG
JSR PC,MSTCLR ;ISSUE MASTER CLEAR TO CLEAN UP
ENDTST
L10066: TRAP C$ETST
```

```
;*****
.SBTTL TEST 35 - SILO-DISABLED MESSAGE TEST - BIT MODE, NO CRC
;*
;* THE LINE UNIT IS INITIALIZED AND A MESSAGE IS INITIATED (USING STEPLU)
;* WITH LULP (REG 12) SET TO LOOP THE DATA INTERNALLY IN THE USYRT, WITH SILO
;* DISABLE SET, AND WITH NO ERROR DETECTION. TWO FLAGS, 000,125,252, AND
;* TERMINATING FLAGS ARE THEN SENT BY LOADING THE TRANSMITTED CHARS INTO
;* REG AX1. THE PROGRAM MONITORS OACT, IACT, RSOM, REOM, ORDY, OCOR, ICIR,
;* IRDY, AND THE RECEIVED CHARS ARE READ FROM AX0 AND COMPARED TO EXPECTED
;* VALUES.
;*****
BGNTST
```

```
T35::
MOV #18$,RETADR ;SET TEST EXIT ADDRESS FOR ERRORS
JSR PC,INTRN ;FIND OUT WHICH USYRT CHIP
0
0
JSR PC,MSTCLR ;ISSUE MASTER CLEAR
JSR PC,SETUP ;PROGRAM THE USYRT
000
CRC2!CRC1!IERR
```

6705	033050	000000		000		
6706	033052	000000		000		
6707	033054	012737	000014	002400	MOV	#14,REGNUM ;SET REG NO. = 14
6708	033062	012737	000140	002366	MOV	#TXEN!DISSI,WRIBYT
6709	033070	004737	003750		JSR	PC,WRITLU ;SET TXEN AND DISSI IN REG 14
6710	033074	012737	000140	002426	MOV	#TXEN!DISSI,DISILO ;SET DISABLE SILO FLAG
6711	033102	012737	000012	002400	MOV	#12,REGNUM ;SET LU REG NO. = 12
6712	033110	112737	000040	002366	MOVB	#LULP,WRIBYT
6713	033116	004737	003750		JSR	PC,WRITLU ;SET LULP IN REG 12
6714	033122	012701	003220		MOV	#MSG1,R1 ;GET POINTER TO MSG
6715	033126	004737	004662		JSR	PC,OSIRDY ;CHK ORDY = 1
6716	033132	000001			1	
6717	033134	012737	000002	002402	MOV	#2,AXNUM ;SET AX BYTE NO. FOR AX1
6718	033142	112137	002374		MOVB	(R1)+,WAX15 ;GET A CHAR
6719	033146	112137	002376		MOVB	(R1)+,WAX16
6720	033152	004737	004312		JSR	PC,WRITAX ;LOAD CHAR INTO USYRT TX BUFFER
6721	033156	004737	004662		JSR	PC,OSIRDY ;CHK ORDY = 0
6722	033162	000000			0	
6723	033164	004737	005352		JSR	PC,OACTIV ;CHK OACT = 0
6724	033170	000000			0	
6725	033172	004737	005254		JSR	PC,STPLU ;CLOCK LU FOR 3 CYCLES
6726	033176	000003			3	
6727	033200	004737	005352		JSR	PC,OACTIV ;CHK OACT = 1
6728	033204	000001			1	
6729	033206	012703	000004		MOV	#4,R3 ;INIT COUNTER
6730	033212	112137	002374	4\$:	MOVB	(R1)+,WAX15 ;GET ANOTHER CHAR
6731	033216	112137	002376		MOVB	(R1)+,WAX16
6732	033222	020327	000001		CMP	R3,#1 ;SEE IF LOADING LAST DATA CHAR YET
6733	033226	001006			BNE	5\$;BR IF NOT
6734	033230	005737	002430		TST	CHPTYP ;SEE IF SIG USYRT
6735	033234	001403			BEQ	5\$;BR IF YES
6736	033236	112737	000002	002376	MOVB	#TEOM,WAX16 ;SET TEOM WITH LAST DATA CHAR
6737	033244	004737	004662	5\$:	JSR	PC,OSIRDY ;CHK ORDY = 1
6738	033250	000001			1	
6739	033252	004737	004312		JSR	PC,WRITAX ;LOAD ANOTHER CHAR INTO USYRT TX BUFFER
6740	033256	004737	004662		JSR	PC,OSIRDY ;CHK ORDY = 0
6741	033262	000000			0	
6742	033264	004737	006714		JSR	PC,IACTIV ;CHK IACT = 0
6743	033270	000000			0	
6744	033272	004737	007102		JSR	PC,RSEOM ;CHK RSOM = 0, REOM = 0
6745	033276	000000			0	
6746	033300	004737	005254		JSR	PC,STPLU ;CLOCK LU FOR 8 CYCLES
6747	033304	000010			8.	
6748	033306	004737	005352		JSR	PC,OACTIV ;CHK OACT = 1
6749	033312	000001			1	
6750	033314	004737	004662		JSR	PC,OSIRDY ;CHK ORDY = 1
6751	033320	000001			1	
6752	033322	005303			DEC	R3 ;DECR COUNTER
6753	033324	001332			BNE	4\$;BR IF NOT DONE YET
6754	033326	004737	006430		JSR	PC,ISIRDY ;CHK IRDY = 0
6755	033332	000000			0	
6756	033334	005737	002430		TST	CHPTYP ;SEE IF SIG USYRT
6757	033340	001007			BNE	11\$;BR IF NOT
6758	033342	105037	002374		CLRB	WAX15
6759	033346	112737	000002	002376	MOVB	#TEOM,WAX16 ;LOAD EOM CHAR
6760	033354	004737	004312		JSR	PC,WRITAX ;LOAD ANOTHER CHAR INTO USYRT TX BUFFER

```
6761 033360 004737 005254      11$: JSR    PC,STPLU      ;CLOCK LU FOR 3 CYCLES
6762 033364 000003                3
6763 033366 004737 006430      JSR    PC,ISIRDY     ;CHK IRDY = 1
6764 033372 000002                2
6765 033374 004737 005352      JSR    PC,OACTIV     ;CHK OACT = 1
6766 033400 000001                1
6767 033402 004737 006714      JSR    PC,IACTIV     ;CHK IACT = 1
6768 033406 000001                1
6769 033410 004737 007102      JSR    PC,RSEOM      ;CHK RSOM = 1, REOM = 0
6770 033414 000001                1
6771 033416 004737 006430      JSR    PC,ISIRDY     ;CHK IRDY = 0
6772 033422 000000                0
6773 033424 012737 000000 002402  MOV    #0,AXNUM      ;SET AX BYTE NO. FOR AX0
6774 033432 123727 002370 000000  CMPB   RAX15,#000    ;COMPARE RCV'D CHAR TO 000
6775 033440 001415                BEQ    9$            ;BR IF MATCH
6776 033442 012737 000000 002404  MOV    #0,GOODAT     ;SET EXPECTED DATA
6777 033450 013737 002370 002406  6$:  MOV    RAX15,BADDAT ;SET ACTUAL DATA
6778 033456 004737 004526      JSR    PC,GETALL     ;GET REGS FOR PRINTOUT
6779                                ;REPORT INCORRECT DATA CHAR RCV'D
6780                                ERRDF 26,EM26,ERR3
(4) 033462 104455
(5) 033464 000032
(5) 033466 013071
(5) 033470 015142
6781 033472 000511
6782 033474 004737 005254      9$:  JSR    PC,STPLU      ;CLOCK LU FOR 8 CYCLES
6783 033500 000010                8
6784 033502 004737 006430      JSR    PC,ISIRDY     ;CHK IRDY = 1
6785 033506 000002                2
6786 033510 004737 005352      JSR    PC,OACTIV     ;CHK OACT = 1
6787 033514 000001                1
6788 033516 004737 006714      JSR    PC,IACTIV     ;CHK IACT = 1
6789 033522 000001                1
6790 033524 004737 007102      JSR    PC,RSEOM      ;CHK RSOM = 0, REOM = 0
6791 033530 000000                0
6792 033532 004737 006430      JSR    PC,ISIRDY     ;CHK IRDY = 0
6793 033536 000000                0
6794 033540 123727 002370 000125  CMPB   RAX15,#125    ;COMPARE 2ND RCV'D CHAR TO 125
6795 033546 001404                BEQ    12$          ;BR IF MATCH
6796 033550 012737 000125 002404  MOV    #125,GOODAT   ;SET EXPECTED DATA
6797 033556 000734                BR     6$            ;BR TO REPORT ERROR
6798 033560 004737 005254      12$: JSR    PC,STPLU      ;CLOCK LU FOR 8 CYCLES
6799 033564 000010                8
6800 033566 004737 006430      JSR    PC,ISIRDY     ;CHK IRDY = 1
6801 033572 000002                2
6802 033574 004737 005352      JSR    PC,OACTIV     ;CHK OACT = 1
6803 033600 000001                1
6804 033602 004737 006714      JSR    PC,IACTIV     ;CHK IACT = 0
6805 033606 000000                0
6806 033610 004737 007102      JSR    PC,RSEOM      ;CHK RSOM = 0, REOM = 1
6807 033614 000002                2
6808 033616 004737 006430      JSR    PC,ISIRDY     ;CHK IRDY = 0
6809 033622 000000                0
6810 033624 123727 002370 000252  CMPB   RAX15,#252    ;COMPARE 3RD RCV'D CHAR TO 252
6811 033632 001404                BEQ    14$          ;BR IF MATCH
6812 033634 012737 000252 002404  MOV    #252,GOODAT   ;SET EXPECTED DATA
```

```
TRAP  C$ERDF
.WORD 26
.WORD EM26
.WORD ERR3
```

6813	033642	000702				BR	6\$:BR TO REPORT ERROR
6814	033644	012737	000014	002400	14\$:	MOV	#14,REGNUM	:SET REG NO. = 14
6815	033652	012737	000040	002366		MOV	#DISSI,WRIBYT	
6816	033660	004737	003750			JSR	PC,WRITLU	:CLEAR TX ENABLE
6817	033664	012737	000011	002400		MOV	#11,REGNUM	:SET REG NO. = 11
6818	033672	012737	000200	002366		MOV	#OC,WRIBYT	
6819	033700	004737	003750			JSR	PC,WRITLU	:SET OC TO SHUT DOWN TRANSMITTER
6820	033704	004737	005146			JSR	PC,WAIT50	:WAIT FOR SHUTDOWN
6821	033710	004737	005352			JSR	PC,OACTIV	:CHK FOR OACT = 0
6822	033714	000000				0		
6823	033716	005037	002426		18\$:	CLR	DISILO	:CLEAR DISABLE SILO FLAG
6824	033722	004737	003576			JSR	PC,MSTCLR	:ISSUE MASTER CLEAR TO CLEAN UP
6825	033726				ENDTST			
(3)	033726							
(3)	033726	104401						L10067: TRAP C\$ETST

6826
6827
6828
6829
6830
6831
6832
6833
6834
6835
6836
6837
6838
6839
6840
6841
6842
6843
6844
6845
6846
6847
6848

```
*****  
:SBTTL TEST 36 - RECEIVER BUFFER TEST - CHAR MODE, CRC  
:  
:* FIRST, A MASTER CLEAR IS DONE AND THE PROGRAM CHECKS FOR ICIR = 1 AND IRDY  
:* = 0. THEN, 2 SOM CHARS ARE LOADED AND CLOCKED INTO THE USYRT, AND 64  
:* BYTES OF A 256-BYTE BINARY COUNT DATA PATTERN (000-377) ARE LOADED INTO  
:* THE TX SILO.  
:* THE LINE UNIT IS THEN CLOCKED UNTIL IRDY = 1, AND THE PROGRAM CHECKS FOR  
:* THIS TO OCCUR WITHIN 40-43 CYCLES. THE PROGRAM READS THE RCV SILO, CHECKS THE  
:* CHAR FOR 000, AND CHECKS FOR IRDY = 0 AGAIN.  
:* THE LINE UNIT IS THEN CLOCKED IN GROUPS OF 8 CYCLES, AND AFTER EACH, THE  
:* PROGRAM CHECKS FOR ICIR = 1, IRDY = 1, UNTIL THE 64TH GROUP, AFTER WHICH  
:* IT CHECKS FOR ICIR = 0, IRDY = 1. THE SECOND DATA CHAR IS READ FROM THE  
:* RECEIVER SILO AND COMPARED TO 001. THEN, THE PROGRAM CHECKS FOR ICIR = 1,  
:* IRDY = 1 AGAIN.  
:* THE REST OF THE BINARY COUNT DATA BYTES ARE CYCLED 8 CLOCKS AND READ AND  
:* COMPARED A BYTE AT A TIME.  
:*****  
:BGNTST
```

6849	033730							
(3)	033730							T36::
6850	033730	012737	034204	002362		MOV	#A10,RETADR	:SET TEST EXIT ADDRESS FOR ERRORS
6851								-----
6852								: DO MASTER CLR, CHK FOR ICIR = 1, IRDY = 0
6853								-----
6854	033736	004737	003576			JSR	PC,MSTCLR	:ISSUE MASTER CLEAR
6855	033742	004737	006430			JSR	PC,ISIRDY	:CHK ICIR = 1, IRDY = 0
6856	033746	000001				1		
6857								-----
6858								: LOAD AND CLOCK 2 SOM'S, LOAD 64 BYTES OF BINARY COUNT PATTERN INTO TX SILO,
6859								: CLOCK LINE UNIT, CHK FOR IRDY = 1 WITHIN 40-43 CYCLES
6860								-----
6861	033750	004737	005540			JSR	PC,INITRN	:LOAD 2 SOM'S, CLOCK THEM INTO USYRT
6862	033754	000226				SYNCH		
6863	033756	000011				STRIP!DDCMP		
6864	033760	005003				CLR	R3	:INIT BINARY COUNT DATA FOR WRITING
6865	033762	010337	002422		2\$:	MOV	R3,TXWORD	

```

6866 033766 004737 005174      JSR    PC,LDTXSI      ;LOAD A DATA BYTE INTO TX SILO
6867 033772 005203              INC    R3             ;INCR DATA
6868 033774 020327 000100      CMP    R3,#64.       ;SEE IF 64 BYTES LOADED YET
6869 034000 002770              BLT    2$            ;BR IF NOT YET
6870 034002 004737 007434      JSR    PC,RCV1ST     ;RECEIVE AND TIME FIRST CHARACTER
6871 034006 000050
6872
6873      ;-----
6874      ; READ RCV SILO, COMPARE FIRST CHAR TO 000, CHK FOR ICIR = 1, IRDY = 0
6875 034010 005004
6876 034012 004737 007750      9$:   CLR    R4             ;INIT PATTERN FOR READING
6877 034016 000000              JSR    PC,CKDATA     ;READ RCV SILO, COMPARE DATA
6878 034020 000000              0             ;EXPECTED DATA = 000
6879 034022 005204              0             ;DON'T CLOCK LINE UNIT
6880 034024 004737 006430      16$:  INC    R4             ;INCR DATA FOR READING
6881 034030 000001              JSR    PC,ISIRDY     ;CHK FOR ICIR = 1, IRDY = 0
6882
6883      ;-----
6884      ; CLOCK 63 CHARS INTO RCV SILO, CHK ICIR = 1, IRDY = 1
6885 034032 004737 005254      18$:  JSR    PC,STPLU      ;CLOCK LU FOR 8 CYCLES
6886 034036 000010              8.
6887 034040 010337 002422      MOV    R3,TXWORD
6888 034044 004737 005174      JSR    PC,LDTXSI     ;LOAD ANOTHER WORD INTO TX SILO
6889 034050 005203              INC    R3             ;INCR PATTERN FOR WRITING
6890 034052 004737 006430      JSR    PC,ISIRDY     ;CHK ICIR = 1, IRDY = 1
6891 034056 000003              3
6892 034060 020327 000177      CMP    R3,#127.     ;SEE IF 63 MORE CHARS CLOCKED YET
6893 034064 002762              BLT    18$          ;BR IF NOT YET
6894
6895      ;-----
6896      ; CLOCK 1 MORE CHAR INTO RCV SILO, CHK ICIR = 0, IRDY = 1
6897 034066 004737 005254      JSR    PC,STPLU      ;CLOCK LU FOR 8 CYCLES
6898 034072 000010              8.
6899 034074 004737 006430      JSR    PC,ISIRDY     ;CHK ICIR = 0, IRDY = 1
6900 034100 000002              2
6901
6902      ;-----
6903      ; READ, COMPARE, CLOCK REST OF DATA CHARS
6904 034102 010437 034112      20$:  MOV    R4,21$      ;SET EXPECTED DATA
6905 034106 004737 007750      JSR    PC,CKDATA     ;READ AND COMPARE DATA
6906 034112 000000      21$:  0             ;EXPECTED SILO ENTRY GOES HERE
6907 034114 000000              0             ;DON'T CLOCK LINE UNIT
6908 034116 005204      22$:  INC    R4             ;INCR DATA PATTERN FOR READS
6909 034120 020427 000400      CMP    R4,#400      ;SEE IF ALL DONE READING YET
6910 034124 001427              BEQ    32$          ;BR IF DONE READING
6911 034126 004737 006430      JSR    PC,ISIRDY     ;CHK ICIR = 1, IRDY = 1
6912 034132 000003              3
6913 034134 004737 005254      JSR    PC,STPLU      ;CLOCK LU FOR 8 CYCLES
6914 034140 000010              8.
6915 034142 020327 000377      CMP    R3,#377      ;SEE IF ALL CHARS LOADED INTO TX SILO YET
6916 034146 003007              BGT    24$          ;BR IF YES
6917 034150 010337 002422      MOV    R3,TXWORD
6918 034154 004737 005174      JSR    PC,LDTXSI     ;LOAD ANOTHER CHAR INTO TX SILO
6919 034160 005203              INC    R3             ;INCR DATA PATTERN FOR WRITING
6920 034162 000137 034102      JMP    20$
6921 034166 012737 001000 002422 24$:  MOV    #TXEOM,TXWORD
  
```


6922 034174 004737 005174
6923 034200 000137 034102
6924 034204
6925 034204 004737 003576
6926 034210
(3) 034210
(3) 034210 104401

JSR PC,LDTXSI ;LOAD EOM INTO TX SILO
JMP 20\$
32\$: JSR PC,MSTCLR ;ISSUE MASTER CLEAR TO CLEAN UP
A10: ENDIST

L10070: TRAP C\$ETST

6927
6928
6929
6930
6931
6932
6933
6934
6935
6936
6937
6938
6939
6940
6941
6942
6943

:SBITL TEST 37 - RECEIVER CHAR LENGTH TIMING TEST - CHAR MODE, NO CRC
:*****
:* THE LINE UNIT IS PLACED IN CHAR MODE, WITH NO ERROR DETECTION, AND A MSG IS
:* INITIATED WITH 2 SYNCH CHARS. NEXT, FIFTEEN 000 CHARS ARE LOADED INTO THE
:* TRANSMITTER SILO. THE LINE UNIT IS THEN CLOCKED USING STEPLU WITH LULOOP
:* SET, WHILE THE RECEIVER CHAR LENGTH IS SET TO THE FOLLOWING VALUES : 5,6,7,8.
:* FOR EACH RCV CHAR LENGTH, THE PROGRAM CHECKS TO MAKE SURE THET USYRT RECEIVER
:* FLAGS OCCUR THE PROPER NO. OF CYCLES APART, FOR EACH RCV CHAR LENGTH.
:* (FOR EXAMPLE A 5-BIT CHAR TAKES 5 CLOCK CYCLES TO BE RECEIVED). A MASTER
:* CLEAR IS THEN DONE TO TERMINATE THE OPERATION.
:*****

6944 034212
(3) 034212
6945 034212 012737 034404 002362
6946 034220 004737 005540
6947 034224 000000
6948 034226 000341
6949 034230 012701 000017
6950 034234 005037 002422
6951 034240 004737 005174
6952 034244 005301
6953 034246 001374
6954 034250 004737 005146
6955 034254 012737 000006 002402
6956 034262 012737 000000 002374
6957 034270 012737 000005 002376
6958 034276 004737 004312
6959 034302 004737 005254
6960 034306 100012
6961 034310 012737 000006 002376
6962 034316 004737 004312
6963 034322 004737 007434
6964 034326 000005
6965 034330 012737 000007 002376
6966 034336 004737 004312
6967 034342 004737 011074
6968 034346 000006
6969 034350 012737 000000 002376
6970 034356 004737 004312
6971 034362 004737 011074
6972 034366 000007
6973 034370 004737 011074
6974 034374 000010

BGNTST
T37::
MOV #24\$,RETADR ;SET TEST EXIT ADRS FOR ERRORS
JSR PC,INITRN ;DO MASTER CLR, LOAD 2 SOM'S
000
CRC2!CRC1!IDLE!DDCMP
MOV #15.,R1 ;INIT COUNTER
CLR TXWORD
3\$: JSR PC,LDTXSI ;LOAD A 000 CHAR INTO TX SILO
DEC R1 ;DECR COUNTER
BNE 3\$;BR IF NOT DONE LOADING YET
JSR PC,WAIT50 ;WAIT FOR SILO TO RIPPLE
MOV #6,AXNUM ;SET BYTE NO. = 6 FOR AX3
MOV #000,WAX15 ;SET DATA FOR AX3-15 = 0
MOV #RXLEN2!RXLENO,WAX16 ;SET RCV LEN = 5
JSR PC,WRITAX ;LOAD AX3
JSR PC,STPLU ;CLK LU UNTIL TX'ING 1ST DATA CHAR
CHPCHK!10.
MOV #RXLEN2!RXLEN1,WAX16 ;SET RCV LEN = 6
JSR PC,WRITAX ;LOAD AX3
JSR PC,RCV1ST ;CLOCK 5-BIT DATA CHAR
5
MOV #RXLEN2!RXLEN1!RXLENO,WAX16 ;SET RCV LEN = 7
JSR PC,WRITAX ;LOAD AX3
JSR PC,RXCHAR ;RCV 5-BIT DATA CHAR, CLK 6-BIT
6
MOV #0,WAX16 ;SET RCV LEN = 8
JSR PC,WRITAX ;LOAD AX3
JSR PC,RXCHAR ;RCV 6-BIT DATA CHAR, CLK 7-BIT
7
JSR PC,RXCHAR ;RCV 7-BIT DATA CHAR, CLK 8-BIT
8.

6975 034376 004737 011074
6976 034402 000010
6977 034404 004737 003576
6978 034410
(3) 034410
(3) 034410 104401

JSR PC,RXCHAR ;RCV 8-BIT DATA CHAR
8.
24\$: JSR PC,MSTCLR ;ISSUE MASTER CLEAR TO CLEAN UP
ENDTST

L10071: TRAP C\$ETST

6979
6980
6981
6982
6983
6984
6985
6986
6987
6988
6989
6990
6991
6992
6993
6994
6995

```
*****  
:SBTTL TEST 38 - RECEIVER CHAR LENGTH TIMING TEST - BIT MODE, NO CRC  
:  
:* THE LINE UNIT IS PLACED IN BIT MODE WITH NO ERROR DETECTION, AND A MESSAGE IS  
:* INITIATED WITH 2 FLAG CHARS. NEXT, FIFTEEN 000 CHARS ARE LOADED INTO THE  
:* TRANSMITTER SILO. THE LINE UNIT IS THEN CLOCKED USING STEPLU WITH LULOOP  
:* SET, WHILE THE RCV CHAR LENGTH IS SET TO THE FOLLOWING VALUES : 8,8,8,7,6,5,  
:* 4,3,2,1. FOR EACH RCV CHAR LENGTH, THE PROGRAM CHECKS TO MAKE SURE THAT THE  
:* USYRT RECEIVER FLAGS OCCUR THE PROPER NO. OF CYCLES APART, FOR EACH RCV  
:* CHAR LENGTH. (FOR EXAMPLE, A 5 BIT CHAR TAKES 5 CLOCK CYCLES TO BE RECEIVED).  
:* A MASTER CLEAR IS THEN DONE TO TERMINATE THE OPERATION.  
:*****  
:BGNTST
```

6996 034412
(3) 034412
6997 034412 012737 034704 002362
6998 034420 004737 005540
6999 034424 000000
7000 034426 000300
7001 034430 012701 000017
7002 034434 005037 002422
7003 034440 004737 005174
7004 034444 005301
7005 034446 001374
7006 034450 004737 005146
7007 034454 012737 000006 002402
7008 034462 012737 000000 002374
7009 034470 004737 007434
7010 034474 000040
7011 034476 012737 000000 002376
7012 034504 004737 004312
7013 034510 004737 011074
7014 034514 000010
7015 034516 012737 000007 002376
7016 034524 004737 004312
7017 034530 004737 011074
7018 034534 000010
7019 034536 012737 000006 002376
7020 034544 004737 004312
7021 034550 004737 011074
7022 034554 000007
7023 034556 012737 000005 002376
7024 034564 004737 004312
7025 034570 004737 011074
7026 034574 000006
7027 034576 012737 000004 002376

```
T38::  
MOV #24$,RETADR ;SET TEST EXIT ADRS FOR ERRORS  
JSR PC,INITRN ;DO MASTER CLR, LOAD 2 SOM'S  
000  
CRC2!CRC1  
MOV #15.,R1 ;INIT COUNTER  
CLR TXWORD  
3$: JSR PC,LDTXSI ;LOAD A 000 CHAR INTO TX SILO  
DEC R1 ;DECR COUNTER  
BNE 3$ ;BR IF NOT DONE LOADING YET  
JSR PC,WAIT50 ;WAIT FOR SILO TO RIPPLE  
MOV #6,AXNUM ;SET BYTE NO. = 6 FOR AX3  
MOV #000,WAX15 ;SET DATA FOR AX3-15 = 0  
JSR PC,RCV1ST ;CLOCK FIRST 8-BIT DATA CHAR  
32.  
MOV #0,WAX16 ;SET RCV LEN = 8  
JSR PC,WRITAX ;LOAD AX3  
JSR PC,RXCHAR ;RCV FIRSI 8-BIT DATA CHAR, CLK SECOND 8-BIT  
8.  
MOV #RXLEN2!RXLEN1!RXLENO,WAX16 ;SET RCV LEN = 7  
JSR PC,WRITAX ;LOAD AX3  
JSR PC,RXCHAR ;RCV SECOND 8-BIT DATA CHAR, CLK 3RD 8-BIT  
8.  
MOV #RXLEN2!RXLEN1,WAX16 ;SET RCV !EN = 6  
JSR PC,WRITAX ;LOAD AX3  
JSR PC,RXCHAR ;RCV 3RD 8-BIT DATA CHAR, CLK 7-BIT  
7  
MOV #RXLEN2!RXLENO,WAX16 ;SET RCV LEN = 5  
JSR PC,WRITAX ;LOAD AX3  
JSR PC,RXCHAR ;RCV 7-BIT DATA CHAR, CLK 6-BIT  
6  
MOV #RXLEN2,WAX16 ;SET RCV LEN = 4
```

7028	034604	004737	004312		JSR	PC,WRITAX	;LOAD AX3
7029	034610	004737	011074		JSR	PC,RXCHAR	;RCV 6-BIT DATA CHAR, CLK 5-BIT
7030	034614	000005			5		
7031	034616	012737	000003	002376	MOV	#RXLEN1!RYLENO,WAX16	;SET RCV LEN = 3
7032	034624	004737	004312		JSR	PC,WRITAX	;LOAD AX3
7033	034630	004737	011074		JSR	PC,RXCHAR	;RCV 5-BIT DATA CHAR, CLR 4-BIT
7034	034634	000004			4		
7035	034636	012737	000002	002376	MOV	#RXLEN1,WAX16	;SET RCV LEN = 2
7036	034644	004737	004312		JSR	PC,WRITAX	;LOAD AX3
7037	034650	004737	011074		JSR	PC,RXCHAR	;RCV 4-BIT DATA CHAR, CLK 3-BIT
7038	034654	000003			3		
7039	034656	012737	000001	002376	MOV	#RXLENO,WAX16	;SET RCV LEN = 1
7040	034664	004737	004312		JSR	PC,WRITAX	;LOAD AX3
7041	034670	004737	011074		JSR	PC,RXCHAR	;RCV 3-BIT DATA CHAR, CLK 2-BIT
7042	034674	000002			2		
7043	034676	004737	011074		JSR	PC,RXCHAR	;RCV 2-BIT DATA CHAR, CLK 1-BIT
7044	034702	000001			1		
7045	034704	004737	003576		JSR	PC,MSTCLR	;ISSUE MASTER CLEAR TO CLEAN UP
7046	034710			24\$:			
(3)	034710			ENDTST			
(3)	034710	104401					L10072: TRAP C\$ETST

7047
7048
7049
7050
7051

```
*****  
:SBTTL TEST 39 - TRANSMITTER UNDERRUN ERROR, IDLE MARKING, CHAR MODE,NO CRC  
:*  
:* THE LINE UNIT IS PLACED IN CHAR MODE, AND THE IDLE BIT IS SET. THEN, A  
:* MSG IS INITIATED, A 000 CHAR IS SENT, AND THE TX BUFFER IS NOT SERVICED  
:* IN RESPONSE TO THE USYRT TX BUFFER EMPTY FLAG, WHICH CAUSES A TX UNDERRUN  
:* ERROR. THEN, THE RECEIVER IS CLOCKED AND CHECKED FOR TWO 377 CHARS TO BE  
:* RECEIVED (LINE MARKING) BEFORE SHUTTING DOWN WITH A MASTER CLEAR.  
:*****  
BGNTST
```

7061	034712						
(3)	034712						
7062	034712	012737	034774	002362	MOV	#24\$,RETADR	;SET TEST EXIT ADDRESS FOR ERRORS
7063	034720	004737	005540		JSR	PC,INITRN	;MST CLR, LOAD 2 SOM'S
7064	034724	000226			SYNCH		
7065	034726	000341			CRC2!CRC1!IDLE!DDCMP		
7066	034730	012737	000000	002422	MOV	#000,TXWORD	
7067	034736	004737	005174		JSR	PC,LDTXSI	;LOAD 000 CHAR INTO TX SILO
7068	034742	004737	005254		JSR	PC,STPLU	;CLOCK LINE UNIT UNTIL LINE GOES MARKING
7069	034746	000063			51.		
7070	034750	004737	007354		JSR	PC,RDRXSI	;READ 000 CHAR
7071	034754	004737	007750		JSR	PC,CKDATA	;READ AND CHECK FOR MARK CHAR (377)
7072	034760	000377			377		
7073	034762	000000			000		
7074	034764	004737	007750		JSR	PC,CKDATA	;READ AND CHECK FOR ANOTHER MARK CHAR
7075	034770	000377			377		
7076	034772	000000			000		
7077	034774	004737	003576		JSR	PC,MSTCLR	;ISSUE MASTER CLEAR TO CLEAN UP
7078	035000			24\$:			
(3)	035000			ENDTST			
(3)	035000	104401					L10073: TRAP C\$ETST

7079
7080
7081
7082
7083
7084
7085
7086
7087
7088
7089
7090
7091
7092
7093
7094
7095
7096
7097
7098

```
*****  
:SBTTL TEST 40 - MSG TERMINATION WITH GA CHARS - BIT MODE,NO CRC  
:  
:* THE DEVICE IS ENABLED FOR TRANSMIT AND RECEIVE, AND A MESSAGE IS  
:* INITIATED IN BIT MODE.  
:* 2 FLAG CHARACTERS ARE SENT, FOLLOWED BY  
:* THE FOLLOWING DATA CHARACTERS : 000, 125, 252, 377, 000. THEN THE LOOP  
:* MODE BIT (STRIP) IS SET AND 2 TERMINATING GO-AHEAD CHARACTERS ARE  
:* SENT. EACH USYRT RCV FLAG IS TIMED AS IT IS RECEIVED, AND THE 5 DATA  
:* WORDS ARE READ AND COMPARED TO EXPECTED VALUES.  
:* ALSO, THE FIRST GA CHAR IS CHECKED BY SCANNING THE TXDATA BIT AS THE GA  
:* IS BEING TRANSMITTED (GA = 376 OCTAL).  
:* THE TEST ALSO CHECKS FOR SETTING OF RAB AND EBLK  
:* IN LOOP MODE.  
:*****  
:BGNTST
```

7099 035002
(3) 035002
7100 035002 012737 035136 002362
7101 035010 004737 005540
7102 035014 000000
7103 035016 000310
7104 035020 004737 011016
7105 035024 003224
7106 035026 000005
7107 035030 012737 005000 002422
7108 035036 004737 005174
7109 035042 004737 005174
7110 035046 004737 005146
7111 035052 004737 005254
7112 035056 100071
7113 035060 004737 011176
7114 035064 000376
7115 035066 004737 007750
7116 035072 000000
7117 035074 000000
7118 035076 004737 007750
7119 035102 000125
7120 035104 000000
7121 035106 004737 007750
7122 035112 000252
7123 035114 000000
7124 035116 004737 007750
7125 035122 000377
7126 035124 000010
7127 035126 004737 007750
7128 035132 003000
7129 035134 000010
7130 035136 004737 003576
7131
7132 035142
(3) 035142

```
MOV #24$,RETADR ;SET TEST EXIT ADRS FOR ERRORS T40::  
JSR PC,INITRN ;MST CLR, LOAD 2 SOM'S  
000  
CRC2!CRC1!STRIP  
JSR PC,LODMSG ;LOAD DATA CHARS INTO TX SILO  
MSG1+4  
5  
MOV #TXGOA!TXEOM,TXWORD  
JSR PC,LDTXSI ;LOAD A GA CHAR INTO TX SILO  
JSR PC,LDTXSI ;LOAD ANOTHER GA  
JSR PC,WAIT50 ;ALLOW SILO TO RIPPLE  
JSR PC,STPLU ;CLOCK LU UNTIL GA CHAR IS TX'ING  
CHPCHK!57.  
JSR PC,CKTBIT ;SCAN TXDATA BIT FOR GA CHAR  
376  
JSR PC,CKDATA ;RCV 000 CHAR, CLK 125  
000  
0  
JSR PC,CKDATA ;RCV 125 CHAR, CLK 252  
125  
0  
JSR PC,CKDATA ;RCV 252 CHAR, CLK 377  
252  
0  
JSR PC,CKDATA ;RCV 377 CHAR  
377  
8.  
JSR PC,CKDATA ;RCV 000 CHAR, CHK RAB = 1, EBLK = 1  
3000  
8.  
24$: JSR PC,MSTCLR ;ISSUE MASTER CLEAR
```

ENDTST

L10074:

(3) 035142 104401

TRAP CSETST

7133
7134
7135
7136
7137
7138
7139
7140
7141
7142
7143
7144
7145
7146
7147
7148
7149
7150

```
*****  
:SBTTL TEST 41 - IDLE SYNCHS TEST - CHAR MODE  
:*  
:* THE DEVICE IS ENABLED FOR TRANSMIT AND RECEIVE, AND A MESSAGE IS  
:* INITIATED IN CHAR MODE. 24(DEC) SYNCHS ARE SENT.  
:* EACH SYNCH IS TIMED AS IT IS RECEIVED, AND THE BITS ARE CHECKED  
:* FOR A VALID SYNCH CHAR FOR EACH OF THE 22 SYNCHS WHICH FOLLOW  
:* THE FIRST TWO (THESE PERFORM SYNCHRONIZATION, AND ARE NOT READ).  
:* WHILE THE LAST SYNCH IS BEING TRANSMITTED, OC IS SET, AND THE  
:* NEXT CHAR RCV'D AFTER THE SYNCH IS CHECKED TO BE 377 (LINE MARKING).  
:* THEN, A MASTER CLEAR IS ISSUED.  
:* THE SYNCH CHAR USED IS 226 (OCTAL).  
:*****  
BGNTST
```

7151 035144

(3) 035144

7152 035144 012737 035304 002362

7153 035152 004737 005540

7154 035156 000226

7155 035160 000341

7156 035162 012701 000026

7157 035166 012737 000226 002422

7158 035174 004737 005174

7159 035200 005301

7160 035202 001374

7161 035204 004737 005146

7162 035210 004737 007434

7163 035214 000030

7164 035216 012701 000025

7165 035222 004737 007750

7166 035226 000226

7167 035230 000010

7168 035232 005301

7169 035234 001372

7170 035236 004737 007750

7171 035242 000226

7172 035244 000004

7173 035246 012737 000011 002400

7174 035254 112737 000200 002366

7175 035262 004737 003750

7176 035266 004737 005254

7177 035272 000014

7178 035274 004737 007750

7179 035300 000377

7180 035302 000000

7181 035304 004737 003576

(3) 035310

(3) 035310 104401

7183

7184

```
T41::  
:SET TEST EXIT ADRS FOR ERRORS  
:MST CLR, LOAD 2 SOM'S  
:INIT COUNTER  
:LOAD AN SOM INTO TX SILO  
:DECR COUNTER  
:BR IF MORE TO LOAD  
:ALLOW SILC TO RIPPLE  
:CLK LU UNTIL 3RD SYNCH RCV'D (RCVR IS ACTIVE)  
:INIT COUNTER  
:READ A SYNCH, CLK NEXT ONE  
:DECR COUNTER  
:BR IF NOT ALL CHECKED  
:CHECK LAST SYNCH  
:SET REG NO. = 11  
:SET OC IN REG 11  
:FINISH CLOCKING CHAR, AND THEN SOME  
:RCV A MARK CHAR, CHK IT  
:ISSUE MASTER CLEAR TO CLEAN UP
```

24\$:
ENDTST

L10075:

TRAP CSETST

CZDMRC M8203 STATIC DIAG #1
CZDMRC.P11 13-MAR-80 14:38

MACY11 30A(1052) 13-MAR-80 14:40 L 13 PAGE 6-19
TEST 41 - IDLE SYNCHS TEST - CHAR MODE

SEQ 0167

7185
7186
7187
7188
7189
7190
7191

:SBITL TEST 42 - STRIP SYNCH TEST
:*
:* THE DEVICE IS ENABLED FOR TRANSMIT AND RECEIVE, AND A MESSAGE IS

7193
 7194
 7195
 7196
 7197
 7198
 7199
 7200
 7201
 7202
 7203
 7204
 7205
 7206
 (3)
 7207
 7208
 7209
 7210
 7211
 7212
 7213
 7214
 7215
 7216
 7217
 7218
 7219
 7220
 7221
 7222
 7223
 7224
 7225
 7226
 7227
 7228
 7229
 7230
 7231
 7232
 7233
 7234
 7235
 7236
 7237
 7238
 7239
 7240
 7241
 7242
 7243
 7244
 7245
 7246
 7247

035312
 035312
 035312 012737 035532 002362
 035320 012701 035540
 035324 011137 035334
 035330 004737 005540
 035334 000000
 035336 000311
 035340 012737 000400 002422
 035346 012702 000026
 035352 004737 005174
 035356 005302
 035360 001374
 035362 004737 011016
 035366 003246
 035370 000006
 035372 012737 001000 002422
 035400 004737 005174
 035404 004737 005174
 035410 004737 005146
 035414 011137 035436
 035420 012702 000027
 035424 004737 005254
 035430 100010
 035432 004737 011176
 035436 000000
 035440 005302
 035442 001373
 035444 004737 007434
 035450 000010
 035452 004737 007750
 035456 000377
 035460 000010
 035462 004737 007750
 035466 000000
 035470 000010
 035472 004737 007750
 035476 000125
 035500 000010
 035502 004737 007750
 035506 000252
 035510 000040
 035512 004737 005352

:* INITIATED IN CHAR MODE AND WITH THE STRIP SYNCH
 :* BIT SET. THEN 24 (DEC) SYNCHS ARE SENT
 :* FOLLOWED BY THE FOLLOWING DATA CHARACTERS : 377, 000, 125, 252,
 :* AND 2 TERMINATING SYNCHS.
 :* EACH OF THE 23 SYNCHS AFTER THE FIRST ARE CHECKED AT THE TRANSMITTER OUTPUT,
 :* BY SCANNING THE TXDATA BIT.
 :* EACH USYRT RCV FLAG IS TIMED AS IT IS RECEIVED, AND THE 4 DATA WORDS
 :* ARE READ AND COMPARED TO EXPECTED VALUES.
 :* FINALLY, THE LINE UNIT IS CLOCKED FOR SEVERAL CHAR TIMES, AND A CHECK
 :* IS MADE FOR OACT = 0 (TEOM SHOULD CAUSE TX ENABLE TO DROP).
 :* THE ABOVE TEST IS REPEATED FOR EACH OF THE FOLLOWING SYNCH CHAR DATA
 :* PATTERNS : 226,000,125,252,376,177.

 BGNTST

T42::
 MOV #24\$,RETADR ;SET TEST EXIT ADRS FOR ERRORS
 MOV #SYNPAT,R1 ;GET POINTER TO DATA
 2\$: MOV (R1),3\$;GET A SYNCH PATTERN
 JSR PC,INITRN ;MST CLR, LOAD 2 SOM'S
 3\$: .WORD 0 ;SYNCH PATTERN GOES HERE
 CRC2!CRC1!STRIP!DDCMP
 MOV #TXSOM,TXWORD
 6\$: MOV #22.,R2 ;LOAD 22 SOM'S INTO TX SILO
 JSR PC,LDTXSI
 DEC R2
 BNE 6\$
 JSR PC,LODMSG ;LOAD DATA CHARS INTO TX SILO
 MSG4
 6
 MOV #TXEOM,TXWORD
 JSR PC,LDTXSI ;LOAD A TEOM
 JSR PC,LDTXSI ;LOAD ANOTHER TEOM
 JSR PC,WAIT50 ;ALLOW SILO TO RIPPLE
 MOV (R1),16\$;GET CURRENT SYNCH PATTERN
 MOV #23.,R2 ;INIT COUNTER
 JSR PC,STPLU ;CLOCK OUT FIRST SYNCH
 CHPCHK!8.
 14\$: JSR PC,CKTBIT ;CHECK TX'D SYNCH
 16\$: .WORD 0 ;SYNCH PATTERN GOES HERE
 DEC R2 ;DECR COUNTER
 BNE 14\$;BR IF NOT DONE CHECKING LAST 23 SYNCHS
 JSR PC,RCV1ST ;CLOCK UNTIL 000 CHAR RCV'D
 8.
 JSR PC,CKDATA ;RCV 377, CLOCK 000
 377
 8.
 JSR PC,CKDATA ;RCV 000, CLK 125
 000
 8.
 JSR PC,CKDATA ;RCV 125, CLK 252
 125
 8.
 JSR PC,CKDATA ;RCV 252, CLK END OF MSG
 252
 32.
 JSR PC,OACTIV ;CHK FOR OACT = 0

7248	035516	000000		0				
7249	035520	062701	000002	ADD	#2,R1	:	INIT SYNCH PATTERN POINTER	
7250	035524	020127	035554	CMP	R1,#SYNPAT+12.	:	SEE IF ALL PATTERNS CHECKED YET	
7251	035530	103675		BLO	2\$:	BR IF NOT YET	
7252	035532	004737	003576	JSR	PC,MSTCLR	:	ISSUE MASTER CLEAR TO CLEAN UP	
7253	035536			ENDTST				
(3)	035536							
(3)	035536	104401					L10076:	TRAP C\$ETST
7254								
7255	035540	000226		SYNPAT:	226			
7256	035542	000000			000			
7257	035544	000125			125			
7258	035546	000252			252			
7259	035550	000376			376			
7260	035552	000177			177			
7261								
7262								
7263								
7264								
7265								

.SBTTL HARDWARE PARAMETER CODING SECTION

```
:/ THE HARDWARE PARAMETER CODING SECTION CONTAINS MACROS  
:/ THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES. THE  
:/ MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE  
:/ INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES. THE  
:/ MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS  
:/ WITH THE OPERATOR.  
:////
```

```
7267  
7268  
7269  
7270  
7271  
7272  
7273  
7274  
7275  
7276  
7277  
7278  
7279  
7280 035554          BGNHRD  
      (3) 035554 000022  
      (3) 035556  
7281  
7282 035556          GPRMA  ADDRES,2,0,160000,177776,YES  
      (4) 035556 001031  
      (4) 035560 035622  
      (4) 035562 160000  
      (4) 035564 177776  
7283 035566          GPRMA  VECTOR,4,0,0,674,YES  
      (4) 035566 002031  
      (4) 035570 035650  
      (4) 035572 000000  
      (4) 035574 000674  
7284 035576          GPRMD  PRIRTY,6,0,7000,4,7,YES  
      (4) 035576 003032  
      (4) 035600 035701  
      (4) 035602 007000  
      (4) 035604 000004  
      (4) 035606 000007  
7285 035610          GPRMD  ISRUN,24,0,7,0,7,YES  
      (4) 035610 012032  
      (4) 035612 035732  
      (4) 035614 000007  
      (4) 035616 000000  
      (4) 035620 000007  
7286  
7287 035622          ENDHRD  
      (2)  
      (3) 035622  
7288  
7289 035622 042504 044526 042503  ADDRES: .ASCIZ /DEVICE CSR ADDRESS : /  
      035630 041440 051123 040440  
      035636 042104 042522 051523  
      035644 035040 000040  
7290 035650 042504 044526 042503  VECTOR: .ASCIZ /DEVICE VECTOR ADDRESS : /  
      035656 053040 041505 047524  
      035664 020122 042101 051104  
      035672 051505 020123 020072  
      035700 000  
7291 035701 104 053105 041511  PRIPTY: .ASCIZ /DEVICE PRIORITY LEVEL : /  
      035706 020105 051120 047511  
      035714 044522 054524 046040
```

```
.WORD L10077-L$HARD/2  
L$HARD::  
  
.WORD T$CODE  
.WORD ADDRES  
.WORD T$LLOLIM  
.WORD T$SHILIM  
  
.WORD T$CODE  
.WORD VECTOR  
.WORD T$LLOLIM  
.WORD T$SHILIM  
  
.WORD T$CODE  
.WORD PRIRTY  
.WORD 7000  
.WORD T$LLOLIM  
.WORD T$SHILIM  
  
.WORD T$CODE  
.WORD ISRUN  
.WORD 7  
.WORD T$LLOLIM  
.WORD T$SHILIM  
  
L10077: .EVEN
```

	035722	053105	046105	035040
	035730	000040		
7292	035732	044515	051103	050117
	035740	047522	042503	051523
	035746	051117	051040	047125
	035754	051440	044527	041524
	035762	020110	020055	054524
	035770	042520	030040	044440
	035776	020106	043117	026106
	036004	030440	044440	020106
	036012	047117	035040	000040

ISRUN: .ASCIZ /MICROPROCESSOR RUN SWITCH - TYPE 0 IF OFF, 1 IF ON : /

7293
7294
7295
7296
7297
7298
7299

.EVEN

```
7301 .SBTTL SOFTWARE PARAMETER CODING SECTION
7302
7303
7304 :////////////////////////////////////////////////////////////////////
7305 :/ THE SOFTWARE PARAMETER CODING SECTION CONTAINS MACROS
7306 :/ THAT ARE USED BY THE SUPERVISOR TO BUILD P-TABLES. THE
7307 :/ MACROS ARE NOT EXECUTED AS MACHINE INSTRUCTIONS BUT ARE
7308 :/ INTERPRETED BY THE SUPERVISOR AS DATA STRUCTURES. THE
7309 :/ MACROS ALLOW THE SUPERVISOR TO ESTABLISH COMMUNICATIONS
7310 :/ WITH THE OPERATOR.
7311 :////////////////////////////////////////////////////////////////////
7312
7313 036020 BGNSFT
7314 (3) 036020 000000
7315 (3) 036022 L$SOFT: .WORD L10100-L$SOFT/2
7316
7317
7318
7319
7320
7321
7322
7323
7324
7325
7326
7327
7328 036022
7329 036222 036222
7330 036222 000240
7331 036224 000240
7332 036226 000240
7333
7334 :***** PATCH AREA FOR DEBUG *****
7335 PATCH:
7336 . = +200
7337 NOP
7338 NOP
7339 NOP
7340 :*****
7341
7342
7343
7344
7345
7346
7347
7348
7349
7350
7351
7352
7353
7354
7355
7356
7357
7358
7359
7360
7361
7362
7363
7364
7365
7366
7367
7368
7369
7370
7371
7372
7373
7374
7375
7376
7377
7378
7379
7380
7381
7382
7383
7384
7385
7386
7387
7388
7389
7390
7391
7392
7393
7394
7395
7396
7397
7398
7399
7400
7401
7402
7403
7404
7405
7406
7407
7408
7409
7410
7411
7412
7413
7414
7415
7416
7417
7418
7419
7420
7421
7422
7423
7424
7425
7426
7427
7428
7429
7430
7431
7432
7433
7434
7435
7436
7437
7438
7439
7440
7441
7442
7443
7444
7445
7446
7447
7448
7449
7450
7451
7452
7453
7454
7455
7456
7457
7458
7459
7460
7461
7462
7463
7464
7465
7466
7467
7468
7469
7470
7471
7472
7473
7474
7475
7476
7477
7478
7479
7480
7481
7482
7483
7484
7485
7486
7487
7488
7489
7490
7491
7492
7493
7494
7495
7496
7497
7498
7499
7500
7501
7502
7503
7504
7505
7506
7507
7508
7509
7510
7511
7512
7513
7514
7515
7516
7517
7518
7519
7520
7521
7522
7523
7524
7525
7526
7527
7528
7529
7530
7531
7532
7533
7534
7535
7536
7537
7538
7539
7540
7541
7542
7543
7544
7545
7546
7547
7548
7549
7550
7551
7552
7553
7554
7555
7556
7557
7558
7559
7560
7561
7562
7563
7564
7565
7566
7567
7568
7569
7570
7571
7572
7573
7574
7575
7576
7577
7578
7579
7580
7581
7582
7583
7584
7585
7586
7587
7588
7589
7590
7591
7592
7593
7594
7595
7596
7597
7598
7599
7600
7601
7602
7603
7604
7605
7606
7607
7608
7609
7610
7611
7612
7613
7614
7615
7616
7617
7618
7619
7620
7621
7622
7623
7624
7625
7626
7627
7628
7629
7630
7631
7632
7633
7634
7635
7636
7637
7638
7639
7640
7641
7642
7643
7644
7645
7646
7647
7648
7649
7650
7651
7652
7653
7654
7655
7656
7657
7658
7659
7660
7661
7662
7663
7664
7665
7666
7667
7668
7669
7670
7671
7672
7673
7674
7675
7676
7677
7678
7679
7680
7681
7682
7683
7684
7685
7686
7687
7688
7689
7690
7691
7692
7693
7694
7695
7696
7697
7698
7699
7700
7701
7702
7703
7704
7705
7706
7707
7708
7709
7710
7711
7712
7713
7714
7715
7716
7717
7718
7719
7720
7721
7722
7723
7724
7725
7726
7727
7728
7729
7730
7731
7732
7733
7734
7735
7736
7737
7738
7739
7740
7741
7742
7743
7744
7745
7746
7747
7748
7749
7750
7751
7752
7753
7754
7755
7756
7757
7758
7759
7760
7761
7762
7763
7764
7765
7766
7767
7768
7769
7770
7771
7772
7773
7774
7775
7776
7777
7778
7779
7780
7781
7782
7783
7784
7785
7786
7787
7788
7789
7790
7791
7792
7793
7794
7795
7796
7797
7798
7799
7800
7801
7802
7803
7804
7805
7806
7807
7808
7809
7810
7811
7812
7813
7814
7815
7816
7817
7818
7819
7820
7821
7822
7823
7824
7825
7826
7827
7828
7829
7830
7831
7832
7833
7834
7835
7836
7837
7838
7839
7840
7841
7842
7843
7844
7845
7846
7847
7848
7849
7850
7851
7852
7853
7854
7855
7856
7857
7858
7859
7860
7861
7862
7863
7864
7865
7866
7867
7868
7869
7870
7871
7872
7873
7874
7875
7876
7877
7878
7879
7880
7881
7882
7883
7884
7885
7886
7887
7888
7889
7890
7891
7892
7893
7894
7895
7896
7897
7898
7899
7900
7901
7902
7903
7904
7905
7906
7907
7908
7909
7910
7911
7912
7913
7914
7915
7916
7917
7918
7919
7920
7921
7922
7923
7924
7925
7926
7927
7928
7929
7930
7931
7932
7933
7934
7935
7936
7937
7938
7939
7940
7941
7942
7943
7944
7945
7946
7947
7948
7949
7950
7951
7952
7953
7954
7955
7956
7957
7958
7959
7960
7961
7962
7963
7964
7965
7966
7967
7968
7969
7970
7971
7972
7973
7974
7975
7976
7977
7978
7979
7980
7981
7982
7983
7984
7985
7986
7987
7988
7989
7990
7991
7992
7993
7994
7995
7996
7997
7998
7999
8000
```

ABORT = 000004	2395#													
ADDRS = 035622	4441	4447	4462	4482	4500	4521	4539	4558	4577	7282	7289#			
ADR = 000020 G	2354#													
ANBITS = 002561	2794#	5473	5474	5524	5525	5571	5572							
APA = 000200	2573#													
ASBCO = 000020	2530#													
ASBC1 = 000040	2529#													
ASBC2 = 000100	2528#													
ASSEMB = 000010	2211													
AXNUM 002402	2715#	3367	3405	3418	3443	3445	3451*	3457*	3458	3460*	3462	3653	3661*	
	3704*	3900	3906*	3935*	3941*	4183	4185*	4194*	4203*	5330*	5342*	5346*	5356	
	5368*	5379*	5407*	5414*	5416	5428*	5470*	5485*	5521*	5536*	5568*	5583*	5628*	
	5646*	5683*	5763*	5774*	5879*	6186*	6253*	6363*	6443*	6526*	6589*	6668*	6717*	
	6773*	6955*	7007*											
AX0.15 = 002322	2643#	3450	4471	4490	4510	4529	4547	4567						
AX0.16 = 002324	2644#	4471	4490	4510	4529	4547	4567							
AX1 = 000001	2426#	2498#												
AX1.15 = 002326	2645#	4471	4490	4510	4529	4547	4567							
AX1.16 = 002330	2646#	4471	4490	4510	4529	4547	4567							
AX2 = 000002	2425#	2497#												
AX2.15 = 002332	2647#	4473	4492	4512	4531	4549	4569							
AX2.16 = 002334	2648#	4473	4492	4512	4531	4549	4569							
AX3.15 = 002336	2649#	4473	4492	4512	4531	4549	4569							
AX3.16 = 002340	2650#	4473	4492	4512	4531	4549	4569							
AX315U = 000372	2593#	5358	5418	5635										
A1 024222	5246	5272	5291	5300	5310#									
A10 034204	6850	6925#												
A2 026630	5748	5787#												
A3 027310	5812	5916	5920#											
A4 027424	5942	5968#												
A5 027534	5991	6014#												
A6 027730	6039	6074#												
A7 030274	6100	6116	6125	6138	6157	6161	6165#							
A8 030510	6182	6222#												
A9 031040	6240	6302#												
BADDAT 002406	2717#	4072*	4073*	4094*	4095*	4450	4465	4504	4561	4834*	4868*	4899*	4932*	
	4968*	5010*	5062*	5098*	5133*	5167*	5202*	5266*	5285*	5304*	5363*	5374*	5423*	
	5433*	5480*	5490*	5531*	5540*	5578*	5587*	5641*	5651*	5699*	5722*	5769*	5779*	
	5884*	5907*	6385*	6400*	6483*	6548*	6629*	6673*	6777*					
BCC = 000001	2474#	4098	4100	4106										
BCCCHK = 100000	2658#	4096												
BIT0 = 000001 G	2354#	2376	2388	2397	2426	2438	2450	2462	2474	2486	2498	2510	2522	
	2534	2546	2555	2567	2580	2592	2603	2675	3445	3486	3615	3827	3868	
	3908	4278	4669											
BIT00 = 000001 G	2354#													
BIT01 = 000002 G	2354#													
BIT02 = 000004 G	2354#													
BIT03 = 000010 G	2354#													
BIT04 = 000020 G	2354#													
BIT05 = 000040 G	2354#													
BIT06 = 000100 G	2354#													
BIT07 = 000200 G	2354#													
BIT08 = 000400 G	2354#													
BIT09 = 001000 G	2354#													
BIT1 = 000002 G	2354#	2375	2387	2396	2415	2425	2437	2449	2461	2473	2485	2497	2509	
	2521	2533	2545	2554	2566	2579	2591	2602	2676	3502	3811	3922		

BIT10 = 002000 G	2354#	2613	2625											
BIT11 = 004000 G	2354#	2612	2624											
BIT12 = 010000 G	2354#													
BIT13 = 020000 G	2354#													
BIT14 = 040000 G	2354#													
BIT15 = 100000 G	2354#	2656	2658	2659	3583	3737								
BIT2 = 000004 G	2354#	2374	2386	2395	2414	2424	2436	2448	2460	2472	2484	2496	2508	
	2520	2532	2544	2553	2565	2578	2590	2601						
BIT3 = 000010 G	2354#	2373	2385	2394	2413	2423	2435	2447	2459	2471	2483	2495	2507	
	2519	2531	2543	2552	2564	2577	2589							
BIT4 = 000020 G	2354#	2372	2384	2412	2422	2434	2446	2458	2470	2482	2494	2506	2518	
	2530	2542	2563	2576	2588									
BIT5 = 000040 G	2354#	2383	2404	2411	2421	2433	2445	2457	2469	2481	2493	2505	2517	
	2529	2541	2562	2575	2587	2600								
BIT6 = 000100 G	2354#	2371	2382	2403	2410	2420	2432	2444	2456	2468	2480	2492	2504	
	2516	2528	2540	2561	2574	2586	2599							
BIT7 = 000200 G	2354#	2370	2381	2393	2402	2409	2431	2443	2455	2467	2479	2491	2503	
	2515	2527	2539	2551	2560	2573	2585	2598	4066	4067				
BIT8 = 000400 G	2354#	2615	2627											
BIT9 = 001000 G	2354#	2614	2626											
BOE = 000400 G	2354#													
BPOLL = 000100	2403#													
BSEL1 002450	2736#	3231*	3233*	3234*	3248*	3249*	3254*	3346*	3348*	3449*	3587*	3588*	3590*	
	3680*	3681*	3686*	3699*	4684*	4685*	4686	4921*	5623*					
BSEL2 002452	2737#	4686*	4687*	5259	5279	5298								
BSEL4 002454	2738#	3284	3305*	4960*										
CARR = 000001	2486#													
CHPCHK= 100000	2656#	5948	5997	6048	6106	6148	6193	6246	6330	6960	7112	7228		
CHPTYP 002430	2726#	3584	3694*	3698*	3738	4638*	6734	6756						
CKDATA 007750	4057#	6876	6905	7071	7074	7115	7118	7121	7124	7127	7165	7170	7178	
	7235	7238	7241	7244										
CKTBIT 011176	4274#	6332	7113	7229										
CRCCHK= 100000	2659#	4329												
CRCTYO= 000001	2580#													
CRCTY1= 000002	2579#													
CRCTY2= 000004	2578#													
CRC1 = 000100	2432#	6042	6522	6585	6704	6948	7000	7065	7103	7155	7212			
CRC2 = 000200	2431#	6042	6522	6585	6704	6948	7000	7065	7103	7155	7212			
CS = 000004	2484#													
C\$AU = 000052	2211#	4780												
C\$AUTO= 000061	2211#	4725												
C\$BRK = 000022	2211#													
C\$BSEG= 000004	2211#	4859	5090	5124	5159	5194	5469	5520	5567	5627	5689			
C\$BSUB= 000002	2211#	5687	5713	5749										
C\$CEFG= 000045	2211#													
C\$CLCK= 000062	2211#													
C\$CLEA= 000012	2211#	4742												
C\$CLOS= 000035	2211#													
C\$CLP1= 000006	2211#													
C\$CVEC= 000036	2211#													
C\$DCLN= 000044	2211#													
C\$DODU= 000051	2211#	4722												
C\$DRPT= 000024	2211#													
C\$DU = 000053	2211#	4761												
C\$EDIT= 000003	2211#	2251												
C\$ERDF= 000055	2211#	3492	3498	3508	3514	3621	3627	3785	3817	3823	3833	3839	3874	

EM4	012241	4368#	4935															
EM40	013462	4141	4404#															
EM41	013476	4150	4405#															
EM42	013517	4156	4406#															
EM43	013534	4407#																
EM44	013561	4408#																
EM45	013606	4409#																
EM46	013633	4410#																
EM47	013666	4411#																
EM48	013721	4412#																
EM49	013754	4413#																
EM5	012314	4369#	5269	5307														
EM50	014013	4414#																
EM51	014047	4415#																
EM52	014107	4416#	5340															
EM53	014150	4417#	5354															
EM54	014210	4080	4418#															
EM6	012345	4370#	5288															
EM60	014232	4419#	6072															
EM65	014246	3785	4420#															
EM7	012376	3492	4371#															
EM8	012413	3498	4372#															
EM9	012434	3508	4373#															
ENAX =	000004	2424#	2496#	3369	3420													
ENDIT	021466	4660	4698#															
ENDPAT	003220	3135#																
ENDTRN	006274	3767#	5967	6013	6073	6221	6301											
EOM =	000002	2396#																
ERRFLG	002356	2705#																
ERROR1	002420	2722#	3365*	3378*	3403*	3429*	4639*	5335	5349									
ERR1	014602	4440#	4811															
ERR2	014634	4446#	4837	4871	4902	4935	4971	5013	5065	5101	5136	5170	5205	5269				
		5288	5307															
ERR3	015142	4461#	5366	5377	5426	5436	5483	5493	5534	5543	5581	5590	5644	5654				
		5725	5772	5782	5887	5910	6388	6403	6486	6551	6632	6676	6780					
ERR4	015624	3492	3498	3508	3514	3621	3627	3785	3817	3823	3833	3839	3874	3880				
		4080	4289	4295	4480#													
ERR5	016302	4499#	5702															
ERR6	017014	3914	3920	3928	3934	4519#												
ERR7	017472	4538#	6072	6115	6124	6137	6156	6164										
ERR8	020124	4085	4104	4110	4119	4125	4135	4141	4150	4156	4556#							
ERR9	020632	4576#	5340	5354														
EVL =	000004	2354#																
ESEND =	002100	2211#																
ESLOAD =	000035	2211#	2251															
FMT1	011454	4349#	4441	4447	4462	4482	4500	4521	4539	4558	4577							
FMT10	011717	4358#	4481	4520	4557													
FMT11	011750	4359#	4501															
FMT19	012007	4360#	5245															
FMT2	011464	4350#	4448	4463	4483	4502	4522	4540	4559	4578								
FMT24	012040	4361#	5271	5290	5309													
FMT27	021602	4760	4763#															
FMT3	011506	4351#	4450	4465	4504	4561												
FMT4	011550	4352#	4451	4466	4470	4485	4489	4505	4509	4524	4528	4542	4546	4562				
		4566	4580															
FMT5	011563	4353#	4452	4467	4471	4486	4490	4506	4510	4525	4529	4543	4547	4563				

FMT6	011613	4567	4581											
		4354#	4454	4469	4473	4488	4492	4508	4512	4527	4531	4545	4549	4565
		4569	4583											
FMT7	011646	4355#	4449	4484	4541	4560	4579							
FMT8	011656	4356#	4464	4503	4523									
FMT9	011712	4357#	4453	4468	4472	4487	4491	4507	4511	4526	4530	4544	4548	4564
		4568	4582											
FRSTIM	002412	2719#	4641	4648*										
FSAU =	000015	2211#	4779	4780										
FSAUTO=	000020	2211#	4713	4725										
FSEGN =	000040	2211#	2217	4440	4446	4461	4480	4499	4519	4538	4556	4576	4599	4615
		4633	4713	4739	4756	4779	4801	4814	4826	4839	4854	4859	4877	4888
		4904	4915	4937	4953	4973	4990	5014	5019	5034	5066	5073	5085	5090
		5107	5119	5124	5142	5154	5159	5176	5188	5194	5211	5238	5312	5328
		5341	5355	5367	5378	5383	5399	5427	5437	5444	5465	5469	5484	5499
		5516	5520	5535	5549	5563	5567	5582	5596	5621	5627	5645	5661	5678
		5687	5689	5709	5713	5727	5728	5746	5749	5773	5783	5788	5789	5811
		5888	5911	5921	5941	5970	5990	6016	6038	6076	6096	6166	6181	6223
		6239	6303	6318	6338	6354	6418	6434	6501	6517	6564	6580	6647	6660
		6679	6696	6825	6849	6926	6944	6978	6996	7046	7061	7078	7099	7132
		7151	7182	7206	7253	7280	7313	7337						
FSCLEA=	000007	2211#	4739	4742										
FSDU =	000016	2211#	4756	4761										
FSEND =	000041	2211#	2217	4442	4455	4474	4493	4513	4532	4550	4570	4584	4601	4699
		4725	4742	4761	4780	4801	4814	4826	4839	4854	4873	4877	4888	4904
		4915	4937	4953	4973	4990	5014	5019	5034	5066	5073	5085	5103	5107
		5119	5138	5142	5154	5172	5176	5188	5207	5211	5238	5312	5328	5341
		5355	5367	5378	5383	5399	5427	5437	5444	5465	5484	5495	5499	5516
		5535	5545	5549	5563	5582	5592	5596	5621	5645	5656	5661	5678	5687
		5704	5709	5713	5727	5728	5746	5749	5773	5783	5788	5789	5811	5888
		5911	5921	5941	5970	5990	6016	6038	6076	6096	6166	6181	6223	6239
		6303	6318	6338	6354	6418	6434	6501	6517	6564	6580	6647	6660	6679
		6696	6825	6849	6926	6944	6978	6996	7046	7061	7078	7099	7132	7151
		7182	7206	7253	7287	7316	7337							
FSHARD=	000004	2211#	7280	7287										
FSHW =	000013	2211#	2296	2310										
FSINIT=	000006	2211#	4633	4699										
FSJMP =	000050	2211#												
FSMOD =	000000	2211#	2217	7337										
FSMSG =	000011	2211#	4440	4442	4446	4455	4461	4474	4480	4493	4499	4513	4519	4532
		4538	4550	4556	4570	4576	4584							
FSPROT=	000021	2211#	4615	4619										
FSPWR =	000017	2211#												
FSRPT =	000012	2211#	4599	4601										
FSSEG =	000003	2211#	4859	4873	5090	5103	5124	5138	5159	5172	5194	5207	5469	5495
		5520	5545	5567	5592	5627	5656	5689	5704					
		2211#	7313	7316										
FSSOFT=	000005	2211#												
FSSRV =	000010	2211#												
FSSUB =	000002	2211#	5687	5709	5713	5727	5749	5788						
FSSW =	000014	2211#	2324	2327										
FSTEST=	000001	2211#	4801	4814	4826	4839	4854	4877	4888	4904	4915	4937	4953	4973
		4990	5019	5034	5073	5085	5107	5119	5142	5154	5176	5188	5211	5238
		5312	5328	5383	5399	5444	5465	5499	5516	5549	5563	5596	5621	5661
		5678	5728	5746	5789	5811	5921	5941	5970	5990	6016	6038	6076	6096
		6166	6181	6223	6239	6303	6318	6338	6354	6418	6434	6501	6517	6564
		6580	6647	6660	6679	6696	6825	6849	6926	6944	6978	6996	7046	7061

GETALL 004526	7078	7099	7132	7151	7182	7206	7253							
	3442#	3490	3496	3506	3512	3619	3625	3783	3815	3821	3831	3837	3872	
	3878	3912	3918	3926	3932	4078	4083	4102	4108	4117	4123	4133	4139	
	4148	4154	4287	4293	5364	5375	5424	5434	5481	5491	5532	5541	5579	
	5588	5642	5652	5700	5723	5770	5780	5885	5908	6070	6113	6122	6135	
	6154	6162	6386	6401	6484	6549	6630	6674	6778					
GETPRM 021310	4661	4672#	4679											
GETREG 004016	3319#	3448	4835	4869	4900	4933	4969	5011	5063	5099	5134	5168	5203	
	5267	5286	5305	5338	5352									
GOAH = 000010	2394#													
GOODAT 002404	2716#	4070*	4071*	4092*	4093*	4450	4465	4504	4561	4832*	4833*	4867*	4897*	
	4898*	4930*	4931*	4966*	4967*	5008*	5009*	5060*	5061*	5097*	5132*	5166*	5201*	
	5265*	5284*	5303*	5361*	5362*	5372*	5373*	5421*	5422*	5431*	5432*	5479*	5488*	
	5489*	5530*	5539*	5577*	5586*	5639*	5640*	5649*	5650*	5697*	5698*	5720*	5721*	
	5767*	5768*	5777*	5778*	5883*	5906*	6384*	6399*	6409*	6482*	6547*	6628*	6672*	
	6776*	6796*	6812*											
GSCNTO= 000200	2211#													
GSDLM= 000372	2211#													
GSDISP= 000003	2211#													
GSEXCP= 000400	2211#													
GSHILI= 000002	2211#													
GSLOLI= 000001	2211#													
GSNO = 000000	2211#													
GSOFFS= 000400	2211#	7282	7283	7284	7285									
GSOFSI= 000376	2211#	7282	7283	7284	7285									
GSPRMA= 000001	2211#	7282	7283											
GSPRMD= 000002	2211#	7284	7285											
GSPRML= 000000	2211#													
GSRADA= 000140	2211#													
GSRADB= 000000	2211#													
GSRADL= 000040	2211#													
GSRADL= 000120	2211#													
GSRADO= 000020	2211#	7282	7283	7284	7285									
GSXFER= 000004	2211#													
GYES = 000010	2211#	7282	7283	7284	7285									
HDX = 000020	2412#	2482#												
HELP = 000001	2197#	2243	2253	2276	3204									
HOE = 100000 G	2354#													
IACT = 000100	2468#	3870	3876											
IACTIV 006714	3860#	3987	4008	6374	6379	6394	6407	6415	6454	6463	6470	6475	6492	
	6498	6537	6542	6557	6561	6600	6609	6616	6621	6638	6644	6742	6767	
	6788	6804												
IBE = 010000 G	2354#													
IC = 000200	2402#	2467#	6496	6559	6642									
ICIR = 000010	2507#	3829	3835											
IDL = 000010	2577#													
IDLE = 000040	2433#	6145	6185	6948	7065	7155								
IDU = 000040 G	2354#													
IER = 020000 G	2354#													
IERR = 000002	2437#	4032	4038	6359	6439	6522	6585	6704						
INITRN 005540	3651#	5943	5992	6040	6101	6143	6183	6241	6319	6698	6861	6946	6998	
	7063	7101	7153	7210										
INTFLG 002354	2703#													
INTGRL= 000010	2589#	2593												
IRDY = 000020	2470#	3813	3819	3998										
ISIRDY 006430	3803#	3989	4002	4006	4010	4250	4256	6754	6763	6771	6784	6792	6800	

	6808	6855	6880	6890	6899	6911								
ISR = 000100 G	2354#													
ISRUN 035732	7285	7292#												
IXE = 004000 G	2354#													
ISAU = 000041	2211#	4779#	4780#											
ISAUTO= 000041	2211#	4713#	4725#											
ISCLN = 000041	2211#	4739#	4742#											
ISDU = 000041	2211#	4756#	4761#											
ISHRD = 000041	7280#	7287#												
ISINIT= 000041	2211#	4633#	4699#											
ISMOD = 000041	"	2217#	7337#											
ISMSG = 000041	2211#	4440#	4442#	4446#	4455#	4461#	4474#	4480#	4493#	4499#	4513#	4519#	4532#	
	4538#	4550#	4556#	4570#	4576#	4584#								
ISPROT= 000040	2211#	4615#												
ISPTAB= 000041	2211#													
ISPWR = 000041	2211#													
ISRPT = 000041	2211#	4599#	4601#											
ISSEG = 000041	2211#	4801	4826	4854	4859#	4873#	4888	4915	4953	4990	5034	5085	5090#	
	5103#	5119	5124#	5138#	5154	5159#	5172#	5188	5194#	5207#	5238	5328	5399	
	5465	5469#	5484	5495#	5516	5520#	5535	5545#	5563	5567#	5582	5592#	5621	
	5627#	5645	5656#	5678	5687	5689#	5704#	5713	5746	5749	5811	5941	5990	
	6038	6096	6181	6239	6318	6354	6434	6517	6580	6660	6696	6849	6944	
	6996	7061	7099	7151	7206									
ISSETU= 000041	2211#													
ISSFT = 000041	7313#	7316#												
ISSRV = 000041	2211#													
ISSUB = 000041	2211#	4801	4826	4854	4888	4915	4953	4990	5034	5085	5119	5154	5188	
	5238	5328	5399	5465	5516	5563	5621	5678	5687#	5709#	5713#	5727#	5746	
	5749#	5773	5783	5788#	5811	5941	5990	6038	6096	6181	6239	6318	6354	
	6434	6517	6580	6660	6696	6849	6944	6996	7061	7099	7151	7206		
ISTST = 000041	2211#	4801#	4814#	4826#	4839#	4854#	4877#	4888#	4904#	4915#	4937#	4953#	4973#	
	4990#	5014	5019#	5034#	5066	5073#	5085#	5107#	5119#	5142#	5154#	5176#	5188#	
	5211#	5238#	5312#	5328#	5341	5355	5367	5378	5383#	5399#	5427	5437	5444#	
	5465#	5499#	5516#	5549#	5563#	5596#	5621#	5661#	5678#	5687	5713	5728#	5746#	
	5749	5789#	5811#	5888	5911	5921#	5941#	5970#	5990#	6016#	6038#	6076#	6096#	
	6166#	6181#	6223#	6239#	6303#	6318#	6338#	6354#	6418#	6434#	6501#	6517#	6564#	
	6580#	6647#	6660#	6679#	6696#	6825#	6849#	6926#	6944#	6978#	6996#	7046#	7061#	
	7078#	7099#	7132#	7151#	7182#	7206#	7253#							
I422 = 000200	2585#	2593												
JSJMP = 000167	2211#													
LDMSG1 011374	4319#													
LDTXSI 005174	3558#	3671	3672	3730	4223	5823	5824	5848	5896	6127	6325	6368	6448	
	6531	6594	6667	6866	6888	6918	6922	6951	7003	7067	7108	7109	7158	
	7215	7222	7223											
LOADAT 002410	2718#	4501	5695*	5696*										
LODMSG 011016	4216#	4321	7104	7218										
LOE = 040000 G	2354#													
LOGDEV 002344	2699#	4667*	4673*	4674	4676	4722	4760							
LOOPIN 004074	3345#	5252												
LOT = 000010 G	2354#													
LULoop= 000010	2373#	3254	3449	3587	3680	5623								
LULP = 000040	2404#	2469#	6364	6444	6527	6590	6712							
LUREG 002302	2635	2636	2637	2638	2639	2640	2641	2642	2643	2644	2645	2646	2647	
	2648	2649	2650	2693#										
LUR10 = 002302	2635#	3321	4452	4467	4486	4506	4525	4543	4563	4581				
LUR11 = 002304	2636#	4452	4467	4486	4506	4525	4543	4563	4581					

LSSPCP	002020	G	2251#				
LSSPTP	002024	G	2251#				
LSSSTA	002030	G	2251#				
LSSW	002302	G	2324#				
LSTEST	002114	G	2251#				
LSTIML	002014	G	2251#				
LSUNIT	002012	G	2251#	4674			
L10000	002300		2296	2310#			
L10001	002302		2324	2327#			
L10002	014632		4442#				
L10003	015140		4455#				
L10004	015622		4474#				
L10005	016300		4493#				
L10006	017012		4513#				
L10007	017470		4532#				
L10010	020122		4550#				
L10011	020630		4570#				
L10012	021106		4584#				
L10013	021110		4601#				
L10015	021466		4699#				
L10016	021546		4725#				
L10017	021550		4742#				
L10020	021600		4761#				
L10021	021634		4780#				
L10022	021716		4814#				
L10023	022002		4839#				
L10024	022126		4877#				
L10025	022230		4904#				
L10026	022356		4937#				
L10027	022476		4973#				
L10030	022652		5014	5019#			
L10031	023106		5066	5073#			
L10032	023224		5107#				
L10033	023342		5142#				
L10034	023460		5176#				
L10035	023576		5211#				
L10036	024226		5312#				
L10037	024560		5341	5355	5367	5378	5383#
L10040	025050		5427	5437	5444#		
L10041	025260		5499#				
L10042	025464		5549#				
L10043	025670		5596#				
L10044	026126		5661#				
L10045	026360		5728#				
L10046	026264		5709#				
L10047	026356		5727#				
L10050	026632		5789#				
L10051	026630		5773	5783	5788#		
L10052	027310		5888	5911	5921#		
L10053	027430		5970#				
L10054	027540		6016#				
L10055	027734		6076#				
L10056	030300		6166#				
L10057	030510		6223#				
L10060	031040		6303#				
L10061	031140		6338#				

PATA	002571	2805#	4857																	
PATB	002615	2828#	4875	5045																
PATC	002625	2839#	5088																	
PATCH	036022	7328#																		
PATD	002630	2845#	5105	5122																
PATE	002633	2851#	5140	5157																
PATF	002641	2860#	5174	5191																
PATG	002644	2866#	4993	5209																
PATH	002654	2877#	4997	5331																
PATI	002664	2888#	5343	5347	5401	5486	5489	5718	5721											
PATJ	002674	2899#	5381	5404	5408	5409														
PATK	002704	2910#	5467	5565																
PATL	003014	2984#	5497	5518	5594															
PATM	003022	2993#	4830	4833	4895	4898	4928	4931	4964	4967	5001	5037	5547							
PATN	003032	3004#	5017	5747																
PATO	003050	3021#	5680	5785																
PATP	003066	3038#	5681	5707																
PATU	003104	3055#	5625																	
PATV	003152	3095#	5624	5659																
PNT =	001000	2354#																		
POLL =	000200	2409#																		
PRI =	002000	2354#																		
PRIOR	002350	2701#																		
PRIRTY	035701	7284	7291#																	
PRI00 =	000000	2354#																		
PRI01 =	000040	2354#																		
PRI02 =	000100	2354#																		
PRI03 =	000140	2354#																		
PRI04 =	000200	2354#																		
PRI05 =	000240	2354#																		
PRI06 =	000300	2354#																		
PRI07 =	000340	2354#	4715	4717	4803	4805														
PSTACK	002346	2700#	3516	3629	3841	3882	3936	4165	4301	4635*										
RAB =	000004	2472#	4129	4131	4137															
RABT =	000004	2532#																		
RAX15	002370	2710#	3382*	3383*	3453	5358*	5359	5363	5418*	5419	5423	5477	5480	5528						
		5531	5575	5578	5635*	5636*	5637	5641	5765	5769	5881	5884	5904	5907						
		6382	6385	6397	6400	6411	6480	6483	6545	6548	6626	6629	6670	6673						
		6774	6777	6794	6810															
RAX16	002372	2711#	3386*	3387*	3455	3910	3916	3924	3930	5370	5374	5429	5433	5486						
		5490	5537	5540	5584	5587	5647	5651	5693	5699	5718	5722	5775	5779						
RCVBUF	003262	3168#	4325																	
RCV1ST	007434	3976#	6870	6963	7009	7162	7233													
RDALL =	000004	2436#																		
RDAX =	000020	2422#	2494#	3369																
RDRXSI	007354	3952#	4063	4245	7070															
READAX	004124	3363#	3452	3907	5348	5415	5476	5527	5574	5632	5692	5717	5754	5880						
		5903	6381	6396	6410	6479	6544	6625	6669											
READLU	003672	3272#	3323	3373	3381	3385	3424	3485	3501	3614	3683	3695	3780	3810						
		3826	3867	3954	3958	3997	4075	4282	4829	4863	4894	4926	4963	5004						
		5052	5093	5127	5162	5197	5835	5867	6067	6110	6119	6132	6151	6159						
READY =	000200	2491#	3374	3425																
REDBYT	002364	2708#	3284*	3285*	3324	3374	3382	3386	3425	3488	3494	3504	3510	3617						
		3623	3684	3696	3781	3813	3819	3829	3835	3870	3876	3955	3959	3998						
		4076	4285	4291	4830	4834	4864*	4865	4868	4895	4899	4927*	4928	4932						
		4964	4968	5005*	5006	5010	5053*	5056*	5058	5062	5094*	5095	5098	5128*						

RX7 = 000200	2443#	2515#												
R14NRW 002560	2791#	4861	4864											
SAVE4 002414	2720#	4643*	4646	4723	4812									
SAVE6 002416	2721#	4644*	4647	4724	4813									
SAVLEN 002432	2727#	3260*	4064	4199*	4640*									
SCRACH 002342	2698#													
SEC = 000020	2576#													
SECA = 000020	2434#													
SEFR = 000040	2411#													
SELSBY= 000002	2415#													
SEL4 002454	2739#	4688*	4689*											
SEL6 002456	2740#	3232*	3347*	4690*	4691*									
SETUP 010640	4183#	6357	6437	6520	6583	6702								
SFPTBL 002302 G	2324#													
SIGQ = 000100	2504#													
SIGR = 000200	2503#													
SOM = 000001	2397#													
STALL 005164	3545#	3589	3591	3682	3687	3700								
STARES 002444	2732#	4663*	4668*	5243										
STARST 021260	4651	4654	4662#											
STBY = 000002	2485#													
STEPLU= 000020	2372#	3588	3590	3681	3686	3699								
STEPMP= 000001	2376#	3233	3234											
STPCLK 003540	3230#	3282	3306	4961										
STPERR 007646	4029#													
STPLU 005254	3579#	3747	3993	4036	4161	4252	4280	5833	5865	5901	6064	6107	6117	
	6129	6149	6329	6372	6376	6390	6405	6413	6452	6458	6488	6535	6539	
	6553	6598	6604	6634	6725	6746	6761	6782	6798	6885	6897	6917	6959	
	7068	7111	7176	7227										
	2575#													
STR = 000040	2435#	5945	6042	6103	6145	6321	6359	6522	6863	7103	7212			
STRIP = 000010	2702#	3479	3480	3482*	3483*	3520*	3608	3609	3611*	3612*	3633*			
SUBRPC 002352	3707*	3727*	3728*	3754*	3769*	3770*	3786*	3804	3805	3807*	3808*	3654*	3655*	
	3862	3864*	3865*	3886*	3901	3902	3904*	3905*	3940*	3979*	3980*	3845*	3861	
	4060*	4171*	4204*	4243*	4244*	4259*	4276*	4277*	4307*	4481	4520	4016*	4059*	
SVCGBL- 000000	2211#	2217	2225#	2251	2274	2296	2324	3191	3197	3492	3498	3508	3514	3621
	4499	4519	4538	4556	4576	4599	4615	4633	4713	4739	4756	4779	4480	
	7313	7339#												
SVCINS= 000001	2211#	2222#	2251	2274	2296	2324	3191	3197	3492	3498	3508	3514	3621	
	3627	3785	3817	3823	3833	3839	3874	3880	3917	3920	3928	3934	4080	
	4085	4104	4110	4119	4125	4135	4141	4150	4156	4289	4295	4441	4442	
	4447	4448	4449	4450	4451	4452	4453	4454	4455	4462	4463	4464	4465	
	4466	4467	4468	4469	4470	4471	4472	4473	4474	4481	4482	4483	4484	
	4485	4486	4487	4488	4489	4490	4491	4492	4493	4500	4501	4502	4503	
	4504	4505	4506	4507	4508	4509	4510	4511	4512	4513	4520	4521	4522	
	4523	4524	4525	4526	4527	4528	4529	4530	4531	4532	4539	4540	4541	
	4542	4543	4544	4545	4546	4547	4548	4549	4550	4557	4558	4559	4560	
	4561	4562	4563	4564	4565	4566	4567	4568	4569	4570	4577	4578	4579	
	4580	4581	4582	4583	4584	4601	4650	4651	4653	4654	4656	4657	4659	
	4660	4676	4677	4699	4715	4722	4725	4742	4758	4760	4761	4780	4803	
	4811	4814	4837	4839	4859	4871	4873	4877	4902	4904	4920	4935	4937	
	4971	4973	5013	5014	5019	5065	5066	5073	5090	5101	5103	5107	5124	
	5136	5138	5142	5159	5170	5172	5176	5194	5205	5207	5211	5245	5269	
	5271	5288	5290	5307	5309	5312	5340	5341	5354	5355	5366	5367	5377	
	5378	5383	5426	5427	5436	5437	5444	5469	5483	5484	5493	5495	5499	
	5520	5534	5535	5543	5545	5549	5567	5581	5582	5590	5592	5596	5627	

CZDMRC MB203 STATIC DIAG #1 CZDMR..P11 13-MAR-80 14:38		MACY11 30A(1052) 13-MAR-80 14:40		G 15 PAGE 8-15 CROSS REFERENCE TABLE -- USER SYMBOLS											SEQ 0188
TXCHAR 006122	3725#	5946	5949	5952	5955	5958	5961	5964	5995	5998	6001	6004	6007		
	6010	6046	6049	6052	6055	6058	6061	6104	6146	6190	6194	6199	6204		
	6209	6212	6215	6218	6244	6247	6250	6257	6262	6267	6272	6277	6282		
	6287	6292	6295	6298											
TXDATA= 000040	2505#	4285	4291												
TXEN = 000100	2420#	2492#	4927	6044	6708	6710									
TXEOM = 001000	2614#	3151	3152	3153	3154	3160	3161	5953	5956	5959	5962	5965	6002		
	6005	6008	6011	6053	6056	6059	6062	6210	6213	6216	6219	6293	6296		
	6299	6921	7107	7221											
TXGA = 000010	2552#														
TXGOA = 004000	2612#	7107													
TXLENO= 000040	2600#	4064	6188	6202	6255	6265	6275	6285							
TXLEN1= 000100	2599#	4064	6197	6202	6260	6265	6280	6285							
TXLEN2= 000200	2598#	4064	6188	6197	6202	6270	6275	6280	6285						
TXSOM = 000400	2615#	3144	3145	3663	5822	6126	7213		6285						
TXWORD 002422	2723#	3559*	3561	3564	3663*	3664*	3729*	4222*	5822*	5847*	5895*	6126*	6324*		
	6367*	6447*	6530*	6593*	6666*	6865*	6887*	6917*	6921*	6950*	7002*	7066*	7107*		
	7157*	7213*	7221*												
TX0 = 000001	2388#	2546#													
TX1 = 000002	2387#	2545#													
TX2 = 000004	2386#	2544#													
TX3 = 000010	2385#	2543#													
TX4 = 000020	2384#	2542#													
TX5 = 000040	2383#	2541#													
TX6 = 000100	2382#	2540#													
TX7 = 000200	2381#	2539#													
T\$ARGC= 000001	2251#	4441#	4447#	4448#	4449#	4450#	4451#	4452#	4453#	4454#	4462#	4463#	4464#		
	4465#	4466#	4467#	4468#	4469#	4470#	4471#	4472#	4473#	4481#	4482#	4483#	4484#		
	4485#	4486#	4487#	4488#	4489#	4490#	4491#	4492#	4500#	4501#	4502#	4503#	4504#		
	4505#	4506#	4507#	4508#	4509#	4510#	4511#	4512#	4520#	4521#	4522#	4523#	4524#		
	4525#	4526#	4527#	4528#	4529#	4530#	4531#	4539#	4540#	4541#	4542#	4543#	4544#		
	4545#	4546#	4547#	4548#	4549#	4557#	4558#	4559#	4560#	4561#	4562#	4563#	4564#		
	4565#	4566#	4567#	4568#	4569#	4577#	4578#	4579#	4580#	4581#	4582#	4583#	4760#		
	5245#	5271#	5290#	5309#											
T\$CODE= 012032	7282#	7283#	7284#	7285#											
T\$ERRN= 000032	2211#	3492#	3498#	3508#	3514#	3621#	3627#	3785#	3817#	3823#	3833#	3839#	3874#		
	3880#	3914#	3920#	3928#	3934#	4080#	4085#	4104#	4110#	4119#	4125#	4135#	4141#		
	4150#	4156#	4289#	4295#	4811#	4837#	4871#	4902#	4935#	4971#	5013#	5065#	5101#		
	5136#	5170#	5205#	5269#	5288#	5307#	5340#	5354#	5366#	5377#	5426#	5436#	5483#		
	5493#	5534#	5543#	5581#	5590#	5644#	5654#	5702#	5725#	5772#	5782#	5887#	5910#		
	6072#	6115#	6124#	6137#	6156#	6164#	6388#	6403#	6486#	6551#	6632#	6676#	6780#		
T\$EXCP= 000000	7282#	7283#	7284#	7285#											
T\$FLAG= 000040	5014#	5066#	5341#	5355#	5367#	5378#	5427#	5437#	5484#	5535#	5582#	5645#	5773#		
	5783#	5888#	5911#												
T\$GMAN= 000000	2211#														
T\$HILI= 000007	7282#	7283#	7284#	7285#											
T\$LAST= 000001	2211#	7339#													
T\$LOLI= 000000	7282#	7283#	7284#	7285#											
T\$LSYM= 010000	2211#	2310	2327	4442	4455	4474	4493	4513	4532	4550	4570	4584	4601		
	4699	4725	4742	4761	4780	4814	4839	4877	4904	4937	4973	5019	5073		
	5107	5142	5176	5211	5312	5383	5444	5499	5549	5596	5661	5709	5727		
	5728	5788	5789	5921	5970	6016	6076	6166	6223	6303	6338	6418	6501		
	6564	6647	6679	6825	6926	6978	7046	7078	7132	7182	7253	7287	7316		
T\$LTNO= 000052	7339#														
T\$NEST= 177777	2211#	2217#	2296#	2310#	2324#	2327#	4440#	4442#	4446#	4455#	4461#	4474#	4480#		
	4493#	4499#	4513#	4519#	4532#	4538#	4550#	4556#	4570#	4576#	4584#	4599#	4601#		

TSNSO = 000000
TSNS1 = 000005

TSNS2 = 000002

TSNS3 = 000003
TSPTNU= 000000
TSSAVL= 177777
T\$SEGL= 177777

T\$SEKO= 010000

T\$SUBN= 000000

T\$TAGL= 177777
T\$TAGN= 010101

T\$TEMP= 000000

T\$TEST= 000052

T\$TSTM= 177777

4615#	4619#	4633#	4699#	4713#	4725#	4739#	4742#	4756#	4761#	4779#	4780#	4801#
4814#	4826#	4839#	4854#	4859#	4873#	4877#	4888#	4904#	4915#	4937#	4953#	4973#
4990#	5019#	5034#	5073#	5085#	5090#	5103#	5107#	5119#	5124#	5138#	5142#	5154#
5159#	5172#	5176#	5188#	5194#	5207#	5211#	5238#	5312#	5328#	5383#	5399#	5444#
5465#	5469#	5495#	5499#	5516#	5520#	5545#	5549#	5563#	5567#	5592#	5596#	5621#
5627#	5656#	5661#	5678#	5687#	5689#	5704#	5709#	5713#	5727#	5728#	5746#	5749#
5788#	5789#	5811#	5921#	5941#	5970#	5990#	6016#	6038#	6076#	6096#	6166#	6181#
6223#	6239#	6303#	6318#	6338#	6354#	6418#	6434#	6501#	6517#	6564#	6580#	6647#
6660#	6679#	6696#	6825#	6849#	6926#	6944#	6978#	6996#	7046#	7061#	7078#	7099#
7132#	7151#	7182#	7206#	7253#	7280#	7287#	7313#	7316#	7337#			
2217#	7337											
2296#	2310	2324#	2327	4440#	4442	4446#	4455	4461#	4474	4480#	4493	4499#
4513	4519#	4532	4538#	4550	4556#	4570	4576#	4584	4599#	4601	4615#	4619
4633#	4699	4713#	4725	4739#	4742	4756#	4761	4779#	4780	4801#	4814	4826#
4839	4854#	4877	4888#	4904	4915#	4937	4953#	4973	4990#	5019	5034#	5073
5085#	5107	5119#	5142	5154#	5176	5188#	5211	5238#	5312	5328#	5383	5399#
5444	5465#	5499	5516#	5549	5563#	5596	5621#	5661	5678#	5728	5746#	5789
5811#	5921	5941#	5970	5990#	6016	6038#	6076	6096#	6166	6181#	6223	6239#
6303	6318#	6338	6354#	6418	6434#	6501	6517#	6564	6580#	6647	6660#	6679
6696#	6825	6849#	6926	6944#	6978	6996#	7046	7061#	7078	7099#	7132	7151#
7182	7206#	7253	7280#	7287	7313#	7316						
4859#	4873	5090#	5103	5124#	5138	5159#	5172	5194#	5207	5469#	5495	5520#
5545	5567#	5592	5627#	5656	5687#	5709	5713#	5727	5749#	5788		
5689#	5704											
2211#												
2211#	4859#	4873#	5090#	5103#	5124#	5138#	5159#	5172#	5194#	5207#	5469#	5484
5495#	5520#	5535	5545#	5567#	5582	5592#	5627#	5645	5656#	5689#	5704#	
4859#	4873	5090#	5103	5124#	5138	5159#	5172	5194#	5207	5469#	5484	5495
5520#	5535	5545	5567#	5582	5592	5627#	5645	5656	5689#	5704		
2211#	4801#	4826#	4854#	4888#	4915#	4953#	4990#	5034#	5085#	5119#	5154#	5188#
5238#	5328#	5399#	5465#	5516#	5563#	5621#	5678#	5687#	5713#	5746#	5749#	5811#
5941#	5990#	6038#	6096#	6181#	6239#	6318#	6354#	6434#	6517#	6580#	6660#	6696#
6849#	6944#	6996#	7061#	7099#	7151#	7206#						
2211#												
2211#	2296#	2324#	4440#	4446#	4461#	4480#	4499#	4519#	4538#	4556#	4576#	4599#
4615#	4633#	4713#	4739#	4756#	4779#	4801#	4826#	4854#	4888#	4915#	4953#	4990#
5034#	5085#	5119#	5154#	5188#	5238#	5328#	5399#	5465#	5516#	5563#	5621#	5678#
5687#	5713#	5746#	5749#	5811#	5941#	5990#	6038#	6096#	6181#	6239#	6318#	6354#
6434#	6517#	6580#	6660#	6696#	6849#	6944#	6996#	7061#	7099#	7151#	7206#	7280#
7313#												
2274#	2310#	2327#	4442#	4455#	4474#	4493#	4513#	4532#	4550#	4570#	4584#	4601#
4619#	4699#	4725#	4742#	4761#	4780#	4814#	4839#	4873#	4877#	4904#	4937#	4973#
5014#	5019#	5066#	5073#	5103#	5107#	5138#	5142#	5172#	5176#	5207#	5211#	5312#
5341#	5355#	5367#	5378#	5383#	5427#	5437#	5444#	5484#	5495#	5499#	5535#	5545#
5549#	5582#	5592#	5596#	5645#	5656#	5661#	5704#	5709#	5727#	5728#	5773#	5783#
5788#	5789#	5888#	5911#	5921#	5970#	6016#	6076#	6166#	6223#	6303#	6338#	6418#
6501#	6564#	6647#	6679#	6825#	6926#	6978#	7046#	7078#	7132#	7182#	7253#	7282#
7283#	7284#	7285#	7287#	7316#	7337#							
2211#	4801#	4826#	4854#	4888#	4915#	4953#	4990#	5034#	5085#	5119#	5154#	5188#
5238#	5328#	5399#	5465#	5516#	5563#	5621#	5678#	5687	5713	5746#	5749	5811#
5941#	5990#	6038#	6096#	6181#	6239#	6318#	6354#	6434#	6517#	6580#	6660#	6696#
6849#	6944#	6996#	7061#	7099#	7151#	7206#	7339					
2211#	3492	3498	3508	3514	3621	3627	3785	3817	3823	3833	3839	3874
3880	3914	3920	3928	3934	4080	4085	4104	4110	4119	4125	4135	4141
4150	4156	4289	4295	4441	4442	4447	4448	4449	4450	4451	4452	4453

4454	4455	4462	4463	4464	4465	4466	4467	4468	4469	4470	4471	4472	
4473	4474	4481	4482	4483	4484	4485	4486	4487	4488	4489	4490	4491	
4492	4493	4500	4501	4502	4503	4504	4505	4506	4507	4508	4509	4510	
4511	4512	4513	4520	4521	4522	4523	4524	4525	4526	4527	4528	4529	
4530	4531	4532	4539	4540	4541	4542	4543	4544	4545	4546	4547	4548	
4549	4550	4557	4558	4559	4560	4561	4562	4563	4564	4565	4566	4567	
4568	4569	4570	4577	4578	4579	4580	4581	4582	4583	4584	4601	4650	
4653	4656	4659	4676	4699	4715	4722	4725	4742	4758	4760	4761	4780	
4803	4811	4814	4837	4839	4859	4871	4873	4877	4902	4904	4920	4935	
4937	4971	4973	5013	5014	5019	5065	5066	5073	5090	5101	5103	5107	
5124	5136	5138	5142	5159	5170	5172	5176	5194	5205	5207	5211	5245	
5269	5271	5288	5290	5307	5309	5312	5340	5341	5354	5355	5366	5367	
5377	5378	5383	5426	5427	5436	5437	5444	5469	5483	5484	5493	5495	
5499	5520	5534	5535	5543	5545	5549	5567	5581	5582	5590	5592	5596	
5627	5644	5645	5654	5656	5661	5687	5689	5702	5704	5709	5713	5725	
5727	5728	5749	5772	5773	5782	5783	5788	5789	5887	5888	5910	5911	
5921	5970	6016	6072	6076	6115	6124	6137	6156	6164	6166	6223	6303	
6338	6388	6403	6418	6486	6501	6551	6564	6632	6647	6676	6679	6780	
6825	6926	6978	7046	7078	7132	7182	7253						
TSTSTS= 000001	2211#	4801#	4826#	4854#	4888#	4915#	4953#	4990#	5034#	5085#	5119#	5154#	5188#
	5238#	5328#	5399#	5465#	5516#	5563#	5621#	5678#	5746#	5811#	5941#	5990#	6038#
	6096#	6181#	6239#	6318#	6354#	6434#	6517#	6580#	6660#	6696#	6849#	6944#	6996#
	7061#	7099#	7151#	7206#									
TSSAU = 010021	4779#	4780											
TSSAUT= 010016	4713#	4725											
TSSCLE= 010017	4739#	4742											
TSSDU = 010020	4756#	4761											
TSSHAR= 010077	7280#	7287											
TSSHW = 010000	2296#	2310											
TSSINI= 010015	4633#	4699											
TSSMSG= 010012	4440#	4442	4446#	4455	4461#	4474	4480#	4493	4499#	4513	4519#	4532	4538#
	4550	4556#	4570	4576#	4584								
TSSPRO= 010014	4615#												
TSSRPT= 010013	4599#	4601											
TSSSEG= 010000	4859#	4873#	5090#	5103#	5124#	5138#	5159#	5172#	5194#	5207#	5469#	5484	5495#
	5520#	5535	5545#	5567#	5582	5592#	5627#	5645	5656#	5689#	5704#		
TSSSOF= 010100	7313#	7316											
TSSSUB= 010051	5687#	5709	5713#	5727	5749#	5773	5783	5788					
TSSSW = 010001	2324#	2327											
TSTES= 010076	4801#	4814	4826#	4839	4854#	4877	4888#	4904	4915#	4937	4953#	4973	4990#
	5014	5019	5034#	5066	5073	5085#	5107	5119#	5142	5154#	5176	5188#	5211
	5238#	5312	5328#	5341	5355	5367	5378	5383	5399#	5427	5437	5444	5465#
	5499	5516#	5549	5563#	5596	5621#	5661	5678#	5728	5746#	5789	5811#	5888
	5911	5921	5941#	5970	5990#	6016	6038#	6076	6096#	6166	6181#	6223	6239#
	6303	6318#	6338	6354#	6418	6434#	6501	6517#	6564	6580#	6647	6660#	6679
	6696#	6825	6849#	6926	6944#	6978	6996#	7046	7061#	7078	7099#	7132	7151#
	7182	7206#	7253										

T1 021636 G
T10 023226 G
T11 023344 G
T12 023462 G
T13 023600 G
T14 024230 G
T15 024562 G
T16 025052 G
T17 025262 G

2274 4801#
2274 5119#
2274 5154#
2274 5188#
2274 5238#
2274 5328#
2274 5399#
2274 5465#
2274 5516#

	5132	5160*	5164	5166	5195*	5199	5201	5690*	5714*	5754*	5757*	6044*	6364*
	6444*	6496*	6527*	6559*	6590*	6642*	6663*	6708*	6712*	6815*	6818*	7174*	
WRITAX 004312	3401#	3666	4188	4200	5334	5412	5475	5526	5573	5631	6189	6198	6203
	6208	6256	6261	6266	6271	6276	6281	6286	6291	6720	6739	6760	6958
	6962	6966	6970	7012	7016	7020	7024	7028	7032	7036	7040		
WRITLU 003750	3257	3298#	3371	3408	3412	3416	3422	3562	3565	3670	3773	4033	4039
	4193	4862	4892	4919	4995	5048	5092	5126	5161	5196	5691	5715	5755
	5758	5759	6045	6365	6445	6497	6528	6560	6591	6643	6664	6709	6713
	6816	6819	7175										
XYZ = 000100	2586#	2593											
XSALWA= 000000	2211#												
XSALS= 000040	2211#												
XSOFFS= 000400	2211#												
XSTRUE= 000020	2211#												
\$LSTIN= 000001	2220#												
\$LSTTA= 000001	2221#												
. = 036234	2203#	2693#	3168#	3197#	4434#	4764#	5014	5066	5341	5355	5367	5378	5427
	5437	5484	5535	5582	5645	5773	5783	5888	5911	7329#			

ENDSRV	580#	2211#													
ENDSUB	596#	2211#	5709	5727	5788										
ENDSW	614#	2211#	2327												
ENDTST	624#	2211#	4814	4839	4877	4904	4937	4973	5019	5073	5107	5142	5176	5211	5312
	5383	5444	5499	5549	5596	5661	5728	5789	5921	5970	6016	6076	6166	6223	6303
	6338	6418	6501	6564	6647	6679	6825	6926	6978	7046	7078	7132	7182	7253	
EQUALS	642#	2211#	2354												
ERRDF	714#	2211#	3492	3498	3508	3514	3621	3627	3785	3817	3823	3833	3839	3874	3880
	3914	3920	3928	3934	4080	4085	4104	4110	4119	4125	4135	4141	4150	4156	4289
	4295	4811	4837	4871	4902	4935	4971	5013	5065	5101	5136	5170	5205	5269	5288
	5307	5340	5354	5366	5377	5426	5436	5483	5493	5534	5543	5581	5590	5644	5654
	5702	5725	5772	5782	5887	5910	6072	6115	6124	6137	6156	6164	6388	6403	6486
	6551	6632	6676	6780											
ERRHRD	718#	2211#													
ERROR	722#	2211#													
ERRSF	726#	2211#													
ERRSOF	730#	2211#													
ERRTBL	734#	2211#													
ESCAPE	744#	2211#	5014	5066	5341	5355	5367	5378	5427	5437	5484	5535	5582	5645	5773
	5783	5888	5911												
EXIT	771#	2211#													
FEQUAL	810#	2211#													
GETBYT	824#	2211#													
GETPRI	834#	2211#													
GETWOR	829#	2211#													
GMANIA	839#	2211#													
GMANID	848#	2211#													
GMANIL	859#	2211#													
GPHARD	868#	2211#	4676												
GPRMA	874#	2211#	7282	7283											
GPRMD	903#	2211#	7284	7285											
GPRML	934#	2211#													
HEADER	954#	2211#	2251												
INLOOP	962#	2211#													
IOSETU	966#	2211#													
IOSTAR	974#	2211#													
KT11	982#	2211#													
LASTAD	1147#	2211#	7339												
MANUAL	1162#	2211#													
MEMORY	1166#	2211#													
MSBYTE	2000#	2211#	2251#												
MSCHEC	2118#	2211#													
MSCNTO	2182#	2211#	7282#	7283#	7284#	7285#									
MSCOUN	2066#	2211#	4441#	4447#	4448#	4449#	4450#	4451#	4452#	4453#	4454#	4462#	4463#	4464#	4465#
	4466#	4467#	4468#	4469#	4470#	4471#	4472#	4473#	4481#	4482#	4483#	4484#	4485#	4486#	4487#
	4488#	4489#	4490#	4491#	4492#	4500#	4501#	4502#	4503#	4504#	4505#	4506#	4507#	4508#	4509#
	4510#	4511#	4512#	4520#	4521#	4522#	4523#	4524#	4525#	4526#	4527#	4528#	4529#	4530#	4531#
	4539#	4540#	4541#	4542#	4543#	4544#	4545#	4546#	4547#	4548#	4549#	4557#	4558#	4559#	4560#
	4561#	4562#	4563#	4564#	4565#	4566#	4567#	4568#	4569#	4577#	4578#	4579#	4580#	4581#	4582#
	4583#	4760#	5245#	5271#	5290#	5309#									
MSDATA	1867#	2211#	2251#	3191#	3197#										
MSDECR	2029#	2211#	2310#	2327#	4442#	4455#	4474#	4493#	4513#	4532#	4550#	4570#	4584#	4601#	4619#
	4699#	4725#	4742#	4761#	4780#	4814#	4839#	4873#	4877#	4904#	4937#	4973#	5019#	5073#	5103#
	5107#	5138#	5142#	5172#	5176#	5207#	5211#	5312#	5383#	5444#	5495#	5499#	5545#	5549#	5592#
	5596#	5656#	5661#	5704#	5709#	5727#	5728#	5788#	5789#	5921#	5970#	6016#	6076#	6166#	6223#
	6303#	6338#	6418#	6501#	6564#	6647#	6679#	6825#	6926#	6978#	7046#	7078#	7132#	7182#	7253#

MSDEFA	7287#	7316#	7337#												
MSENDE	2170#	2211#	2310#	7283#	7284#	7285#									
	2074#	2211#	2310#	2327#	4442#	4455#	4474#	4493#	4513#	4532#	4550#	4570#	4584#	4601#	4699#
	4725#	4742#	4761#	4780#	4814#	4839#	4873#	4877#	4904#	4937#	4973#	5019#	5073#	5103#	5107#
	5138#	5142#	5172#	5176#	5207#	5211#	5312#	5383#	5444#	5495#	5499#	5545#	5549#	5592#	5596#
	5656#	5661#	5704#	5709#	5727#	5728#	5788#	5789#	5921#	5970#	6016#	6076#	6166#	6223#	6303#
	6338#	6418#	6501#	6564#	6647#	6679#	6825#	6926#	6978#	7046#	7078#	7132#	7182#	7253#	7287#
	7316#	7337#													
MSERRI	1649#	2211#	3492#	3498#	3508#	3514#	3621#	3627#	3785#	3817#	3823#	3833#	3839#	3874#	3880#
	3914#	3920#	3928#	3934#	4080#	4085#	4104#	4110#	4119#	4125#	4135#	4141#	4150#	4156#	4289#
	4295#	4811#	4837#	4871#	4902#	4935#	4971#	5013#	5065#	5101#	5136#	5170#	5205#	5269#	5288#
	5307#	5340#	5354#	5366#	5377#	5426#	5436#	5483#	5493#	5534#	5543#	5581#	5590#	5644#	5654#
	5702#	5725#	5772#	5782#	5887#	5910#	6072#	6115#	6124#	6137#	6156#	6164#	6388#	6403#	6486#
	6551#	6632#	6676#	6780#											
MSESCA	2006#	2211#	5014#	5066#	5341#	5355#	5367#	5378#	5427#	5437#	5484#	5535#	5582#	5645#	5773#
	5783#	5888#	5911#												
MSESCS	2010#	2211#	5014#	5066#	5341#	5355#	5367#	5378#	5427#	5437#	5484#	5535#	5582#	5645#	5773#
	5783#	5888#	5911#												
MSEXCP	2101#	2211#	7282#	7283#	7284#	7285#									
MSEXIT	2014#	2211#													
MSEXSE	2022#	2211#													
MSEXTJ	2018#	2211#													
MSGEN	2038#	2211#	2217#	2251#	2274#	2296#	2310#	2324#	2327#	3191#	3197#	4440#	4442#	4446#	4455#
	4461#	4474#	4480#	4493#	4499#	4513#	4519#	4532#	4538#	4550#	4556#	4570#	4576#	4584#	4599#
	4601#	4615#	4633#	4699#	4713#	4725#	4739#	4742#	4756#	4761#	4779#	4780#	4801#	4814#	4826#
	4839#	4854#	4873#	4877#	4888#	4904#	4915#	4937#	4953#	4973#	4990#	5019#	5034#	5073#	5085#
	5103#	5107#	5119#	5138#	5142#	5154#	5172#	5176#	5188#	5207#	5211#	5238#	5312#	5328#	5383#
	5399#	5444#	5465#	5495#	5499#	5516#	5545#	5549#	5563#	5592#	5596#	5621#	5656#	5661#	5678#
	5687#	5704#	5709#	5713#	5727#	5728#	5746#	5749#	5788#	5789#	5811#	5921#	5941#	5970#	5990#
	6016#	6038#	6076#	6096#	6166#	6181#	6223#	6239#	6303#	6318#	6338#	6354#	6418#	6434#	6501#
	6517#	6564#	6580#	6647#	6660#	6679#	6696#	6825#	6849#	6926#	6944#	6978#	6996#	7046#	7061#
	7078#	7099#	7132#	7151#	7182#	7206#	7253#	7280#	7287#	7313#	7316#	7339#			
MSGENB	1938#	2211#													
MSGETS	2035#	2211#	2310#	2327#	4442#	4455#	4474#	4493#	4513#	4532#	4550#	4570#	4584#	4601#	4619#
	4699#	4725#	4742#	4761#	4780#	4814#	4839#	4873#	4877#	4904#	4937#	4973#	5019#	5073#	5103#
	5107#	5138#	5142#	5172#	5176#	5207#	5211#	5312#	5383#	5444#	5484#	5495#	5499#	5535#	5545#
	5549#	5582#	5592#	5596#	5645#	5656#	5661#	5704#	5709#	5727#	5728#	5788#	5789#	5921#	5970#
	6016#	6076#	6166#	6223#	6303#	6338#	6418#	6501#	6564#	6647#	6679#	6825#	6926#	6978#	7046#
	7078#	7132#	7182#	7253#	7287#	7316#	7337#								
MSGETT	1877#	2211#	5014#	5066#	5341#	5355#	5367#	5378#	5427#	5437#	5484#	5535#	5582#	5645#	5773#
	5783#	5888#	5911#												
MSGNGB	1902#	2211#	2217#	2251#	2274#	2296#	2324#	3191#	3197#	4440#	4446#	4461#	4480#	4499#	4519#
	4538#	4556#	4576#	4599#	4615#	4633#	4713#	4739#	4756#	4779#	7280#	7313#	7339#		
MSGNIN	2049#	2211#	2251#	2274#	2296#	2324#	3191#	3197#	3492#	3498#	3508#	3514#	3621#	3627#	3785#
	3817#	3823#	3833#	3839#	3874#	3880#	3914#	3920#	3928#	3934#	4080#	4085#	4104#	4110#	4119#
	4125#	4135#	4141#	4150#	4156#	4289#	4295#	4441#	4442#	4447#	4448#	4449#	4450#	4451#	4452#
	4453#	4454#	4455#	4462#	4463#	4464#	4465#	4466#	4467#	4468#	4469#	4470#	4471#	4472#	4473#
	4474#	4481#	4482#	4483#	4484#	4485#	4486#	4487#	4488#	4489#	4490#	4491#	4492#	4493#	4500#
	4501#	4502#	4503#	4504#	4505#	4506#	4507#	4508#	4509#	4510#	4511#	4512#	4513#	4520#	4521#
	4522#	4523#	4524#	4525#	4526#	4527#	4528#	4529#	4530#	4531#	4532#	4539#	4540#	4541#	4542#
	4543#	4544#	4545#	4546#	4547#	4548#	4549#	4550#	4557#	4558#	4559#	4560#	4561#	4562#	4563#
	4564#	4565#	4566#	4567#	4568#	4569#	4570#	4577#	4578#	4579#	4580#	4581#	4582#	4583#	4584#
	4601#	4650#	4651#	4653#	4654#	4656#	4657#	4659#	4660#	4676#	4677#	4699#	4715#	4722#	4725#
	4742#	4758#	4760#	4761#	4780#	4803#	4811#	4814#	4837#	4839#	4859#	4871#	4873#	4877#	4902#
	4904#	4920#	4935#	4937#	4971#	4973#	5013#	5014#	5019#	5065#	5066#	5073#	5090#	5101#	5103#
	5107#	5124#	5136#	5138#	5142#	5159#	5170#	5172#	5176#	5194#	5205#	5207#	5211#	5245#	5269#

	5271#	5288#	5290#	5307#	5309#	5312#	5340#	5341#	5354#	5355#	5366#	5367#	5377#	5378#	5383#
	5426#	5427#	5436#	5437#	5444#	5469#	5483#	5484#	5493#	5495#	5499#	5520#	5534#	5535#	5543#
	5545#	5549#	5567#	5581#	5582#	5590#	5592#	5596#	5627#	5644#	5645#	5654#	5656#	5661#	5687#
	5689#	5702#	5704#	5709#	5713#	5725#	5727#	5728#	5749#	5772#	5773#	5782#	5783#	5788#	5789#
	5887#	5888#	5910#	5911#	5921#	5970#	6016#	6072#	6076#	6115#	6124#	6137#	6156#	6164#	6166#
	6223#	6303#	6338#	6388#	6403#	6418#	6486#	6501#	6551#	6564#	6632#	6647#	6676#	6679#	6780#
	6825#	6926#	6978#	7046#	7078#	7132#	7182#	7253#	7280#	7282#	7283#	7284#	7285#	7287#	7313#
	7316#	7339#													
MSGNLS	1913#	2211#	4873#	5103#	5138#	5172#	5207#	5495#	5545#	5592#	5656#	5704#			
MSGNSU	1898#	2211#	5687#	5713#	5749#										
MSGNTA	1890#	2211#	2310#	2327#	4442#	4455#	4474#	4493#	4513#	4532#	4550#	4570#	4584#	4601#	4699#
	4725#	4742#	4761#	4780#	4814#	4839#	4877#	4904#	4937#	4973#	5019#	5073#	5107#	5142#	5176#
	5211#	5312#	5383#	5444#	5499#	5549#	5596#	5661#	5709#	5727#	5728#	5788#	5789#	5921#	5970#
	6016#	6076#	6166#	6223#	6303#	6338#	6418#	6501#	6564#	6647#	6679#	6825#	6926#	6978#	7046#
	7078#	7132#	7182#	7253#	7287#	7316#									
MSGNTE	1894#	2211#	4801#	4826#	4854#	4888#	4915#	4953#	4990#	5034#	5085#	5119#	5154#	5188#	5238#
	5328#	5399#	5465#	5516#	5563#	5621#	5678#	5746#	5811#	5941#	5990#	6038#	6096#	6181#	6239#
	6318#	6354#	6434#	6517#	6580#	6660#	6696#	6849#	6944#	6996#	7061#	7099#	7151#	7206#	
MSHAPT	1739#	2211#	2251#												
MSHNAP	1824#	2211#	2251#												
MSINCR	2026#	2211#	2217#	2296#	2324#	3492#	3498#	3508#	3514#	3621#	3627#	3785#	3817#	3823#	3833#
	3839#	3874#	3880#	3914#	3920#	3928#	3934#	4080#	4085#	4104#	4110#	4119#	4125#	4135#	4141#
	4150#	4156#	4289#	4295#	4440#	4441#	4442#	4446#	4447#	4448#	4449#	4450#	4451#	4452#	4453#
	4454#	4455#	4461#	4462#	4463#	4464#	4465#	4466#	4467#	4468#	4469#	4470#	4471#	4472#	4473#
	4474#	4480#	4481#	4482#	4483#	4484#	4485#	4486#	4487#	4488#	4489#	4490#	4491#	4492#	4493#
	4499#	4500#	4501#	4502#	4503#	4504#	4505#	4506#	4507#	4508#	4509#	4510#	4511#	4512#	4513#
	4514#	4520#	4521#	4522#	4523#	4524#	4525#	4526#	4527#	4528#	4529#	4530#	4531#	4532#	4538#
	4539#	4540#	4541#	4542#	4543#	4544#	4545#	4546#	4547#	4548#	4549#	4550#	4556#	4557#	4558#
	4559#	4560#	4561#	4562#	4563#	4564#	4565#	4566#	4567#	4568#	4569#	4570#	4576#	4577#	4578#
	4579#	4580#	4581#	4582#	4583#	4584#	4599#	4601#	4615#	4633#	4650#	4653#	4656#	4659#	4676#
	4699#	4713#	4715#	4722#	4725#	4739#	4742#	4756#	4758#	4760#	4761#	4779#	4780#	4801#	4803#
	4811#	4814#	4826#	4837#	4839#	4854#	4859#	4871#	4873#	4877#	4888#	4902#	4904#	4915#	4920#
	4935#	4937#	4953#	4971#	4973#	4990#	5013#	5014#	5019#	5034#	5065#	5066#	5073#	5085#	5090#
	5101#	5103#	5107#	5119#	5124#	5136#	5138#	5142#	5154#	5159#	5170#	5172#	5176#	5188#	5194#
	5205#	5207#	5211#	5238#	5245#	5269#	5271#	5288#	5290#	5307#	5309#	5312#	5328#	5340#	5341#
	5354#	5355#	5366#	5367#	5377#	5378#	5383#	5399#	5426#	5427#	5436#	5437#	5444#	5465#	5469#
	5483#	5484#	5493#	5495#	5499#	5516#	5520#	5534#	5535#	5543#	5545#	5549#	5563#	5567#	5581#
	5582#	5590#	5592#	5596#	5621#	5627#	5644#	5645#	5654#	5656#	5661#	5678#	5687#	5689#	5702#
	5704#	5709#	5713#	5725#	5727#	5728#	5746#	5749#	5772#	5773#	5782#	5783#	5788#	5789#	5811#
	5887#	5888#	5910#	5911#	5921#	5941#	5970#	5990#	6016#	6038#	6072#	6076#	6096#	6115#	6124#
	6137#	6156#	6164#	6166#	6181#	6223#	6239#	6303#	6318#	6338#	6354#	6388#	6403#	6418#	6434#
	6486#	6501#	6517#	6551#	6564#	6580#	6632#	6647#	6660#	6676#	6679#	6696#	6780#	6825#	6849#
	6926#	6944#	6978#	6996#	7046#	7061#	7078#	7099#	7132#	7151#	7182#	7206#	7253#	7280#	7313#
MSIOSE	1700#	2211#													
MSLDRO	1942#	2211#	4650#	4653#	4656#	4659#	4676#	4715#	4722#	4803#					
MSMASK	1671#	2211#													
MSMCHI	4#	2211#													
MSMCLO	1624#	2211#													
MSMSK1	1677#	2211#													
MSPOP	1881#	2211#	2310#	2327#	4442#	4455#	4474#	4493#	4513#	4532#	4550#	4570#	4584#	4601#	4619#
	4699#	4725#	4742#	4761#	4780#	4814#	4839#	4873#	4877#	4904#	4937#	4973#	5019#	5073#	5103#
	5107#	5138#	5142#	5172#	5176#	5207#	5211#	5312#	5383#	5444#	5495#	5499#	5545#	5549#	5592#
	5596#	5656#	5661#	5704#	5709#	5727#	5728#	5788#	5789#	5921#	5970#	6016#	6076#	6166#	6223#
	6303#	6338#	6418#	6501#	6564#	6647#	6679#	6825#	6926#	6978#	7046#	7078#	7132#	7182#	7253#
	7287#	7316#	7337#												
MSPRIN	1636#	2211#	4441#	4447#	4448#	4449#	4450#	4451#	4452#	4453#	4454#	4462#	4463#	4464#	4465#

	4466#	4467#	4468#	4469#	4470#	4471#	4472#	4473#	4481#	4482#	4483#	4484#	4485#	4486#	4487#
	4488#	4489#	4490#	4491#	4492#	4500#	4501#	4502#	4503#	4504#	4505#	4506#	4507#	4508#	4509#
	4510#	4511#	4512#	4520#	4521#	4522#	4523#	4524#	4525#	4526#	4527#	4528#	4529#	4530#	4531#
	4539#	4540#	4541#	4542#	4543#	4544#	4545#	4546#	4547#	4548#	4549#	4557#	4558#	4559#	4560#
	4561#	4562#	4563#	4564#	4565#	4566#	4567#	4568#	4569#	4577#	4578#	4579#	4580#	4581#	4582#
	4583#	4760#	5245#	5271#	5290#	5309#									
MSBUSH	1631#	2211#	2217#	2296#	2324#	4440#	4446#	4461#	4480#	4499#	4519#	4538#	4556#	4576#	4599#
	4615#	4633#	4713#	4739#	4756#	4779#	4801#	4826#	4854#	4859#	4888#	4915#	4953#	4990#	5034#
	5085#	5090#	5119#	5124#	5154#	5159#	5188#	5194#	5238#	5328#	5399#	5465#	5469#	5516#	5520#
	5563#	5567#	5621#	5627#	5678#	5687#	5689#	5713#	5746#	5749#	5811#	5941#	5990#	6038#	6096#
	6181#	6239#	6318#	6354#	6434#	6517#	6580#	6660#	6696#	6849#	6944#	6996#	7061#	7099#	7151#
	7206#	7280#	7313#												
MSPUT	1972#	2211#	4441#	4447#	4448#	4449#	4450#	4451#	4452#	4453#	4454#	4462#	4463#	4464#	4465#
	4466#	4467#	4468#	4469#	4470#	4471#	4472#	4473#	4481#	4482#	4483#	4484#	4485#	4486#	4487#
	4488#	4489#	4490#	4491#	4492#	4500#	4501#	4502#	4503#	4504#	4505#	4506#	4507#	4508#	4509#
	4510#	4511#	4512#	4520#	4521#	4522#	4523#	4524#	4525#	4526#	4527#	4528#	4529#	4530#	4531#
	4539#	4540#	4541#	4542#	4543#	4544#	4545#	4546#	4547#	4548#	4549#	4557#	4558#	4559#	4560#
	4561#	4562#	4563#	4564#	4565#	4566#	4567#	4568#	4569#	4577#	4578#	4579#	4580#	4581#	4582#
	4583#	4760#	5245#	5271#	5290#	5309#									
MSPUT1	1981#	2211#	4441#	4447#	4448#	4449#	4450#	4451#	4452#	4453#	4454#	4462#	4463#	4464#	4465#
	4466#	4467#	4468#	4469#	4470#	4471#	4472#	4473#	4481#	4482#	4483#	4484#	4485#	4486#	4487#
	4488#	4489#	4490#	4491#	4492#	4500#	4501#	4502#	4503#	4504#	4505#	4506#	4507#	4508#	4509#
	4510#	4511#	4512#	4520#	4521#	4522#	4523#	4524#	4525#	4526#	4527#	4528#	4529#	4530#	4531#
	4539#	4540#	4541#	4542#	4543#	4544#	4545#	4546#	4547#	4548#	4549#	4557#	4558#	4559#	4560#
	4561#	4562#	4563#	4564#	4565#	4566#	4567#	4568#	4569#	4577#	4578#	4579#	4580#	4581#	4582#
	4583#	4760#	5245#	5271#	5290#	5309#									
MSRADI	2077#	2211#	7282#	7283#	7284#	7285#									
MSRBRO	1952#	2211#													
MSRNRO	1962#	2211#	4676#												
MSSETS	2032#	2211#	2217#	2296#	2324#	4440#	4446#	4461#	4480#	4499#	4519#	4538#	4556#	4576#	4599#
	4615#	4633#	4713#	4739#	4756#	4779#	4801#	4826#	4854#	4859#	4888#	4915#	4953#	4990#	5034#
	5085#	5090#	5119#	5124#	5154#	5159#	5188#	5194#	5238#	5328#	5399#	5465#	5469#	5516#	5520#
	5563#	5567#	5621#	5627#	5678#	5687#	5689#	5713#	5746#	5749#	5811#	5941#	5990#	6038#	6096#
	6181#	6239#	6318#	6354#	6434#	6517#	6580#	6660#	6696#	6849#	6944#	6996#	7061#	7099#	7151#
	7206#	7280#	7313#												
MSSTAR	1733#	2211#													
MSVC	1933#	2211#	3492	3498	3508	3514	3621	3627	3785	3817	3823	3833	3839	3874	3880
	3914	3920	3928	3934	4080	4085	4104	4110	4119	4125	4135	4141	4150	4156	4289
	4295	4441#	4442#	4447#	4448#	4449#	4450#	4451#	4452#	4453#	4454#	4455#	4462#	4463#	4464#
	4465#	4466#	4467#	4468#	4469#	4470#	4471#	4472#	4473#	4474#	4481#	4482#	4483#	4484#	4485#
	4486#	4487#	4488#	4489#	4490#	4491#	4492#	4493#	4500#	4501#	4502#	4503#	4504#	4505#	4506#
	4507#	4508#	4509#	4510#	4511#	4512#	4513#	4520#	4521#	4522#	4523#	4524#	4525#	4526#	4527#
	4528#	4529#	4530#	4531#	4532#	4539#	4540#	4541#	4542#	4543#	4544#	4545#	4546#	4547#	4548#
	4549#	4550#	4557#	4558#	4559#	4560#	4561#	4562#	4563#	4564#	4565#	4566#	4567#	4568#	4569#
	4570#	4577#	4578#	4579#	4580#	4581#	4582#	4583#	4584#	4601#	4650#	4653#	4656#	4659#	4676#
	4699#	4715#	4722#	4725#	4742#	4758#	4760#	4761#	4780#	4803#	4811	4814#	4837	4839#	4859#
	4871	4873#	4877#	4902	4904#	4920#	4935	4937#	4971	4973#	5013	5014#	5019#	5065	5066#
	5073#	5090#	5101	5103#	5107#	5124#	5136	5138#	5142#	5159#	5170	5172#	5176#	5194#	5205
	5207#	5211#	5245#	5269	5271#	5288	5290#	5307	5309#	5312#	5340	5341#	5354	5355#	5366
	5367#	5377	5378#	5383#	5426	5427#	5436	5437#	5444#	5469#	5483	5484#	5493	5495#	5499#
	5520#	5534	5535#	5543	5545#	5549#	5567#	5581	5582#	5590	5592#	5596#	5627#	5644	5645#
	5654	5656#	5661#	5687#	5689#	5702	5704#	5709#	5713#	5725	5727#	5728#	5749#	5772	5773#
	5782	5783#	5788#	5789#	5887	5888#	5910	5911#	5921#	5970#	6016#	6072	6076#	6115	6124
	6137	6156	6164	6166#	6223#	6303#	6338#	6388	6403	6418#	6486	6501#	6551	6564#	6632
	6647#	6676	6679#	6780	6825#	6926#	6978#	7046#	7078#	7132#	7182#	7253#			
MSLAB	1929#	2211#	3492#	3498#	3508#	3514#	3621#	3627#	3785#	3817#	3823#	3833#	3839#	3874#	3880#

REDEF	1403#	2211#	4650	4653	4656	4659
RFLAGS	1408#	2211#				
SETPRI	1413#	2211#	4715	4803		
SETVEC	1418#	2211#				
SLASH	1424#	2211#				
STARS	1438#	2211#				
SVC	1452#	2210#	2211			
XFER	1612#	2211#				
XFERF	1616#	2211#				
XFERT	1620#	2211#				

. ABS. 036234 000

ERRORS DETECTED: 0

CZDMRC.BIN,CZDMRC.LST/CRF/NL:TOC=SVC34R.MLB,CZDMRC.P11
RUN-TIME: 123 150 14 SECONDS
RUN-TIME RATIO: 541/289-1.8
CORE USED: 17K (33 PAGES)