TEXAS INSTRUMENTS

Improving Man's Effectiveness Through Electronics

Model 990 Computer
Product Documentation Package

990 Diagnostics
(Object)

No. 2261796-0001 Rev. R

Digital Systems Division

TABLE OF CONTENTS

I. Lists of Parts

1. Diagnostics

NOTES: (1) All DOCS and loadable tests are on disk 2250549-0001. The table below shows whether the tests are Relative Record (REL) or Sequential (SEQ) on disk and DSDD diskettes.

(2) For operating instructions, refer first to the Program Description (if it is contained in this volume) and then to the Diagnostic Handbook (945400-9701).

(3) Revision levels of all media are given with the media index in Section II.

Diagnostic		Contained On Media			Disk File Type
Name	Object P/N,Rev	Cassette P/N	SSSD P/N	DSDD P/N	
ACUTST	2250109-1006*A	2250511-0001	2250548-0002	2250549-0002	REL
ADCHK	2250110-1006**	2250518-0001	2250548-0003	2250549-0002	REL
AU04	937754-1006*F	2250503-0001	2250548-0001	2250549-0003	SEQ
AU04TEST	937983-1007*E	2250503-0001	2250548-0001	2250549-0003	SEQ
AU05	2267307-1006**	2250504-0001	2250548-0004	2250549-0002	REL
AU10	937753-1006*K	2250502-0001	2250548-0001	2250549-0002	REL
AU12P7	2268243-1006*B	2250525-0001	2250548-0009	2250549-0002	REL
AU12P8	2268244-1006*B	2250525-0001	2250548-0009	2250549-0002	REL
AU12P9	2268245-1006*B	2250525-0002	2250548-0009	2250549-0002	REL
AU12P10	2268246-1006*B	2250525-0002	2250548-0009	2250549-0002	REL
AU12P11	2268247-1006*B	2250525-0003	2250548-0010	2250549-0002	REL
COMM05	2267316-1006**	2250504-0001	2250548-0004	2250549-0002	REL
CRCOMM	2250100-1006*A	2250513-0001	2250548-0004	2250549-0002	REL
CRDRDR	2250140-1003**	2250509-0001	2250548-0001	2250549-0002	REL
CRT911	2250101-1003**	2250510-0001	2250548-0001	2250549-0002	REL
CRT913	2250128-1006**	2250510-0001	2250548-0001	2250549-0002	REL
CRUEXP	2250111-1006*A	2250514-0001	2250548-0002	2250549-0002	REL
DACHK	2250112-1006**	2250518-0001	2250548-0003	2250549-0002	REL
DDFLOP	2250139-1006*C	2250522-0001	2250548-0003	2250549-0002	REL
DS10PD	2250113-1006*B	2250524-0001	2250548-0003	2250549-0002	REL
DSKM3X	2250102-1006*B	2250523-0001	2250548-0003	2250549-0002	REL
DSKTRI	2250114-1006*A	2250524-0001	2250548-0003	2250549-0002	REL
EMU900	2250229-1006*C	2250517-0001	2250548-0002	2250549-0002	REL
EMU940	2250228-1006*C	2250517-0001	2250548-0002	2250549-0002	REL
EMUTST	2250132-1006*A	2250516-0001	2250548-0002	2250549-0002	REL
EROMBT	2250133-1003**	2250515-0001	2250548-0002	2250549-0002	REL
EXTACU	2250158-1006**	2250511-0001	2250548-0002	2250549-0002	REL
FIVMOD	2250134-1003**	2250520-0001	2250548-0003	2250549-0002	REL
FLPDSK	2250105-1006*C	2250521-0001	2250548-0003	2250549-0002	REL
FLPTST	2250117-1006**	2250508-0001	2250548-0001	2250549-0002	REL
HSTSLV	2267305-1006**	2250504-0001	2250548-0004	2250549-0002	REL
INPMOD	2250120-1003**	2250519-0001	2250548-0003	2250549-0002	REL
IO16	2250121-1003**	2250514-0001	2250548-0002	2250549-0002	REL
LOCLIN	2250122-1003**	2250512-0001	2250548-0002	2250549-0002	REL
LP810	2250106-1006*B	2250508-0001	2250548-0001	2250549-0002	REL

Diagnostic		Contained On Media			Disk File Type
Name	Object P/N,Rev	Cassette P/N	SSSD P/N	DSDD P/N	
LPTEST	2250123-1006*A	2250507-0001	2250548-0001	2250549-0002	REL
MAP12	2268218-1006*A	2250525-0003	2250548-0010	2250549-0002	REL
MAPTST	2250561-1006**	2250502-0001	2250548-0001	2250549-0002	REL
MEMPRT	2250124-1003**	2250506-0001	2250548-0001	2250549-0003	SEQ
OUTMOD	2250125-1003**	2250519-0001	2250548-0003	2250549-0002	REL
PROMPG	2250118-1006*B	2250515-0001	2250548-0002	2250549-0002	REL
RAM04	2250107-1006*A	2250506-0001	2250548-0001	2250549-0003	SEQ
RAM05	2267303-1006*C	2250504-0001	2250548-0004	2250549-0002	REL
RAM10	2250119-1006*A	2250505-0001	2250548-0001	2250549-0002	REL
RMTEIA	2250242-1006**	2250512-0001	2250548-0004	2250549-0002	REL
RMTFLP	2250108-1003**	2250521-0001	2250548-0003	2250549-0002	REL
TAPTST	2250126-1006*A	2250523-0001	2250548-0003	2250549-0002	REL
TILCOU	2250129-1003**	2250522-0001	2250548-0003	2250549-0002	REL
TRACE	2250130-1006*A	2250516-0001	2250548-0002	2250549-0002	REL
TST733	2250250-1006*A	2250509-0001	2250548-0001	2250549-0002	REL
TST820	2250255-1006*A	2250513-0001	2250548-0004	2250549-0002	REL
TTYEIA	2250131-1006*B	2250507-0001	2250548-0001	2250549-0002	REL

2. DOCS

NOTES: (1) The first character of the name indicates the loadable media of the DOCS loader:
U = Unit record media (cassette, cards)
F = SSSD Diskette
D = Disk or DSDD Diskette

DOCS		Contained On Media			
Name	Object P/N	Cassette	SSSD	DSDD	File
DBCDOCS	2250291-1004*A	N/A	N/A	2250549-2	REL
DMNDOCS	2250163-1004*D	N/A	N/A	2250549-2	REL
DMXDOCS	2250162-1004*C	N/A	N/A	2250549-2,3	REL
FBCDOCS	2250291-1003*A	N/A	2250548-7	2250549-3	SEQ
FFPDOCS	2250164-1003*B	N/A	2250548-7	2250549-3	SEQ
FMNDOCS	2250163-1003*D	N/A	2250548-1,7	2250549-3	SEQ
FMXDOCS	2250162-1003*C	N/A	2250548-ALL	2250549-3	SEQ
UBCDOCS	2250291-1002*A	2250501-1	2250548-7	2250549-2	REL
UFPDOCS	2250164-1002*B	2250501-1	2250548-7	2250549-2	REL
UMNDOCS	2250163-1002*D	2250501-1	2250548-7	2250549-2	REL
UMXDOCS	2250162-1002*C	2250501-1	2250548-7	2250549-2	REL

II. Package Inventory for 990 Diagnostics, Object (937782AC)

1. DOCUMENTATION (for all media):

		Part Number	Qty
A.	Diagnostic Handbook (15 January 1979)	945400-9701	1
B.	Product Documentation Package(this document)	2261796-0001	1

2. SOFTWARE MEDIA:

Package Part Number	Media Type
937782-0001	Cassette
937782-0002	Cards
937782-0003	Single-Sided/Single-Density Diskette
937782-0004	DS31
937782-0005	DS25
937782-0006	DS50
937782-0010	DS10
937782-0021	Double-Sided/Double-Density Diskette
937782-0023	DS200

CASSETTE MEDIA (937782-0001AC) includes the following:

1.	UMXDOCS,UBCDOCS,UMNDOCS,UFPDOCS	2250501-0001*E
2.	AU10,MAPTST	2250502-0001*C
3.	AU04TEST,AU04	2250503-0001*A
4.	AU05,COMM05,RAM04,HSTSLV	2250504-0001*B
5.	RAM10	2250505-0001*A
6.	RAM04,MEMPRT	2250506-0001*A
7.	LPTEST,TTYEIA	2250507-0001*C
8.	LP810,FLPTST	2250508-0001*B
9.	TST733,CRDRDR	2250509-0001*B
10.	CRT911,CRT913	2250510-0001**
11.	ACUTST,EXTACU	2250511-0001**
12.	LOCLIN,RMTEIA	2250512-0001*A
13.	CRCOMM,TST820	2250513-0001*C
14.	IO16,CRUEXP	2250514-0001*A
15.	PROMPG,EROMBT	2250515-0001*B
16.	EMUTST,TRACE	2250516-0001*A
17.	EMU900,EMU940	2250517-0001*D
18.	ADCHK,DACHK	2250518-0001**
19.	INPMOD,OUTMOD	2250519-0001**
20.	FIVMOD	2250520-0001**
21.	FLPDSK,RMTFLP	2250521-0001*B
22.	DDFLOP,TILCOU	2250522-0001*C
23.	DSKM3X,TAPTST	2250523-0001*B
24.	DSKTRI,DS10PD	2250524-0001*B
25.	AU12P7,AU12P8	2250525-0001*D
26.	AU12P9,AU12P10	2250525-0002*D
27.	AU12P11,MAP12	2250525-0003*D
28.	Card Reader Test Deck	2250136-3201**

CARD MEDIA (937782-0002AC) includes the following:

1.	UMXDOCS	2250162-1002*C	2.	UMNDOCS	2250163-1002*D
3.	UFPDOCS	2250164-1002*B	4.	UBCDOCS	2250291-1002*A
5.	ACUTST	2250109-1006*A	6.	ADCHK	2250110-1006**
7.	AU04	937754-1006*F	8.	AU04TEST	937983-1007*E
9.	AU05	2267307-1006**	10.	AU10	937753-1006*K
11.	AU12P7	2268243-1006*B	12.	AU12P8	2268244-1006*B
13.	AU12P9	2268245-1006*B	14.	AU12P10	2268246-1006*B
15.	AU12P11	2268247-1006*B	16.	COMM05	2267316-1006**
17.	CRCOMM	2250100-1006*A	18.	CRDRDR	2250140-1003**
19.	Test Deck	2250136-3201**	20.	CRT911	2250101-1003**
21.	CRT913	2250128-1006**	22.	CRUEXP	2250111-1006*A
23.	DACHK	2250112-1006**	24.	DDFLOP	2250139-1006*C
25.	DS10PD	2250113-1006*B	26.	DSKM3X	2250102-1006*B
27.	DSKTRI	2250114-1006*A	28.	EMU900	2250229-1006*C
29.	EMU940	2250228-1006*C	30.	EMUTST	2250132-1006*A
31.	EROMBT	2250133-1003**	32.	EXTACU	2250158-1006**
33.	FIVMOD	2250134-1003**	34.	FLPDSK	2250105-1006*C
35.	FLPTST	2250117-1006**	36.	HSTSLV	2267305-1006**
37.	INPMOD	2250120-1003**	38.	I016	2250121-1003**
39.	LOCLIN	2250122-1003**	40.	LP810	2250106-1006*B
41.	LPTEST	2250123-1006*A	42.	MAP12	2268218-1006*A
43.	MAPTST	2250561-1006**	44.	MEMPRT	2250124-1003**
45.	OUTMOD	2250125-1003**	46.	PROMPG	2250118-1006*B
47.	RAM04	2250107-1006*A	48.	RAM05	2267303-1006*C
49.	RAM10	2250119-1006*A	50.	RMTEIA	2250242-1006**
51.	RMTFLP	2250108-1003**	52.	TAPTST	2250126-1006*A
53.	TILCOU	2250129-1003**	54.	TRACE	2250130-1006*A
55.	TST733	2250250-1006*A	56.	TST820	2250255-1006*A
57.	TTYEIA	2250131-1006*B			

SSSD DISKETTE MEDIA (937782-0003AC) includes the following:

1. Tests Diskette 1, 2250548-0001*L

   Menu:

00) AU10	01) FMXDOCS	02) FMNDOCS	03) FFCOPY
04) AU04	05) AU04TST	06) MAPTSTP	07) RAM10
08) RAM04	09) MEMPRT	0A) LPTEST	0B) TTYEIA
0C) TST733	0D) CRDRDR	0E) CRT911	0F) CRT913
10) FLPTST	11) LP810		

2. Tests Diskette 2, 2250548-0002*L

   Menu:

00) FMXDOCS	01) ACUTST	02) EXTACU	03) LOCLIN
04) IO16	05) CRUEXP	06) PROMPG	07) EROMBT
08) EMUTST	09) TRACE	0A) EMU900	0B) EMU940

3. Tests Diskette 3, 2250548-0003*L

   Menu:

00) ADCHK	01) DACHK	02) INPMOD	03) OUTMOD
04) FIVMOD	05) FLPDSK	06) RMTFLP	07) DDFLOP
08) TILCOU	09) DSKM3X	0A) TAPTST	0B) DSKTRI
0C) DS10PD	0D) FMXDOCS		

4. Tests Diskette 4, 2250548-0004*L

   MENU:

00) FMXDOCS	01) RMTEIA	02) TST820	03) CRCOMM
04) AU05	05) RAM05	06) HSTSLV	07) COMM05

5. Tests Diskette 6, 2250548-0009*L

   Menu:

00) FMXDOCS	01) AU12P7	02) AU12P8	03) AU12P9
04) AU12P10			

6. Tests Diskette 7, 2250548-0010*L

   Menu:

00) FMXDOCS	01) AU12P11	02) MAP12

7. Systems Diskette, 2250548-0007*L

   Menu:

00) FMXDOCS	01) FFPDOCS	02) FMNDOCS	03) FFCOPY
04) UMXDOCS	05) UMNDOCS	06) UFPDOCS	07) FBCDOCS
08) UBCDOCS			

8. Card Reader Test Deck, 2250136-3201**

7. Floppy Loader Cassette, 937921-0001*A

DISK MEDIA (937782-0004,-0005,-0006,-0010,-0023AC)

includes the following:

1. Disk, 2250549-0001*L (NOTE: Only those tests which will run under a DS990 System are listed in the Menu; therefore, all 990/4 tests are omitted from the Menu.)

   DOCSVOLM Menu:

00) FIVMOD	01) UFPDOCS	02) ACUTST	03) AU12P8
04) AU12P11	05) UBCDOCS	06) EMU940	07) RMTEIA
08) EMU900	09) HSTSLV	0A) ADCHK	0B) DMXDOCS
0C) TAPTST	0D) AU05	0E) DSKM3X	0F) TST820
10) TST733	11) AU12P9	12) DBCDOCS	13) AU10
14) CRDRDR	15) CRCOMM	16) OUTMOD	17) CRUEXP
18) AU12P10	19) DMNDOCS	1A) CRT911	1B) EXTACU
1C) TRACE	1D) TTYEIA	1E) INPMOD	1F) DS10PD
20) LP810	21) EMUTST	22) DDFLOP	23) IO16
24) LPTEST	25) RMTFLP	26) DACHK	27) LOCLIN
28) CRT913	29) EROMBT	2A) COMM05	2B) PROMPG
2C) TILCOU	2D) RAM05	2E) MAPTST	2F) AU12P7
30) FLPDSK	31) RAM10	32) UMNDOCS	33) FLPTST
34) UMXDOCS	35) MAP12	36) DSKTRI	

2. Card Reader Test Deck, 2250136-3201**

DSDD MEDIA (937782-0021AC) includes the following:

1. DSDD Diskette 1, 2250549-0002*L

   DOCSVOL1 Menu:

00) FIVMOD	01) UFPDOCS	02) ACUTST	03) AU12P8
04) AU12P11	05) UBCDOCS	06) EMU940	07) RMTEIA
08) EMU900	09) HSTSLV	0A) ADCHK	0B) DMXDOCS
0C) TAPTST	0D) AU05	0E) DSKM3X	0F) TST820
10) TST733	11) AU12P9	12) DBCDOCS	13) AU10
14) CRDRDR	15) CRCOMM	16) OUTMOD	17) CRUEXP
18) AU12P10	19) DMNDOCS	1A) CRT911	1B) EXTACU
1C) TRACE	1D) TTYEIA	1E) INPMOD	1F) DS10PD
20) LP810	21) LPTEST	22) EMUTST	23) DDFLOP
24) IO16	25) RMTFLP	26) DACHK	27) LOCLIN
28) CRT913	29) EROMBT	2A) COMM05	2B) PROMPG
2C) TILCOU	2D) RAM05	2E) MAPTST	2F) AU12P7
30) FLPDSK	31) RAM10	32) UMNDOCS	33) FLPTST
34) UMXDOCS	35) MAP12	36) DSKTRI	

2. DSDD Diskette 2, 2250549-0003*L (contains all non-Menu parts from DOCSVOLM):

   DOCSVOL2 Menu:

   00) DMXDOCS

3. Card Reader Test Deck, 2250136-3201**

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																																		
SHEET																																		
REV STATUS OF SHEETS	REV																																	
	SHEET																																	

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± 010
2 PLACE DECIMAL ± 02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL STD 100

DWN	DATE
CHK	
ENGR	
QA	
APVD	
CONTR NO	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

PD, BATCH COMMAND STREAM (BC) DOCS-990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	2250248-9903

SCALE	REV. F	SHEET 1 of 42

TABLE OF CONTENTS

1.0 SCOPE

This document describes the Batch Command Stream (BCS) version of DOCS (BCDOCS) and its capabilities. The BCS feature is accomplished by implementing several new verbs that are associated with the building and execution of a table of DOCS commands.

The DOCS Diagnostic Control Language (DCL) consists of all the verbs supported by DOCS and the verbs supported by the Test Modules and their associated input data. Each verb is perceived in the same light that SCI commands under DX10 are viewed. Each verb corresponds to a processor (either in DOCS or in the Test Module) and the associated input data required by the processor to accomplish a specific task.

Once one has perceived the diagnostic verbs in this fashion, it is a short step to consider the possibility of prepackaging a set of verbs and their associated input data and then executing this set as though it were one new verb. This is what is referred to as the Batch Command Stream capability.

2.0 REFERENCES

Before continuing with this document, it is recommended that the reader become familiar with the information contained in the DOCS Program Description (P/N 02250248-9901).

3.0 EQUIPMENT AND SOFTWARE REQUIREMENTS

This section describes the minimum equipment requirements and the required modules that must be linked to form BCDOCS.

3.1 Equipment Requirements

Listed below is the minimum required equipment for BCDOCS.

a. 990 with 16K words

b. An appropiate interactive device

c. An appropriate loading device

3.2 Software Requirements

Listed below are the required object modules that must be linked to create a fully linked object (FLO) module of BCDOCS.

ROOT	02250171-1001
INITSTD	02250570-1001
SCAN	02250177-1001
MNVERB	02250176-1001
MXVERB	02250175-1001
IOSERV	02250182-1001
SBDOCS	02250183-1001
DSRINF	02250185-1001

DSR733	02250187-1001
DSR913	02250188-1001
DSR911	02250189-1001
DSRLP	02250191-1001
BCSEDT	02250287-1001
BCSUBR	02250288-1001
BCSVRB	02250289-1001
BCSTXT	02250290-1001
BCIDENT	02250567-1001
<one of the loaders listed below>	
DUMYLDR	02250170-1001
DISCLDR	02250169-1001
FLPYLDR	02250168-1001
UNITLDR	02250167-1001

4.0 SOFTWARE DESCRIPTION

BCDOCS consists of Standard DOCS with four additional modules which enable DOCS to build and execute a batch stream of DOCS commands. Table 1 gives a summary of the Standard DOCS and BCDOCS verbs and their associated formats.

4.1 Batch Command Stream

When the batch stream is being created or executed, it exists as a table within BCDOCS and has the following

structure:

<BCS table> = [<char string input>][<input list>]<table term><CR>

where:

<char string input> = [<char string>]<CR>

<input list> = <char string input>[<input list>]

<char string> = any legal ASCII character string input by SSIRP or inserted by text editing commands (except "@",<CR>,".EL",".BL",".LL",".IL",".DL",".RL", ".CL", or ".PL"). <char string> can also be <null>.

<CR> = >0D which is generated by the "NEW LINE" or "RETURN" keys when in the 'Build List' mode.

<table term> = @

<insert> = >F1 (Text editing insert character)

<delete> = >F2 (Text editing delete character)

Each <char string input> represents a single input that the operator would normally supply whether it be a verb or an associated data item. Each <char string> has a <line #> associated with it where 1 would correspond to the first <char string> of the table.

The batch stream data can be placed in the DOCS table using

three methods. First, DOCS can be placed in the "Build List" mode by executing the Build List verb (.BL). Any operator input (except for some of the control and text editing verbs) following this will be placed in the BCS table until the "Build List" mode is terminated by the input of an "@", CMD ,or HELP key. The second method involves executing the Load List verb (.LL) which brings a file in from the DOCS load media and places it in the BCS table. The Load and Go (.LG) verb may also be used to load a file from the media. This verb differs from the verb .LL in that it begins execution of the list automatically. The third method consists of using the text editing commands (.IL, .DL, .RL, and .CL) in "Normal" or "Build List" modes to modify an existing or new table.

When using the verbs .LL and .LG, the data file must be created offline by using the DX10 Text Editor and Assembler. The Text Editor can be used to create an assembly language program consisting entirely of 'TEXT' statements. The 'TEXT' statements would contain the character strings which make up the <BCS table> (an example is shown in Table 2). After text editing the program, it would be assembled and placed on the DOCS load media in the correct file format. Once this is done, the .LL verb or the .LG verb can be used to load the batch stream into the <BCS table>.

Once the BCS table contains data, it can be executed by entering the .EL verb. This places DOCS in the "Execute List" mode, and DOCS takes all of its inputs from the BCS table (except when the .OI verb is encountered).

Table 1

Standard DOCS Verbs

.IS	Initialize System
.IP	Initialize Parameters
.PV	Print Available Verbs
.DM-<add>-<# wds>	Dump Memory
.MM-<add>-<data>	Modify Memory
.MV-<from>-<to>-<# wds>	Move Memory
.MI-<add>-<# wds>-<data>	Memory Initialize
.MS-<cond>-<add>-<# wds>-<data>	Memory Search
.GO-<add>	Go To Memory Location
.CW-<base>-<# bits>-<data>	Enter Data At CRU Word
.LC-<# wds>-<base>-<data>-<timeout>	Loop on CRU Word
.AD-<data1>-<data2>	Add Hexidecimal Data
.LD-<diag #>	Load Diagnostic

BCS Verbs

.BL	Build List
.EL-<line #>	Execute List
.LL-<DOCS loading parameters>	Load List
.LG-<DOCS loading parameters>	Load and Go
.IL-<line #>	Insert Line
.DL-<line #>	Delete Line
.RL-<line #>	Replace Line
.CL	Compress List
.PL-<line #>-<# lines>-<output dev>	Print Line
or	
-O-<output dev>	Print List
.AT	Automatic Terminate
.LB-<label value>	Establish Label
.GT-<label value>	Unconditional Transfer
.IF-<flag #>-<label value>	Conditional Transfer
.CF-<flag #>	Clear Flag
.SF-<flag #>	Set Flag
.MF-<flag #>-<flag value>	Modify Flag
.OI	Operator Input
.OC-<text>	Operator Comment
.OD-<label value>-<text>-<rspn>	Operator Decision

4.2 BCS Verbs

The verbs described in this document support the Batch Command Stream capability and can be separated into three categories: mode control, text editing, and BCS control. In addition to these verbs, there exists a set of 32 flags (numbered 0 through >1F) which are used by the BCS control verbs. Flag 0 is designated as the System Error Flag and is set any time an error message is output.

4.2.1 Mode Control Verbs

These verbs control the "Build List" and "Execute List" modes and provide a means of loading BCS modules from the DOCS load media.

4.2.1.1 Build List - .BL

This initializes <BCS table> and places DOCS in "Build List" mode. It is an illegal input when DOCS is in "Execute List" mode and will cause a return to "Normal" mode. While in "Build List" mode all inputs are stored in <BCS table>. All verbs entered are executed as they are entered and entering the "@" will terminate the list and "Build List" mode.

4.2.1.2 Execute List - .EL-<line #>

This verb places DOCS in "Execute List" mode and starts

batch stream execution at <line #>. If <line #> is not specified, execution resumes at the default <line #>. When 0 is specified or is the default, execution begins at the beginning of the list. .EL is an illegal input when DOCS is in "Build List" mode and will cause a return to "Normal" mode and 'VERB?'. Execution of this verb causes all of the BCS flags to be cleared.

4.2.1.3 Load List - .LL-<DOCS loading parameters>

This will cause DOCS to display the menu, load the specified file into the <BCS table>, and initialize table pointers. The list is loaded from the media into memory at the location specified by the operator and is then relocated into <BCS table>. It is an illegal input in "Build List" mode. It is also illegal in "Execute List" mode and will cause a return to "Normal" mode.

4.2.1.4 CRASH and Exit to Normal Mode - '@'

The input of the character '@' (whether from the operator or from the <BCS table>) will cause DOCS to exit into the "Normal" mode. If DOCS had previously been in the "Execute List" mode, the execution pointer is reset to the beginning of the <BCS table>.

4.2.1.5 Load and Go - .LG-(DOCS loading parameters>

This verb is the equivalent of doing a '.LL' followed by a '.EL' starting execution at the beginning of the list. This verb is a legal input in both "Build List" mode and in "Execute List" mode. The verb and its parameters can be the last entries in a list currently loaded in the <BCS table> and when the load and go is executed, a new list can be loaded into the <BCS table>. This provides the capability of chaining the execution of various BCS lists. Execution of this verb does not change the status of any of the BCS flags.

4.2.2 Text Editing Verbs

These verbs are legal only in the "Build List" and "Normal" modes and can be used to modify the <BCS table>. While modifications are being made, line numbers associated with the unmodified table entries remain unchanged. Line numbers are reassigned only when the Compress List verb is executed.

4.2.2.1 Insert Line - .IL-<line #>-<char string>

This inserts the <char string><CR> prior to the specified <line #>. Since the <table term> has a line number, insertions can be made throughout the table.

4.2.2.2 Delete Line - .DL-<line #>

Any <char string input> that has an associated <line #> can be deleted from the <BCS table> with the exception of the <table term>.

4.2.2.3 Replace Line - .RL-<line #>-<char string>

This verb will replace any <char string input> that has an associated <line #> with the specified <char string><CR>.

4.2.2.4 Compress List - .CL

This is used to delete all text editing information from the <BCS table>. This involves eliminating any evidence of deleted or inserted lines and in effect reassigns all of the line numbers. The reassignment occurs since line numbers are determined by the relative position of a <char string input> within the <BCS table>.

4.2.2.5 Print List - .PL-<line #>-<# of lines>-<output dev>

or - .PL-0-<output dev>

Print List can be used to print one or more <char string>'s from the <BCS table> to the <output dev>. The second format of the verb can be used to print the entire table. The format of each printed line consists of the line number followed by its corresponding <char string>. All deleted lines are simply denoted by an '*'. All inserted lines appear as the <char string>with no lione number printed. If

<char string> is a null character string, '<CR>' will be printed.

4.2.3 BCS Control Verbs

These verbs are used to control the flow through the Batch Command Stream during execution. They are also used to defer inputting to the operator and to output messages at the BCS level.

4.2.3.1 Establish Label - .LB-<label value>

This verb establishes a label at the position it appears within the BCS table and can be identified by the routines servicing the Transfer type verbs. This allows branching and looping within the Batch Command Stream. Care must be taken not to define duplicate labels since only the first one is detected by the search routines. Care must also be taken to place the label preceeding a DCL verb and not in front of a verb input. .LB is an illegal input during "Normal" mode.

4.2.3.2 Unconditional Transfer - .GT-<label value>

The .GT verb causes BCS execution to resume at the DCL command following <label value>. This verb is an illegal input during "Normal" mode. If the label does not exist, DOCS exits to "Normal" mode from "Execute List" mode.

4.2.3.3 Conditional Transfer - .IF-<flag #>-<label value>

This verb causes BCS execution to resume at the DCL command following <label value> if <flag #> is set. If either the <flag #> or <label value> are nonexistent, DOCS exits to "Normal" mode from "Execute List" mode. .IF is an illegal input in "Normal" mode.

4.2.3.4 Flag Verbs - .CF-<flag #>

- .SF-<flag #>

- .MF-<flag #>-<flag value>

These verbs clear, set, or modify a BCS flag specified by <flag #>. In all cases, DOCS will exit to "Normal" mode if an illegal flag number is specified.

4.2.3.5 Operator Input - .OI

The Operator Input verb provides a means of altering the normal input mechanism for the "Execute List" mode. Instead of obtaining the data from the BCS table, the operator is prompted for the input. In "Build List" mode, ".OI" is placed in <BCS table> and the operator is prompted for the normal input (which is not placed in the table). In the "Execute List" mode, the .OI verb will cause DOCS to prompt the operator for the input. In this way, any data that cannot be predetermined and placed in the Batch Command

Stream can be supplied by the operator at run time. This can also be used to control the execution path through the BCS table as shown by the example in Table 2.

4.2.3.6 Operator Comment - .OC-<text>

This verb provides a means of outputting text to the operator from the Batch Command Stream level. This verb is an illegal input during "Normal" mode.

4.2.3.7 Operator Decision - .OD-<label value>-<text>-<rspn>

This verb allows the operator to make a decision based on information contained in the output message (<text>). If the operator's response (<rspn>) is 'no' (0), transfer is made to <label value> and execution resumes. If the <rspn> is 'yes'(1), execution resumes on the next statement.

4.2.3.8 Automatic Terminate - .AT

When executed, this verb will cause automatic termination of the "Execute List" mode whenever an error message is output. The System Error Flag (#0) is set and termination takes place whenever execution of the current verb ends. DOCS returns to "Normal" mode and outputs 'VERB ?'. Typically, this verb is placed at the beginning of Batch Command Stream. Execution of this verb causes the BCS flag number '0' to be cleared.

4.4 BCS Example Build Procedure

The example in Table 2 shows how to build a BCS list that will ask if the operator wishes to run the diagnostic LP810. If the answer is NO, then the list will return to "Normal" mode by transferring to the end of the list where <table term> is located. If the answer is YES, then the following sequence of events will take place:

a. The diagnostic will be loaded

b. The 'IT' verb parameters will be answered

c. The diagnostic will be executed (by answering YES to 'EXECUTE EA VERB ? (DEF=1)')

For this example, it will be assumed that the default CRU base and interrupt level will be used. The default CRU base is >0060 and interrupt level is >E. It will also be assumed that the operator wishes to execute the 'EA' verb but does not wish to run the manual intervention test. Floppy diskette is the load media.

To build this list using the "Build List" mode the operator would go through the sequence shown in Table 2. Table 3 gives the internal representation of the list after it is built.

Table 2.

```
VERB ? -.BL<CR>

ENTERING BUILD LIST MODE


VERB ? -.OD<CR>

 LABEL VALUE -BB<CR>
 TEXT - DO YOU WANT LP810 RUN?<CR> (1=YES, 0=NO) -.OI<CR>-<CR>

VERB ? -.LD<CR>

INSERT THE CORRECT DISKETTE  PRESS 'RETURN' -<CR>

(THE MENU IS DISPLAYED HERE)

TYPE: DIAG. # OR FILE NAME ? -LP810<CR>
LOAD BIAS? (DEF=386E) -<CR>
END OF TEST = 6240
LP810   MODEL MC-810 PRINTER TEST
VERSION = *A  11/01/78

ENTER PRINTER CRU BASE                    DEFAULT  =0060  -<CR>
USE THE TTY/EIA BOARD (0) ...             DEFAULT = 00 -<CR>
ENTER CLOCK FREQUENCY (0=3MZ,1=4MZ)       DEFAULT = 00 -<CR>
ENTER PRINTER INTERRUPT LEVEL             DEFAULT = 0E -<CR>
DO YOU WANT TO RUN WITH INTERRUPTS ?      DEFAULT = 01 -<CR>
DOES PRINTER HAVE FULL ASCII ...          DEFAULT = 01 -<CR>
DOES PRINTER HAVE THE VCO OPTION ?        DEFAULT = 01 -<CR>
IS COMPUTER LINE CURRENT 60 HZ ?          DEFAULT = 01 -<CR>
DO YOU WANT TO RUN MANUAL ...             DEFAULT = 01 -0<CR>
ENTER COLUMN WIDTH (0=132,1=80)           DEFAULT = 01 -<CR>
EXECUTE EA VERB? (DEF=1) -1<CR>

TEST 01 DOES NOT PRINT, TEST 01      COMPLETE
TEST 02 COMPLETE
TEST 03 COMPLETE
TEST 04 COMPLETE
TEST 05 COMPLETE
TEST 06 COMPLETE
TEST 07 COMPLETE
TEST 08 COMPLETE
TEST 09 COMPLETE
TEST 0A COMPLETE
TEST 0B COMPLETE
TEST 0C COMPLETE
TEST 0D COMPLETE
TEST 0E COMPLETE
```

```
VERB ? -.LB<CR>

 LABEL VALUE  -BB<CR>

VERB ? - @
TERMINATING BUILD LIST MODE
```

Table 3

<BCS table> (internal representation)

```
    .OD<CR>BB<CR>DO YOU WANT LP810 RUN?<CR>BB<CR>.OI<CR>.LD<CR><CR>
    LP810<CR><CR><CR><CR><CR><CR><CR><CR><CR><CR>0<CR><CR>1<CR>
    .LB<CR>BB<CR><table term><CR>
```

<line #>	<char string>	Action
1	.OD	Operator Decision
2	BB	Transfer label on 'no'
3	DO YOU WANT LP810 RUN?	<text> output to operator
4	.OI	Prompts operator for yes/no
5	<CR>	Insert diskette answer
6	.LD	Load Diagnostic
7	LP810	CRT911 Diagnostic
8	<CR>	Default load bias
9	<CR>	Default CRU base
10	<CR>	Default interface type
11	<CR>	Default clock frequency
12	<CR>	Default interrupt level
13	<CR>	Default run with interrupts
14	<CR>	Default ASCII character set
15	<CR>	Default VCO option
16	<CR>	Default line frequency
17	0	Answer for manual intervention
18	<CR>	Default column width
19	1	Answer for execute EA verb
20	.LB	Establish Label
21	BB	Label BB
22	<table term>	Causes DOCS to exit "Execute List" mode to "Normal" mode and return to 'VERB?'

The list can also be built using the DX10 Assembler. It is necessary to use the 'TEXT' statement to set up the list as it must appear in <BCS table>. The following gives the program needed to build the above list:

```
        IDT 'EXAMPLE'
CR      EQU >0D                 EQUATE CR WITH ASCII
        TEXT '.OD'              OPERATOR DECISION
        BYTE CR
        TEXT 'BB'               LABEL TO BRANCH TO ON 'NO'
        BYTE CR
        TEXT 'DO YOU WANT LP810 RUN ?'      MESSAGE
        BYTE CR
        TEXT '.OI'              OPERATOR INPUT REQUEST
        BYTE CR
        TEXT '.LD'              LOAD DIAGNOSTIC
        BYTE CR
        BYTE CR                 CR AFTER 'LOAD DISKETTE...'
        TEXT 'LP810'            TEST TO BE LOADED
        BYTE CR
        BYTE CR                 DEFAULT LOAD BIAS
        BYTE CR                 DEFAULT CRU BASE
        BYTE CR                 DEFAULT INTERFACE TYPE
        BYTE CR                 DEFAULT COMPUTER FREQUENCY
        BYTE CR                 DEFAULT INTERRUPT LEVEL
        BYTE CR                 DEFAULT RUN WITH INTERRUPTS
        BYTE CR                 DEFAULT FULL ASCII CHAR SET
        BYTE CR                 DEFAULT VCO OPTION
        BYTE CR                 DEFAULT LINE FREQUENCY
        TEXT '0'                DO NOT RUN MANUAL INTERVENTION
        BYTE CR
        BYTE CR                 DEFAULT COLUMN WIDTH
        TEXT '1'                EXECUTE EA
        BYTE CR
        TEXT '.LB'              ESTABLISH LABEL
        BYTE CR
        TEXT 'BB'               LABEL 'BB'
        BYTE CR
        TEXT '@'                TABLE TERMINATOR
        BYTE CR
        END
```

The person building the list would use a DX10 system and perform the SCI command XMA giving the file pathname of the above program

as the SOURCE ACCESS NAME. The specified OBJECT ACCESS NAME would be the file that should be placed on the DOCS load media as the list to be loaded by the '.LL'. Once the list is loaded by the '.LL' verb, the list could be executed by the '.EL' verb.

It would be more practical to use the second procedure for building lists that are to be used often. The list can be loaded much faster than it can be built using '.BL'. For special lists, the first procedure using the '.BL' verb is more practical. For developing new lists, the user should use the '.BL' verb to organize and initially develop the list. The list can then be printed and the DX10 Assembler used to make a permanent copy of the list for later loading by the '.LL' verb.

5.0 PART NUMBERS

The part numbers for all modules and Fully Linked Object (FLO) modules can be found in the DOCS Program Description (P/N 02250248-9901).

APPENDIX Algorithms For The BCS Verbs

The following pages present the algorithms for all of the BCS verbs. The algorithms are written in metacode and are anotated to show usage of service routines and methods.

INTERNAL TABLE BCS TABLE REPRESENTATION - BCS$TB

<BCS table> = #<char string input>]#<input list>]<table term><CR>

where:

<char string input> = #<char string>]<CR>

<input list> = <char string input>#<input list>]

<char string> = any legal ASCII character string (up to 50 characters) input by SSIRP or inserted by text editing commands (except "@",< R>,".EL",".BL",".LL", ".LL",".IL",".DL",".RL",".CL", or ".PL"). <char string> can also be <null>.

<CR> = >0D which is generated by the "NEW LINE" or "RETURN" keys when in the 'Build List' mode.

<table term> = @

<insert> = >F1 (Text editing insert character)

<delete> = >F2 (Text editing delete character)

FLAGS, LABELS, AND POINTERS:

BLFLG - Build List Mode Flag
ELFLG - Execute List Mode Flag
BCSENT - BCS$TB Entry Flag
BCS$TB - Label at beginning of BCS table.
BCS$ED - Label at end of BCS$TB space.
BCS$L - Length of BCS$TB (BCS$ED-BCS$TB)
BCS$CE - Pointer to current entry location in BCS$TB. Points to location of <table term>.
BCS$EX - Pointer to next input during table execution. Initially set to beginning of BCS$TB. It is set during .IS and when BCS$TB has been traversed.
BCS$FG - BCS Flag word. This word contains all of the flags (0->1F) that can be set, reset, modified, and tested by the BCS commands. Flag 0 is designated as the System Error Flag and is set any time a call is made to EMOUT.
BCS$NO Echo suppression flag for BCSGET routine

BCS$PF	No print flag
BCS$AT	Automatic terminate enabled flag
BCS$ET	Execute List terminate enabled flag
BCS$VB	BCS verb table

NOTES:

- .DL, .RL, and .IL may fill up table with text editing and text characters and will erase original input.

- .BL will put DOCS in 'Build List Mode' and "@" will take DOCS out but will not compress table.

- .BL initializes the BCS$TB and starts building from scratch.

- .EL will put DOCS is 'Execute List Mode' and "@" will take DOCS out.

- a CRASH will put DOCS in 'Normal Mode' from either Build List or Execute List Mode.

- defaults are set up and maintained for all verb inputs.

```
Build List                    .BL

      If ELFLG set, then output 'ILLEGAL INPUT'
        and CRASH (@)
      If BCSENT set, then
        Begin
          Delete last entry in BCS$TB
            (".BL<CR>" at BCS$CE-4)          -DELSTG
          Output 'ALREADY IN BUILD LIST MODE'
          Return to 'VERB ?'
        End
      Put <table term><CR> at BCS$TB
      BCS$CE <-- BCS$TB
      Set BCSENT
      Set BLFLG
      Output 'ENTERING BUILD LIST MODE'
      Return to 'VERB?'
```

Execute List .EL - <line #>

where:
<line #> = line number at which execution is to start.
If nothing input, execution resumes where left
off from previous .EL.

```
If BCSENT set, then
  Begin
    Delete last entry in BCS$TB
      (".EL<CR>" at BCS$CE-4)          -DELSTG
    Output 'ILLEGAL INPUT'
  End
Else
  If ELFLG set, then output 'ILLEGAL INPUT'
    and CRASH (@)
  Else
    Begin
      Get line number at BCS$EX and
        put in <line #>
      Input <line #>
      If <line #> in BCS$TB, then      -SEARCH(exclusive)
        Begin
          BCS$EX <-- LOC(<line#>)
          If BCS$EX=BCS$TB, then BCS$FG <-- 0
          Set ELFLG
          Output 'ENTERING EXECUTE LIST MODE'
        End
      Else output 'NO TABLE ENTRY AT <line #>'
    End
Return to 'VERB?'
```

```
If ELFLG set, then output 'ILLEGAL INPUT'
  and CRASH (@)
If BCSENT set, then
  Begin
    Delete last entry in BCS$TB
      (".LL<CR>" at BCS$CE-4)          -DELSTG
    Output 'ILLEGAL INPUT'
  End
Else
  Begin
    Go to loader which displays menu and loads
      selected file into memory at user defined
      location.  Transfer data to BCS$TB - do not
      overflow table.
    BCS$EX <-- BCS$TB
    BCS$CE <-- LOC(<table term>)
  End
Return to 'VERB?'
```

Load and Go .LG - <DOCS loading parameters>

```
Begin
    Go to loader which displays menu and loads
      selected file into memory at user defined
      location.  Transfer data to BCS$TB - do not
      overflow table.
    BCS$EX <-- BCS$TB
    BCS$CE <-- LOC(<table term>)
    Set ELFLG
    Output 'ENTERING EXECUTE LIST MODE'
End
Return to 'VERB?'
```

```
Insert Line                .IL - <line #> - <char string>

*** This inserts prior to <line #>


      If ELFLG set, then output 'ILLEGAL INPUT'
        and CRASH (@)
      If BCSEBT set, then delete last entry
        in BCS$TB (".IL<CR>" at BCS$CE-4)   -DELSTG
      Save current value of BCSENT
      Reset BCSENT
      Input <line #>
      If <line #> in BCS$TB, then            - SEARCH(inclusive)
        Begin
          Input <char string>
        Insert <insert><char string><CR> at
          location <line #>
        End
      Else output 'NO TABLE ENTRY AT LINE <line #>'
IL$1  Restore BCSENT
      Return to 'VERB?'
```

```
Delete Line                .DL - <line #>

       If ELFLG set, then output 'ILLEGAL INPUT'
         and CRASH (@)
       If BCSENT set, then delete last entry
         in BCS$TB (".DL<CR>" at BCS$CE-4)     - DELSTG
       Save current value of BCSENT
       Reset BCSENT
       Input <line #>
       If <line #> in BCS$TB then              - SEARCH(exclusive)
         Begin
           If <line #> already deleted then
             Output 'LINE ALREADY DELETED' and go to DL$1
           Else
             Begin
               Replace <char string> at LOC(<line #>) with <delete>
               Compress BCS$TB (starting at LOC(<line #>)+1)
                 up to next <CR>               - DELSTG
                                               - INSSTG
             End
         End
       Else output 'NO TABLE ENTRY AT LINE <line #>'
DL$1   Restore BCSENT
       Return to 'VERB?'
```

```
Replace Line                .RL - <line #> - <char string>

*** Does not allow <table terminator> to be replaced.


       If ELFLG set, then output 'ILLEGAL INPUT'
         and CRASH (@)
       If BCSENT set, then delete last entry
         in BCS$TB (".RL<CR>" at BCS$CE-4)    - DELSTG
       Save current value of BCSENT
       Reset BCSENT
       Input <line #>
       If <line #> in BCS$TB, then             - SEARCH(exclusive)
         Begin
           Input <char string>
           Delete <char string input> at LOC(<line #>) - DELSTG
           Insert <char string><CR> at LOC(<line #>) - INSSTG
         End
       Else output 'NO TABLE ENTRY AT LINE <line #>'
RL$1   Restore BCSENT
       Return to 'VERB?'
```

Compress List .CL

*** Delete all <insert> and <delete> editing characters.

```
        If ELFLG set, then output 'ILLEGAL INPUT'
          and CRASH (@)
        If BCSENT set, then delete last entry in
          BCS$TB (".CL<CR>" at BCS$CE-4)        -DELSTG
        <scan char> <-- (BCS$TB)
        <current pos> <-- BCS$TB
        While <scan char>.NE.<table term>, Do
          If <scan char>=<insert>, then         -CPSCHR
            compress BCS$TB by 1 char at <current pos>
          If <scan char>=<delete>, then         -CPSCHR
            delete <char string input>s at <current pos>
            and go to CL$1
          <current pos> <-- <current pos>+1
CL$1      <scan char> <-- (<current pos>)
        End Do
        Restore NOEFLG
        Return to 'VERB?'
```

```
Print List          .PL - <line #> - <# of lines> - <output dev>
                 or .PL - 0 - <output dev>
    where:
      - Default value for <# of lines> is initially 1.
        After use, it retains its input value.
      - <output dev> is I/O or Err Msg device selected in .IS.
        No paging is done. 1=I/O, 0=Err Msg. Output is done
        with SSOUT.
      - If <line #>+<# of lines> .GT. end of table, then
        print continues to end of table.
      - If <line #> is zero, print the whole table.


      If ELFLG set, then output 'ILLEGAL INPUT' and
        CRASH (@)
      If BCSENT set, then delete last entry in
        BCS$TB (".PL<CR>" at BCS$CE-4)      -DELSTG
      Save current value of BCSENT
      Reset BCSENT
      N$LINE <-- 0
      Input <line #>
      If <line #>=0, then <line #> <-- 1 and
        N$LINE <-- BCS$L
      If <line #> in BCS$TB, then           -SEARCH(inclusive)
        Begin
          If N$LINE=0, then input <# of lines> and
            N$LINE <-- <# of lines>
          Input <output dev>     (use yes/no input)
          LINE$N <-- <line #>
          While N$LINE .NE. 0, Do
            If deleted line, then output '*' and decrement
              N$LINE and increment LINE$N
            If inserted line, then output <char string> -BCSOUT
            If normal line, then output                 -BCSOUT
              '(LINE$N)<char string>' and decrement
              N$LINE and increment LINE$N
            If <char string>=<table term>, then PL$1
          End Do
        End
      Else, output 'NO TABLE ENTRY AT LINE <line #>'
PL$1  Restore BCSENT
      Return to 'VERB?'
```

Establish Label .LB - <label value>

where:
<label value> = any 1-4 character string

```
If both BLFLG and ELFLG reset, then
  output 'ILLEGAL INPUT'
If ELFLG set, then set BCS$NO  BCS$PF
Input <label value>    (no default accepted)
Clear BCS$NO
clear BCS$PF
Return to 'VERB?'
```

Unconditional Transfer .GT - <label value>

```
If both BLFLG and ELFLG reset, then
  output 'ILLEGAL INPUT' and return to 'VERB ?'
If ELFLG set, then set BCS$NO and BCS$PF
Input <label value>
If ELFLG set, then
  Begin
    Get <byte add> of <char input string>      -LBSRCH
      which follows <label value><CR> in
      BCS$TB
    Clear BCS$NO
    Clear BCS$PF
    If <error status>, then
      output 'LABEL NOT FOUND' and CRASH
    Else, BCS$EX <-- <byte add>
  End
Return to 'VERB?'
```

Conditional Transfer .IF - <flag #> - <label value>

*** If <flag #> is set, then transfer to <label value>

where:
<flag #> = BCS flag number (0->1F)
<label value> = any 1-4 character string

```
If both BLFLG and ELFLG reset, then
  output 'ILLEGAL INPUT' and return to 'VERB ?'
If ELFLG set, then set BCS$NO and BCS$PF
Input <flag #> and <label value>
If ELFLG set, then
  Begin
    Get VAL(<flag #>)                                  -FLAG
    Clear BCS$NO
    Clear BCS$PF
    If <error status>, then output
      'NONEXISTENT FLAG' and CRASH (@)
    If VAL(<flag #>) = 1, then
      Begin
        Get <byte add> of <char string input>   -LBSRCH
          which follows <label value><CR> in
          BCS$TB
        If <error status>, then
          output 'LABEL NOT FOUND' and CRASH (@)
        Else, BCS$EX <-- <byte add>
      End
  End
Clear BCS$NO
Clear BCS$PF
Return to 'VERB?'
```

```
Clear Flag                    .CF - <flag #>

    where:
      <flag #> = BCS flag number (0->1F)


      If ELFLG set, then set BCS$NO and BCS$PF
      Input <flag #>
      Clear <flag #>                          -FLAG
      CLear BCS$NO
      Clear BCS$PF
      If return status=>FFFF, then output
        'NONEXISTENT FLAG' and CRASH (@)
        End
      Return to 'VERB?'

Set Flag                      .SF - <flag #>

    where:
      <flag #> = BCS flag number (0->1F)


      If ELFLG set, then set BCS$NO and BCS$PF
      Input <flag #>
      Set <flag #>                            -FLAG
      Clear BCS$NO
      Clear BCS$PF
      If return status=>FFFF, then output
        'NONEXISTENT FLAG' and CRASH (@)
      Return to 'VERB?'
```

Modify Flag .MF - <flag #> - <flag value>

where:
<flag #> = BCS flag number (0->1F)
<flag value> = 0 or 1

If ELFLG set, then set BCS$NO and BCS$PF
Input <flag #> and <flag value> (use yes/no input)
Put <flag value> in <operation> field of flag
Modify <flag #> -FLAG
Clear BCS$NO
Clear BCS$PF
If return status=>FFFF, then output
'NONEXISTENT FLAG' and CRASH (@)
Return to 'VERB?'

Operator Input .OI

*** In Execute List Mode, the operator is prompted to input the required data instead of taking it from BCS$TB. Detection of ".OI" is done by SSGASC and the call is reissued if ELFLG is set. In Build List Mode, the call is reissued but the data input is not stored in BCS$TB.

Return to 'VERB?'

Operator Comment .OC - <text>

*** Output <text> to selected I/O device.

where:
<text> = <char string> up to 50 characters

If both BLFLG and ELFLG reset, then output
'ILLEGAL INPUT' and CRASH (@)
If ELFLG set, then set BCS$PF
Input <text>
Clear BCS$PF
Return to 'VERB?'

```
Operator Decision          .OD - <label value> - <text> - <rspn>

*** Output <text> to selected I/O device. Transfer to
    <label value> if <rspn> is "no".

    where:
      <text> = <char string> up to 50 characters
      <label value> = <char string> up to four characters
      <rspn> = yes/no response from operator or from
               BCS$TB in the form of 1=YES, 0=NO


     If both BLFLG and ELFLG reset
       Then begin
         Output 'ILLEGAL VERB'
         Return to 'VERB ?'
       End
     If ELFLG set
       Then begin
         Set BCS$NO
         Set BCS$PF
       End
     Input <label value>
     Clear BCS$NO
     Input <text>
     Clear BCS$PF
     Input <rspn>
     If ELFLG set and <rspn> is no
       Then begin
         Search for <label value> in BCS$TB       -LBSRCH
         If <error status>
           Then begin
             Output 'LABEL NOT FOUND'
             CRASH
           End
         BCS$EX <-- <byte addr> in BCS$TB
       End
     Return to 'VERB ?'
```

Automatic Terminate .AT

*** Set up flags necessary to terminate the execution of a batch stream if a call to EMOUT occurs.

```
If BCSENT not set
  Then begin
    Set BCS$AT
    Clear BCS flag '0'
  End
Return to 'VERB ?'
```

*** The following takes place in EMOUT:

If BCS$AT set then set BCS$ET

*** The following takes place in VDENTP:

If BCS$ET set then CRASH

CRASH and Exit to Normal Mode '@'

*** This terminates the Build List and Execute List Modes. It
    does not compress the editing characters out of BCS$TB.

```
If DCL routine linked, then
  Begin
    If BLFLG set, then
      Begin
        Reset BLFLG
        Output 'TERMINATING BUILD LIST MODE'
      End
    If ELFLG set, then
      Begin
        BCS$EX <-- BCS$TB
        Reset ELFLG
        Output 'TERMINATING EXECUTE LIST MODE'
      End
    Reset BCSENT
  End
Branch to 'VERB?'
```

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV	
SHEET	
REV STATUS OF SHEETS	REV
	SHEET

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± 010
2 PLACE DECIMAL ± 02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL STD 100

DWN DATE

CHK

ENGR

QA

APVD

CONTR NO

DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

PD, AU04, ARITHMETIC UNIT TEST-990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	937754-9901

SCALE	REV.	SHEET
		1 of 11

TI-0023-C

TABLE OF CONTENTS

1.0 SCOPE

This document describes the 990/4 Arithmetic Unit Test, AU04. This test will operate as a standalone test in the 990/4 computer.

2.0 REFERENCES

For information beyond the scope of this document refer to the following:

PART NUMBER	TITLE
945251-9701	990/4 System Hardware Reference Manual
943441-9701	990 Computer Assembly Language Manual
945400-9701	Diagnostic Handbook
945403-9701	99/4 System Depot Maintenance Manual

3.0 EQUIPMENT AND SOFTWARE REQUIREMENTS

AU04 requires the following equipment:

a) 990/4 Computer with 4K words of memory.

b) Any of the following I/O devices:

1) ASR/KSR Terminal.

2) 913 VDT Terminal.

3) 911 VDT Terminal.

4) 810 or similar line printer.

c) Appropriate loading device.

The loadable test module is AU04, 937754-1006 (FLO).

Linkable Parts: AU04

DSRS

AU04TXT

4.0 LOADING INFORMATION

This is a standalone test and reference should be made to the Diagnostic Handbook for loading procedures from all available media.

5.0 TEST EXECUTION AND DESCRIPTION

The AU04 diagnostic runs as a standalone test or under the control of the Burn-in or Quality-assurance Supervisors. The test verifies the operation of the AU by:

1) Executing each instruction and verifying the execution results and status.
2) Verifying the proper occurrence of clock interrupts at level 5 using the CKON and CKOF instructions.

When the AU04 test is loaded the following message is printed:

AU04 990/4 ARITHMETIC TEST VERSION MM/YY/*R

Where the version is the release date and revision level of the test. If no errors occur during the execution of the test the message

NO ERRORS

is printed at the end of the test. The message

AU TEST COMPLETED

is then printed.

6.0 MESSAGES

When an error occurs an error number is displayed on the Programmer's front panel in the following format:

FFXX

Where FF are the hex digits and XX is the number corresponding to the instruction or interrupt under test as given in tables 4-1 and 4-2.

The error message for the instruction tests is:

ERROR: XXXX INSTRUCTION

Where XXXX is the instruction under test as given in table 4-1 The address relative to the beginning of the program where the error was detected is given as XXXX in the message:

PGM LISTING ADDR: XXXX

The contents of the Workspace Pointer, Program Counter, Status Register, and 16 word Workspace are printed as follows:

ERROR WP= XXXX PC=XXXX ST=XXXX WORKSPACE=

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

Occurrence of an unexpected interrupt will cause the appropriate message in table 4-2 to be printed.

TABLE 4-1 INSTRUCTION ERROR NUMBERS

ERROR NO. DECIMAL	ERROR NO. HEX	INSTRUCTION
1	01	C
2	02	CB
3	03	MOV
4	04	MOVB
5	05	A
6	06	AB
7	07	S
8	08	SB
9	09	SOC
10	0A	SOCB
11	0B	SZC
12	0C	SZCB
13	0D	JMP
14	0E	JLT
15	0F	JLE
16	10	JEQ
17	11	JHE
18	12	JGT
19	13	JNE
20	14	JL
21	15	JH
22	16	JOP
23	17	JNC
24	18	JOC
25	19	JNO
26	1A	COC
27	1B	CZC
28	1C	XOR
29	1D	SRA
30	1E	SLA
31	1F	SRL
32	20	SRC
33	21	LI
34	22	AI
35	23	ANDI
36	24	ORI
37	25	CI
38	26	STWP
39	27	LWPI
40	28	CLR
41	29	NEG
42	2A	INV
43	2B	INC
44	2C	INCT

45	2D	DEC
46	2E	DECT
47	2F	SETO
48	30	SWPB
49	31	ABS
50	32	B
51	33	BL
52	34	BLWP
53	35	X
54	36	RTWP
55	37	RSET
56	38	CKON
57	39	CKOF
58	3A	IDLE
59	3B	DIV
60	3C	MPY
61	3D	XOP

TABLE 4-2 UNEXPECTED INTERRUPT ERROR NUMBERS

ERROR NO. DEC	ERROR NO. HEX	INT. LEVEL	MESSAGE
64	40	2	PARITY ERROR
69	45	XX	UNEXPECTED INT LEVEL= XX
70	46	0/1	POWER FAILURE OCCURRED TEST RESTARTED

7.0 PART NUMBERS

TITLE	PART NUMBER	
Program Description	937754-2001	ROFF Source
	-9001	ROFF Output
	-9901	PD Document
	-0009	SP, Microfiche
AU04, Linked Test	-1006	FLO
	-9006	LML
	-7006	LC
AU04, Main Test Module	-2002	SRC
	-1002	OBJ
	-9002	LST
DSRS, I/O Routines	2268486-2003	SRC
	-1003	OBJ
	-9003	LST
AU04TXT, Messages	2250562-2003	SRC
	-1003	OBJ
	-9003	LST

A

APPLICATION	
NEXT ASSY	USED ON
	7506

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

REV
SHEET

REV STATUS OF SHEETS — REV / SHEET

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± .010
2 PLACE DECIMAL ± .02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN — DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group — Dallas, Texas

PD, AU04TEST, 990/4 ARITHMETIC UNIT TEST-990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	937983-9901

SCALE — REV. E — SHEET 1 of 16

YI-0023-C

TABLE OF CONTENTS

1.0 SCOPE

This document describes the 990/4 Arithmetic Unit Test, AU04TEST. This test will operate as a standalone test in the 990/4 computer.

2.0 REFERENCES

For information beyond the scope of this document refer to the following:

PART NUMBER	TITLE
945251-9701	990/4 System Hardware Reference Manual
943441-9701	990 Computer Assembly Language Manual
945400-9701	Diagnostic Handbook
945403-9701	99/4 System Depot Maintenance Manual

3.0 EQUIPMENT AND SOFTWARE REQUIREMENTS

AU04TEST requires the following equipment:

a) 990/4 Computer with 8K words of memory.

b) Any of the following I/O devices:

1) ASR/KSR Terminal.

2) 913 VDT Terminal.

3) 911 VDT Terminal.

4) 810 or similar line printer.

c) Appropriate loading device.

The loadable test is AU04TEST , 937983-1007 (FLO)

Linkable Parts: AU04A AU04B DSRS AU04MSG

4.0 LOADING INFORMATION

This is a standalone test and reference should be made to teh Diagnostic Handbook for loading procedures from all available media.

5.0 TEST EXECUTION AND DESCRIPTION

The AU04TEST diagnostic performs the following tests:

1) Real Time Clock test
2) Level 2 Interrupt Test
3) Register Test
4) ALU Test
5) Microcode Test

Every effort was made to isolate test functions to a specific subtest to avoid overlap in the subtests. This effort succeeded to a large extent. However, a certain amount of overlap was impossible to prevent. In cases where the overlap of test functions resulted in extensive testing of a particular function, that function is not retested in a later test.

This is especially noticeable in the Microcode test, whereas in the last subtest a high degree of overlap takes place, the Microcode test does not assume a particular function is sufficiently tested unless repeating a test is especially redundant. A good example of this is in testing the interrupt handling microcode, which is tested extensively in the Level 2 Interrupt test, and not tested in the Microcode test.

The only major area of the 990/4 not tested is the CRU logic. This ommission is due to the inability of

hardware to provide a means for the software to have sufficient access to insure correct operation. However, the CRU logic is extensively used in I/O operations and must be functioning for proper operation.

This test is written in such a way that looping on errors is very easy. All branches to error routines are done via the BL instruction. To loop on and error change the word following the BL, which is the address of the error routine, to the address of the desired loop. In this way, instead of branching to an error routine, a branch to any location in the subtest can be taken.

In setting up loops, caution must be used to obtain accurate results. At the beginning of each test except the real time clock test, addressing limits are set. These limits are checked during RTC interrupts to serve as control against the program setting lost. If an Out-of-limit condition is detected, an error is reported.

5.1 TEST 1 -- REAL TIME CLOCK TEST

This test is designed to test the proper functioning of the Real Time Clock and clock interrupts. In addition, the ability of the AU to interrupt out of idle mode is tested by an interrupt on the Real Time Clock.

The Real Time Clock's stability is tested by determining how many times a loop of code can be executed during a full clock period. This count is used as a standard against repeating the same sequence 1000 times and comparing each clock period's count against the standard. The test will take approximately 8.5 seconds with a 60 Hz line frequency.

5.2 TEST 2 -- LEVEL 2 INTERRUPT TEST

This test will test the level 2 interrupt. The test does very extensive testing of the general interrupt handling capabilities of the AU. The ability of the test to service different levels of interrupt in a priority level is tested. The masking of interrupts is also tested.

Areas that are tested in this test, eliminating the need for additional testing in other tests, are the privop and the status register privop flags, the XOP instruction and the XOP status register flag.

5.3 TEST 3 -- REGISTER TEST

This test is designed to test the internal registers on the 990/4. The tested registers are the PC, WP, ST, DA, and DD.

The first register tested is the WP register. This test is done by loading the WP with a test value, then doing a

BLWP instruction to save the loaded test value for comparison. The test value is then compared to determine if an error has occurred. The test values used are >0000, >FFFE, >AAAA, and >5554.

After completing the WP register test, the ST register is tested. The first part of testing the ST is to set all interrupt masks, then test the remainder of the ST register bits by executing instructions to set and reset all the bits. All the ST register bits are tested, except for the PRIVOP and XOP bits, which are tested in TEST 2.

The PC is tested by initializing memory from the end of the test to the end of memory to 0414 (BLWP *R4). Then a branch will be made successively to each memory location so initialized. Upon return, R14 will be compared with the expected value to check for errors. A final test is done on a PC value of >0000 to insure that all bits will turn off.

As a consequence of testing the WP, ST, and PC registers, the DA and DD registers are tested. This is the only possible test of these register, as there is no direct way for software to to access these registers.

5.4 TEST 4 -- ALU TEST

This test is designed to test the ALU's and the data paths associated with ALU functions. The test treats the 4 ALU's as independent 4 bit units. Add and subtract operations are done on the ALU's with the results compared after each operation. If the result is not equal, one of the ALU's is in error and an ALU error is reported. The data is incremented so that all combinations of 4 bits are tested in each ALU.

The second part of the test performs multiply and divide operations on the ALU's with all values of data, from 0 to 64K. This part of the test, in addition to testing the ALU's will find and test the longest delay path thru the system.

5.5 TEST 5 -- MICROCODE TEST

This test is designed to exercise all microcode in the 990/4 Computer. The microcode is tested by executing instructions and addressing modes which will cause all microcode states to be executed at least once. Some microcode states may not be tested in this test (i.e. CRU instructions).

All tested microcode states and untested states are documented in the program listing (PN: 937983-9002). In order to determine the cause of an error, the microcode

flowcharts are necessary.

6.0 MESSAGES

6.1 ERROR MESSAGES

REGISTER 11 IN THE FOLLOWING PRINTOUT CONTAINS THE RELATIVE ADDRESS OF THE BRANCH TO THE ERROR ROUTINE
This message indicates how to trace the cause of the error.

PGM LISTING ADDR: XXXX
This message tells where the branch to print the messages took place. It will usually be useless in tracing errors.

WORK POINTER PC STATUS AT TIME OF ERROR WP = XXXX PC = XXXX ST = XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX
This message is a dump of all registers in use at the time of the error.

ERROR RETURN FROM SUBTEST
This message indicates that an error occurred in the subtest.

ERRORTHE PC FOR THE CURRENT TEST IS OUT OF THE ALLOWABLE ADDRESSING AREA
ADDRESS OF PRIOR AND CURRENT INTERRUPT

CURRENT WP XXXX PC XXXX ST XXXX

PRIOR WP XXXX PC XXXX ST XXXX

This message is displayed when the RTC error check detects an out-of-bounds condition.

RTC DID NOT OCCUR IN WINDOW

This message indicates the RTC interrupt did not occur in the time frame of the first interrupt (± 10).

STATUS REG. AFTER INTER. = XXXX

This messge indicates the interrupt mask was incorrectly set upon receipt of the clock interrupt.

ERROR*THE REAL TIME CLOCK FAILS TO INTERRUPT

ERROR*LEVEL 2 INTERRUPT TEST

ERROR*THE REGISTER TEST HAS AN ERROR

ERROR*ALU ERROR

ERROR*MICROCODE ERROR

6.2 HEADER MESSAGES

Start test message:

AU04TEST 990/4 ARITHMETIC UNIT TEST VERSION MM/YY/*R

Note: version is month, year, and revision level.

Start subtest messages:

```
START SUBTEST 1 - REALTIME CLOCK TEST

START SUBTEST 2 - LEVEL 2 INTERRUPT TEST

START SUBTEST 3 - REGISTER TEST

START SUBTEST 4 - ALU TEST

START SUBTEST 5 - MICROCODE TEST

End of subtest message:

SUBTEST COMPLETE

Count of test loops completed:

LOOPS COMPLETED = XXXX

End of test message:

TEST COMPLETE
```

7.0 PART NUMBERS

TITLE	PART NUMBER
PROGRAM DESCRIPTON	937983-2001 ROFF SRC
	-9001 ROFF OUTPUT
	-9901 PD DOC
AU04TEST	-1006 FLO
	-9006 LML
	-7006 LC
	-0009 SP
AU04A	937983-2002 SRC
	-1002 OBJ
	-9002 LST
AU04B	937984-2002 SRC
	-1002 OBJ
	-9002 LST
AU04MSG	2250295-2002 SRC
	-1002 OBJ
	-9002 LST
DSRS	2260486-2003 SRC
	-1003 OBJ
	-9003 LST

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7046				

REV																														
SHEET																														
REV STATUS OF SHEETS	REV																													
	SHEET																													

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

	DWN	DATE
	CHK	
	ENGR	
	QA	
	APVD	
	CONTR NO	
	DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

PD, AU05, ARITHMETIC UNIT TEST-990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	2267307-9901

SCALE	REV.	SHEET 1 of 47

TI-0023-C

TABLE OF CONTENTS

1.0 SCOPE

This document describes the 990/5 Arithmetic Unit Test. The diagnostic is designed to operate as a standalone test. The diagnostic will accomplish the following goals:

A) Verify the proper functioning of the Arithmetic Logic Unit for the 990/5 computer.

B) To isolate any faults to the bad logic chip or section of logic that may be causing the failure.

2.0 REFERENCES

For information beyond the scope of this document refer to the following:

TITLE	PART NUMBER
990 COMPUTER ASSEMBLY LANGUAGE MANULE	943441-9701
DIAGNOSTIC HANDBOOK	945400-9701
TMS 9900 DATA BOOK	
TMS 9902 DATA BOOK	
TMS 9903 DATA BOOK	

3.0 EQUIPMENT AND SOFTWARE REQUIREMENTS

The following paragraphs describe the minimum requirements to execute the diagnostic:

3.1 EQUIPMENT REQUIREMENTS

A) 990/5 COMPUTER WITH 16K OR 32K WORDS OF MEMORY

B) APPROPRIATE LOADING DEVICE

C) APPROPRIATE LOADING MEDIA

D) APPROPRIATE I/O DEVICE

3.2 SOFTWARE REQUIREMENTS

LOADABLE TEST MODULE:	AU05, 2267307-1006 (FLO)
LINKABLE PARTS:	AU05A
	AU05B
	AU05C
	AU05D
	STANDIO
	AU05MSG

4.0 LOADING

Loading procedures from all available media are found in the DIAGNOSTIC HANDBOOK. (PN 945400-9701)

5.0 TEST DESCRIPTIONS

This diagnostic was designed to isolate faults to a specific area of logic for the 990/5 Arithmetic and Logic Unit. Each subtest will exercise a section of that logic. The individual blocks that are tested are broken down in the following manner:

A) REAL TIME CLOCK TEST

B) INTERRUPT AND XOP TEST

C) REGISTER TEST

D) ARITHMETIC LOGIC UNIT TEST

E) INSTRUCTION TEST

F) TMS 9902 AND TMS 9903 CHIP TEST

5.1 REAL TIME CLOCK TEST

This test will exercise the REAL TIME CLOCK LOGIC. The test has two sections.

Part One will be a test that insures clock interrupts occur. This is done by having the clock interrupt the computer out of an "IDLE" condition.

Part Two will be a test that insures the stability of the clock. This is done in the following manner:

1) A standard (X) is set by executing a loop of code. "X" is determined by how many times the loop is completed before a clock interrupt occurs.

2) "X" is then compared against a new value computed by executing the loop again. "X" is allowed to vary by plus or minus 10. These constants can be found in registers 3 and 6. Register 4 will contain the value that was computed for that pass. This is done 999 times.

Possible causes for error in this test are the following:

1) The Clock didn't generate an interrupt to the CPU.

2) The CPU didn't recognize an interrupt from the Clock.

3) The line frequency is unstable, which causes the clock to be unstable.

Failures one and two are indicated if the following happens:

1) Header Messages are printed and test execution begins.

2) The computer goes to an "IDLE" state.

Recommended action is to check the interrupt logic for the clock. The interrupt level for the clock is generally Level 5 but may be at Level 15. If the operator isn't sure what the level is for that particular system, he may find it at location >05EC + (offset). Where offset is the load bias for the test.

Failure three is indicated by an error message.(See section 6.2.3) Recommended action is to analize the power supply for variation or spikes.

5.2 INTERRUPT AND XOP TEST

This test is designed to test the Interrupt and Extended Operation (XOP) circuitry. There are two parts to this test.

Part one is the interrupt test. This portion of the test will execute a series of instructions that cause a Level 2 interrupt to occur. This is accomplished by modifing location >FBFE. Location >FBFE is reserved for this function. Register 4 contains >FBFE. The instructions used are the following:

```
0)  MOV   *R4,*R4
2)  MOVB  *R4,*R4
4)  A     *R4,*R4
6)  AB    *R4,*R4
8)  S     *R4,*R4
A)  SB    *R4,*R4
C)  C     *R4,*R4
E)  CB    *R4,*R4
10) SZC   *R4,*R4
12) SZCB  *R4,*R4
14) SOC   *R4,*R4
16) SOCB  *R4,*R4
18) COC   *R4,R8
1A) CZC   *R4,R8
1C) XOR   *R4,R8
1E) LDCR  *R4,0
20) LDCR  *R4,8
22) STCR  *R4,0
24) STCR  *R4,8
26) STCR  *R4,8          SPARE
28) CLR   *R4
2A) CLR   *R4            SPARE
2C) CLR   *R4            SPARE
2E) NEG   *R4
30) INV   *R4
32) INC   *R4
34) INCT  *R4
36) DEC   *R4
38) DECT  *R4
3A) CLR   *R4            SPARE
3C) SWPB  *R4
3E) SETO  *R4
40) ABS   *R4
42) MPY   *R4,R8
44) DIV   *R4,R8
```

The hexadecimal numbers before each instruction will be the same as the number in register three if an error occurs. Recommended action is to check the LEVEL 2 interrupt logic. It is possible that one of the instructions is failing. This could be due to the TMS 9900 failing to access memory correctly or failure of the TMS 9900 entirely.

Part two is the XOP test. The only XOP tested is XOP 15, because all XOPs must not alter the STATUS REGISTER, the PROGRAM COUNTER REGISTER, and the WORK SPACE REGISTER. Based upon this assumption, the test uses all possible variations of an XOP instruction. Thus if XOP 15 doesn't alter these registers then the XOP circuitry is working correctly. Below are the XOPs used by the test.

1) XOP 6,15

2) XOP @0,15

3) XOP @6,15

4) XOP *12,15

5) XOP *12+,15

All of these XOPs cause the computer to trap to the contents of location >7E, the work space for that trap will be held in location >7C. The differences are where the data to be used by XOP is stored. In number one the data is stored in the caller's work space register 6. In number two the data is stored in label 0. Number three is the same as number two except that the data is at label 6. Number four has the data stored at the location held in the caller's work space register 12. Number five has the data stored at the location held in the caller's workspace register 12 with an auto-increment of one word every time the XOP is used.

In this test if an error occurs some of the causes may be as follows:

1) INCORRECT MEMORY ACCESSES BY THE TMS 9900

2) FAILURE OF THE TMS 9900

Recommended action would be to check the memory addressing circuitry. The only other action would be to replace the TMS 9900.

5.3 REGISTER TEST

This test will verify that the following internal registers operate correctly.

1) The STATUS REGISTER (ST)
2) The PROGRAM COUNTER REGISTER (PC)
3) The WORK SPACE POINTER REGISTER (WP)

The Status register contains the interrupt mask and the conditional codes for the program. The Program Counter or PC contains the address of the next instruction to be executed by the computer. The WP holds a memory location of the beginning of 16 software registers set up by the program.

Part one of this test checks the Work Space Pointer. By definition, the Work Space is a memory location. So to verify that this register works correctly the following is done:

1) the work space is designated to start at memory location >0
2) a "BLWP" is done to a new PC and WP.
3) R13 is checked to see that it contains the original WP
4) the work space is designated to start at memory location >FFFE. then steps 2 & 3 are repeated
5) the work space is designated to start at memory location >AAAA. then steps 2 & 3 are repeated
6) the work space is designated to start at memory location >5554. then steps 2 & 3 are repeated

Part two of this test checks the Status Register. It does this in the following manner:

1) checks the interrupt mask, bits 12-15 of the status register

   a) LWPI >0

   b) STST R1

   c) check to see that the status is 0

   d) change the LWPI from 0 to 1

   e) change the compare from 0 to 1

   f) repeat steps a - e until reach >F

2) check to see that all bits can be turned off and on

   a) LI R15 >FE0F

   b) RTWP

   c) STST R10

   d) compare R10 to >FE0F

   e) change the LI to 0

   f) change the compare to 0

   g) repeat steps b - d

3) check to see that BLWPs don't change the status

a) LI R15,0

b) RTWP

c) BLWP

d) see that the status in R15 hasn't changed

e) repeat for >FE0F

4) check to see that LWPI, STWP, STST, B, AND BL don't change the status with the same type algorithm

Part three will verify the correct operation of the Program Counter. To accomplish this the test will do the following:

1) initialize all unused memory to >0414 which is a "BLWP *R4" instruction

2) starting at the address one word above the end of of the test it will execute the instruction at that location.

3) compare to see that the PC in R14 is correct

4) repeat until the end of memory

Errors from this test will indicate that either the TMS 9900 made a bad memory access or that the TMS 9900 failed. If the error is from part one, look at register 13. This gives the test value that the work space failed. In part two register 15 gives the value that the status failed. For part three register 5 will give the address of the memory location that the test failed.

Recommended action for these errors is to check the memory data paths. The TMS 9900 could be receiving bad data back. As a last solution replace the TMS 9900 chip.

5.4 ARITHMETIC LOGIC UNIT TEST

This test was designed to verify the functioning of the Arithmetic Logic Unit (ALU) of the TMS 9900 chip. The test will have two parts.

1) test the addition and subtraction instructions

2) test the multiply and divide instructions

It uses these four instructions because they are the basic building blocks of all arithmetic functions.

Part one will verify the functioning of the addition and subtraction instructions in the following manner:

1) ADDITION

a) set all computation registers to a value A
   A starts at 0

b) do the addition of the computation registers

c) do a set zeros corresponding to the answer

d) see if the answer is correct

e) increment A by 1

f) repeat a - e until A = >F

On the first pass this will check bits 3,7,11,15

On the second pass this will check bits 2,6,10,14

On the third pass this will check bits 1,5,9,13

On the fourth pass this will check bits 0,4,8,12

2) SUBTRACTION

This section of the test follows the same algorithm

as the addition test. On changing b and e to subtraction and decrement respectively.

The computation registers are R1 - R4. Registers R5 and R6 are the registers used by the SZC instructions. Registers R0, R8, and R9 are used to control the data being added and subtracted. R10 is the loop control register.

Part two will verify the multiply and divide instructions in the following manner:

1) MULTIPLY

The multiply instruction is checked with a shift and add type algorithm.

a) R1= >F, R1 goes to R2

b) R2 is shifted 5 places to the right

c) R2 is swaped, the number in R2 is now in the lower half of the word
R2 goes to R6
R1 goes to R5
R10 gets the multiply count

d) R6 is added to R11
R8 gets the carry if any

e) R5 is shifted to the right 1 place

f) R10 is decremented

g) d - f are repeated until R10 is 0
R11 and R8 hold the answer

h) MPY R2,R3 which puts the answer in R3 and R4

now the two answers are compared to see that they are equal.

2) DIVIDE

The divide instruction is checked with the same type algorithm, shift and subtract.

If any errors occur in this subtest the only action that can be taken is to replace the TMS 9900 chip. There is only a slight chance of bad memory accessing. Once the values are read in, all computation is done in register to register moves. These are internal to the chip.

5.5 INSTRUCTION TEST

This test will exercise most of the instruction set for the TMS 9900 (see APPENDIX A). Each instruction will be tested in all of its modes. Those instructions not tested are either COMMUNICATION REGISTER UNIT (CRU) instructions or have been previously tested, or are not used by the 990/5 computer. If they aren't used they are treated a NOPs (see APPENDIX A).

The first block of instructions tested are the format 1 instructions (see APPENDIX B). These instructions are tested in the following manner:

1) INSTRUCTION Rx,Ry
2) INSTRUCTION *Rx+,Ry
3) INSTRUCTION *Rx,Ry
4) INSTRUCTION @LABEL (Rx),Ry
5) INSTRUCTION @LABEL ,Ry

Where x and y are some work space register number.

The second block of instructions tested are the format 2 instructions (see APPENDIX B). These instructions are PROGRAM COUNTER modifiers. What is done is the condition they need in the STATUS REGISTER is set. The instruction is then executed and the PC is compared to see if it is correct.

The third block of instructions tested are the format 3 instructions (see APPENDIX B). These instructions perform an operation on two memory locations. They are tested in the following manner:

1) INSTRUCTION Rx,Ry
2) INSTRUCTION *Rx+,Ry
3) INSTRUCTION *Rx,Ry
4) INSTRUCTION @LABEL(Rx),Ry
5) INSTRUCTION @LABEL,Ry

The fourth block of instructions tested are the format 8 instructions (see APPENDIX B). These are the immediate instructions. Each of these instructions is executed and compared to see if a correct operation took place.

The fifth block of instructions tested are the format 6 instructions (see APPENDIX B). These instructions require 1 word operands. These instructions are executed and the operand is checked to see that the correct operation took place.

The sixth block of instructions tested are the format 9 instructions (see APPENDIX B). They are tested in the following manner:

1) INSTRUCTION Rx,Ry
2) INSTRUCTION *Rx+,Ry
1) INSTRUCTION *Rx,Ry
1) INSTRUCTION @LABEL(Rx),Ry
1) INSTRUCTION @LABEL,Ry

If an error occurs in this subtest the probable causes are:

1) THE TMS 9900 MADE BAD MEMORY ACCESSES
2) THE TMS 9900 IS BAD

Recommended action is to check the data paths or replace the TMS 9900.

5.6 TMS 9902 & TMS 9903 COMMUNICATION CHIP TEST

This test is designed to verify the proper functioning of the 990/5 onboard communication ports. Port 1 and Port 2 (TMS 9900) are asychronous communication devices, or TTY/EIA interfaces. Port 3 is a sychronous communication device, or a device that will use telephone line communication.

The TMS 9902 ports are tested first. Port 1 is first and is at CRU base >1700, location V10. Port 2 is next and is at CRU base >1740, location V08. These ports are tested in the following manner:

1) The internal clock is checked. This is done in the same manner as part two of the REAL TIME CLOCK test. (see paragraph 5.1)

2) The transmit and receive functions are checked.

a) set the control word to the correct character length. character length is from 5 - 8 bits

b) transmit that character

c) receive that character

d) compare what was transmitted to what was received they should be equal

e) repeat a - d for all character lengths

3) The parity detection is checked.

a) have the receiver look for odd parity

b) transmit an odd parity character

c) see that on reception the parity detection says that it is correct

d) transmit an even parity character

e) see that parity detection picks up the error

f) repeat the procedure for even parity.

The TMS 9903 is tested last. This is Port 3, CRU base >1780, loctation V05. It will use the same tests as above except that its character length is from 5 - 9.

If an error occurs in this subtest the following are the probable causes:

1) CRU LOGIC FAILURE
2) CLOCK INPUT FROM THE TMS 9900 WAS INCORRECT
3) TMS 9902 OR TMS 9903 CHIP FAILURE

Recommended action is the following:

1) make sure the clock frequency is being received correctly from the TMS 9900
2) check the CRU logic between the COMMUNICATION CHIP in error and the TMS 9900
3) replace the COMMUNICATION CHIP in error

6.0 MESSAGES

The messages for this diagnostic are self explanitory. There are two catagories of messages, header and error messages.

6.1 HEADER MESSAGES

Header messages convey general information to the operator. They are listed below.

AU05 990/5 ARITHMETIC UNIT TEST VERSION MM/YY/**

This is the message displayed each time the test is started. MM indicates the month that the diagnostic was last upgraded. YY indicates the year that the diagnostic was last upgraded. ** indicates the revision level of the diagnostic.

START SUBTEST "X" - "NNNNNNNNNNNNNNN"

This message is displayed at the start of each subtest. X indicates the number of the subtest. N indicates the name of the subtest.

SUBTEST COMPLETE

This message indicates the finish of one of the subtests.

TEST COMPLETE

This message indicates that the entire test has completed.

LOOPS COMPLETED = "XX"

This message indicates how many times the test has run to completion.

6.2 ERROR MESSAGES

These messages indicate that an error occurred and a possible area of logic to check. There are three catagories of error messages:

1) LEVEL TWO INTERRUPT ERRORS

2) REAL TIME CLOCK INTERRUPT ERRORS

3) TEST ERRORS

Each of the error types will be discussed in the following paragraphs.

6.2.1 LEVEL TWO INTERRUPT ERRORS

These messages indicate that for some reason the computer made a bad memory access. The message will be as follows:

MEMORY ERROR INTERRUPT
"XXXXXXXXXXXX"
WORK POINTER PC STATUS AT TIME OF ERROR
WP = "WWWW" PC = "PPPP" ST = "SSSS"

X indicates that either a "TILINE TIMEOUT" or a "PARITY ERROR" occurred. A "TILINE TIMEOUT" means that the computer tried to access unavailable memory or that the access accessed a TILINE SLACE DEVICE that failed to respond. A "PARITY ERROR" means that a bad word was read in from memory. W indicates the beginning of the work space at the time of the error. P indicates the next instruction to be executed at the time of the error. S indicates the status at the time of the error.

Recommended action for this error is:

1) RELOAD AND SEE IF THE ERROR IS REPEATABLE
2) IF REPEATABLE RUN RAM05 (PN 2267303-1006-**)

6.2.2 REAL TIME CLOCK INTERRUPTS

This error is caused on a clock interrupt and when the program has gone above or below the test area.

ERROR . THE PC FOR THE CURRENT TEST IS OUT OF ALLOWABLE ADDRESSING AREA.

ADDRESS OF PRIOR AND CURRENT INTERRUPT

CURRENT WP = "WWWW" PC = "PPPP" ST = "SSSS"

PRIOR WP = "WWWW" PC = "PPPP" ST = "SSSS"

The current line tells what the present workspace, program counter, and status is at the time of the error. The prior line tells what the workspace, program counter, and status was for the last clock interrupt at a valid PC.

Recommended action for this error is:

1) RELOAD AND SEE IF REPEATABLE
2) IF REPEATABLE RUN RAM05 (PN 2267303-1006-**)

6.2.3 TEST ERRORS

These errors are displayed when one of the subtests finds an error. They are self explanitory and have a recommended action.

6.2.3.1 REAL TIME CLOCK ERRORS

There are two errors in subtest 1. They are:

ERROR THE REAL TIME CLOCK FAILS TO INTERRUPT
CHECK THE INTERRUPT LOGIC
CHECK THE SWITCHES

RTC DID NOT OCCUR IN WINDOW
CHECK THE POWER SUPPLY FOR STABILITY OR SPIKES

6.2.3.2 INTERRUPT AND XOP ERRORS

There are two errors in subtest 2. There is an additional message that shows all of the registers. The main messages are below:

ERROR INTERRUPT FAILURE
CHECK THE INTERRUPT LOGIC

ERROR XOP FAILURE
CHECK THE MEMORY DATA PATHS
OR REPLACE THE TMS 9900 CHIP

6.2.3.3 REGISTER TEST ERRORS

There are three errors in subtest 3. There is an additional message that shows all of the registers. The main messages are below:

ERROR THE REGISTER TEST HAS AN ERROR
THE WORKSPACE REGISTER IS IN ERROR
CHECK THE MEMORY DATA PATHS
OR REPLACE THE TMS 9900 CHIP

ERROR THE REGISTER TEST HAS AN ERROR
THE PROGRAM COUNTER REGISTER IS IN ERROR
CHECK THE MEMORY DATA PATHS
OR REPLACE THE TMS 9900 CHIP

ERROR THE REGISTER TEST HAS AN ERROR
THE STATUS REGISTER IS IN ERROR
CHECK THE MEMORY DATA PATHS
OR REPLACE THE TMS 9900 CHIP

6.2.3.4 ARITHMETIC LOGIC UNIT TEST

There is one error in subtest 4. There is also an additional message displayed that shows all of the registers. The main message is below:

ERROR ALU ERROR
SUGGEST REPLACING THE TMS 9900 CHIP
OR CHECKING THE MEMORY DATA PATHS

6.2.3.5 INSTRUCTION TEST ERRORS

There is one error in subtest 5. There is also an additional message displayed that shows all of the registers. The main message is below:

ERROR INSTRUCTION ERROR

SUGGEST REPLACING THE TMS 9900 CHIP

OR CHECKING THE DATA PATHS

INSTRUCTION TYPE IN ERROR

FORMAT = "X"

Where the X is the type of instruction in error (see APPENDICES A AND B).

6.2.3.6 COMM CHIP ERRORS

There is one error message in subtest 5. Also being displayed are messages that will give the exact signal in error. The main message is below:

COMM CHIP CRU BASE = "XXXX"

CRU BASE 1700 = LOC. V10

CRU BASE 1740 = LOC. V08

CRU BASE 1780 = LOC. V05

SUGGEST FIRST CHECK THE CLOCK FREQ. INPUT

NEXT CHECK THE CRU LOGIC

FINALLY TRY REPLACING THE CHIP IN ERROR

7.0 PROGRAM CONTROL

This paragraph will explain how to make the diagnostic loop on itself, if it is loaded by DOCS. It will also explain how to loop on an individual subroutine for error checking. If reloading isn't desired the contents of the locations modified should be written down so that they may be restored when fininshed. The operator will have to reference the DIAGNOSTIC HANDBOOK on how to use the .MM verb and the the .GO verb.

LOOPING ON THE ENTIRE TEST

1) CHANGE THE FIRST WORD OF THE TEST TO (X+3DC) WHERE X IS THE LOAD BIAS.
2) USE THE .GO VERB with address (X+3DC)

LOOPING ON SUBROUTINES

1) REAL TIME CLOCK TEST
   a) modify the contents of location (X+47A) from >0202 to >0460
   b) modify the contents of location (X+47C) from (X+6E8) to (X+442)
   c) use the .GO verb back to (X+3DC)
   d) This starts the test over but loops on the subroutine in error. X is the load bias of the test.

2) INTERRUPT AND XOP TEST

a) modify the contents of location (X+4B2) from >0202 to >0460

b) modify the contents of location (X+4B4) from (X+842) to (X+47A)

c) use the .GO verb back to (X+3DC)

d) this starts the test over but loops on the subroutine in error. X is the load bias of the test.

3) REGISTER TEST

a) modify the contents of location (X+4EA) from >0202 to >0460

b) modify the contents of location (X+4EC) from (X+858) to (X+4B2)

c) use the .GO verb back to (X+3DC)

d) This starts the test over but loops on the subroutine in error. X is the load bias of the test.

4) ALU TEST

a) modify the contents of location (X+522) from >0202 to >0460

b) modify the contents of location (X+524) from (aaaa) to (X+4EA) aaaa is an address supplied by the linking procedure

c) use the .GO verb back to (X+3DC)

d) This starts the test over but loops on the subroutine in error. X is the load bias of the test.

5) INSTRUCTION TEST

a) modify the contents of location (X+55A) from >0202 to >0460

b) modify the contents of location (X+55C) from (aaaa) to (X+522) aaaa is an address supplied by the linking procedure

c) use the .GO verb back to (X+3DC)

d) This starts the test over but loops on the subroutine in error. X is the load bias of the test.

6) COMMUNICATION CHIP TEST

a) modify the contents of location (X+592) from >0720 to >0460

b) modify the contents of location (X+594) from (X+01A) to (X+55A)

c) use the .GO verb back to (X+3DC)

d) This starts the test over but loops on the subroutine in error. X is the load bias of the test.

8.0 PART NUMBERS

TITLE	PART NUMBER	
PROGRAM DESCRIPTION	2267307-2001	ROFF SOURCE
	-9001	ROFF OUTPUT
	-9901	PD DOCUMENT
FICHE KIT	2267307-0009	SP
AU05 - LINKED TEST	2267307-1006	FLO
	-7006	LC
	-9006	LML
TEST PARTS		
AU05A	2267314-1003	OBJ
	-2003	SRC
	-9003	LST
AU05B	2267308-1003	OBJ
	-2003	SRC
	-9003	LST

AU05C	2267309-1003	OBJ
	-2003	SRC
	-9003	LST
AU05D	2267310-1003	OBJ
	-2003	SRC
	-9003	OBJ
STANDIO	2267311-1003	OBJ
	-2003	SRC
	-9003	LST
AU05MSG	2267312-1003	OBJ
	-2003	SRC
	-9003	LST

APPENDIX A

FORMAT 1 INSTRUCTIONS

SZC	SET ZEROS CORRESPONDING
SZCB	SET ZEROS CORRESPONDING BYTE
S	SUBTRACT
SB	SUBTRACT BYTE
C	COMPARE
CB	COMPARE BYTE
A	ADD
AB	ADD BYTE
MOV	MOVE
MOVB	MOVE BYTE
SOC	SET ONES CORRESPONDING
SOCB	SET ONES CORRESPONDING BYTE

FORMAT 2 INSTRUCTIONS

JMP	JUMP
JLT	JUMP LESS THAN
JLE	JUMP LESS THAN OR EQUAL
JEQ	JUMP EQUAL
JHE	JUMP HIGH OR EQUAL
JGT	JUMP GREATER THAN
JNE	JUMP NOT EQUAL
JNC	JUMP NO CARRY
JOC	JUMP ON CARRY
JNO	JUMP NO OVERFLOW
JL	JUMP LOW
JH	JUMP HIGH
JOP	JUMP ON PARITY

FORMAT 3 INSTRUCTIONS

COC	COMPARE ONES CORRESPONDING
CZC	COMPARE ZEROS CORRESPONDING
XOR	EXCLUSIVE OR

FORMAT 8 INSTRUCTIONS

LI	LOAD IMMEDIATE
AI	ADD IMMEDIATE
ANDI	AND IMMEDIATE
ORI	OR IMMEDIATE
CI	COMPARE IMMEDIATE

FORMAT 6 INSTRUCTIONS

CLR	CLEAR
NEG	NEGATE
INV	INVERT
INC	INCREMENT BY ONE
INCT	INCREMENT BY TWO
DEC	DECREMENT BY ONE
DECT	DECREMENT BY TWO
SETO	SET TO ONES
SWPB	SWAP BYTE
ABS	ABSOLUTE
B	BRANCH
X	EXECUTE

FORMAT 9 INSTRUCTIONS

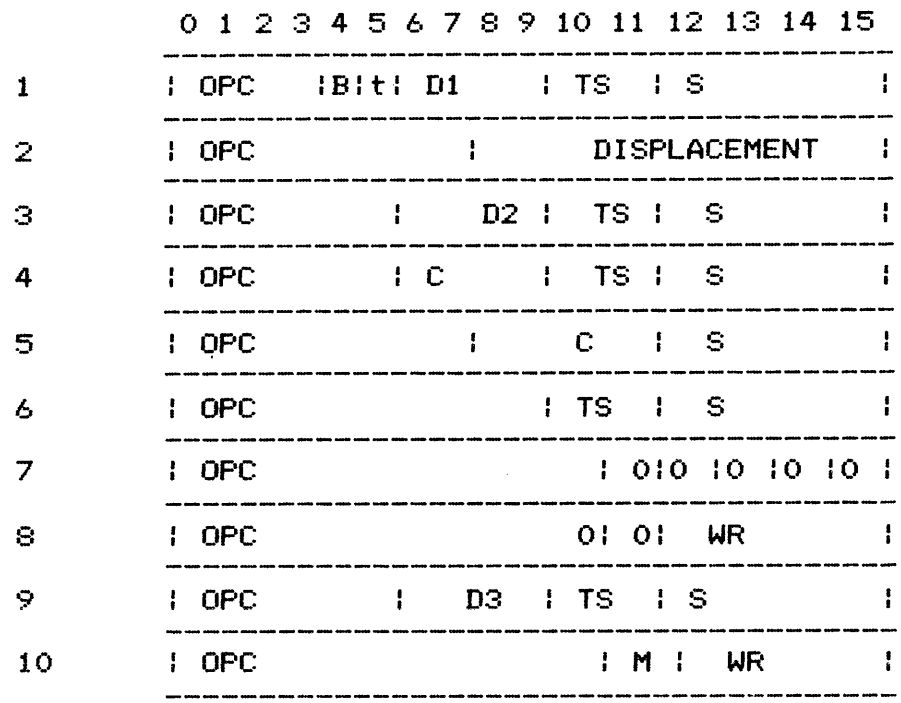
MPY	MULTIPLY
DIV	DIVIDE

INSTRUCTIONS NOT TESTED

LDD	LONG DISTANCE DESTINATION
LDS	LONG DISTANCE SOURCE
LMF	LOAD MAP FILE

APPENDIX B

FORMAT CHART



OPC	OPERATION CODE
B	BYTE INDICATOR
t	ADDREESSING MODE FOR DESTINATION
D1	DESTINATION ADDRESS (ARITH)
D2	WORK SPACE REGISTER (LOGICAL,MULTIPLY,DIVIDE)
D3	EXTENDED OPERATION (XOP)
TS	ADDRESSING MODE FOR SOURCE
S	SOURCE
C	XFR OR SHIFT LENGTH (COUNT)
WR	WORK SPACE REGISTER
M	MAP FIELD

B = 1 MEANS BYTE
TS = 0 DIRECT ADDRESSING
TS = 1 INDIRECT ADDRESSING
TS = 2 SYMBOLIC MEMORY
TS = 2 INDEXED MEMORY
TS = 3 INDIRECT AUTO INCREMENT

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																																		
SHEET																																		
REV STATUS OF SHEETS	REV																																	
	SHEET																																	

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± 010
2 PLACE DECIMAL ± 02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL STD 100

	DATE
DWN	
CHK	
ENGR	
QA	
APVD	
CONTR NO	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

PD, AU10, ARITHMETIC UNIT TEST-990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	937753-9901

SCALE	REV. K	SHEET 1 of 17

TI-0023 C

TABLE OF CONTENTS

1.0 SCOPE

This document describes the 990/10 Arithmetic Unit Test, AU10. This test will operate as a standalone test in the 990/10 computer.

2.0 REFERENCES

For information beyond the scope of this document refer to the following:

PART NUMBER	TITLE
945417-9701	990/10 System Hardware Reference Manual
943441-9701	990 Computer Assembly Language Manual
945400-9701	Diagnostic Handbook

3.0 EQUIPMENT AND SOFTWARE REQUIREMENTS

AU10 requires the following equipment:

a) 990/10 Computer with 8K words of memory.

b) Any of the following I/O devices:

1) ASR/KSR Terminal.

2) 913 VDT Terminal (605 CRT in OOF configuration).

3) 911 VDT Terminal.

4) 810 or similar line printer.

c) Appropriate loading device.

The loadable test module for standard systems is AU10, 937753-1006 (FLO).

Linkable parts: AU10A

AU10B

DSRS

AU10MSG

The loadable test module for Office Systems is AU10OS, 937753-1010 (FLO).

Linkable parts: AU10AOS

AU10B

DSRS

AU10MSG

4.0 LOADING INFORMATION

This is a standalone test and reference should be made to the Diagnostic Handbook for loading procedures from all available media.

5.0 TEST EXECUTION AND DESCRIPTION

The AU10 diagnostic performs the following tests:

1) Real Time Clock test

2) Level 2 Interrupt Test

3) Register Test

4) ALU Test

5) Microcode Test

The primary logic tested by these tests is on the AU1 board. The AU2 board is tested only in that the functioning of the AU2 board is required to support the test operation.

Every effort was made to isolate test functions to a specific subtest to avoid overlap in the subtests. This effort succeeded to a large extent. However, a certain amount of overlap was impossible to prevent. In cases where the overlap of test functions resulted in extensive testing of a particular function, that function is not retested in a later test.

This is especially noticeable in the Microcode test, whereas in the last subtest a high degree of overlap takes place, the Microcode test does not assume a particular function is sufficiently tested unless repeating a test is especially redundant. A good example of this is in testing the interrupt handling microcode,

which is tested extensively in the Level 2 Interrupt test, and not tested in the Microcode test.

The only major area of the 990/10 not tested is the CRU logic. This ommission is due to the inability of hardware to provide a means for the software to have sufficient access to insure correct operation. However, the CRU logic is extensively used in I/O operations and must be functioning for proper operation.

This test also makes no attempt to test mapping logic since it is designed to test both mapped and unmapped systems. Refer to the Diagnostic Handbook for information on the Map Test for the 990/10 computer.

This test is written in such a way that looping on errors is very easy. All branches to error routines are done via the BL instruction. To loop on and error change the word following the BL, which is the address of the error routine, to the address of the desired loop. In this way, instead of branching to an error routine, a branch to any location in the subtest can be taken.

In setting up loops, caution must be used to obtain accurate results. At the beginning of each test except the real time clock test, addressing limits are set. These limits are checked during RTC interrupts to serve

as control against the program setting lost. If an Out-of-limit condition is detected, an error is reported.

5.1 TEST 1 -- REAL TIME CLOCK TEST

This test is designed to test the proper functioning of the Real Time Clock and clock interrupts. In addition, the ability of the AU to interrupt out of idle mode is tested by an interrupt on the Real Time Clock.

The Real Time Clock's stability is tested by determining how many times a loop of code can be executed during a full clock period. This count is used as a standard against repeating the same sequence 1000 times and comparing each clock period's count against the standard. The test will take approximately 8.5 seconds with a 60 Hz line frequency.

5.2 TEST 2 -- LEVEL 2 INTERRUPT TEST

This test will test the level 2 interrupt. All errors that may be caused by software (illop, privop, TILINE timeout) are tested. Memory error is not tested, as this error connot be caused by software. Map errors are tested in the Map test.

The test does very extensive testing of the general interrupt handling capabilities of the AU. The ability

of the test to service different levels of interrupt in a priority level is tested. The masking of interrupts is also tested.

Areas that are tested in this test, eliminating the need for additional testing in other tests, are the privop and the status register privop flags, the XOP instruction and the XOP status register flag.

5.3 TEST 3 -- REGISTER TEST

This test is designed to test the internal registers on the 990/10. The tested registers are the PC, WP, ST, DA, and DD.

The first register tested is the WP register. This test is done by loading the WP with a test value, then doing a BLWP instruction to save the loaded test value for comparison. The test value is then compared to determine if an error has occurred. The test values used are >0000, >FFFE, >AAAA, and >5554.

After completing the WP register test, the ST register is tested. The first part of testing the ST is to set all interrupt masks, then test the remainder of the ST register bits by executing instructions to set and reset all the bits. All the ST register bits are tested,

except for the PRIVOP and XOP bits, which are tested in TEST 2.

The PC is tested by initializing addresses 7000 to 8000 to 0, then doing a BL to those addresses. This should result in a level 2 interrupt. When the interrupt occurs, the trapped PC is compared to the test value PC to check for errors. A final test is done on a PC value of >0000 to insure that all bits will turn off.

As a consequence of testing the WP, ST, and PC registers, the DA and DD registers are tested. This is the only possible test of these register, as there is no direct way for software to to access these registers.

5.4 TEST 4 -- ALU TEST

This test is designed to test the ALU's and the data paths associated with ALU functions. The test treats the 4 ALU's as independent 4 bit units. Add and subtract operations are done on the ALU's with the results compared after each operation. If the result is not equal, one of the ALU's is in error and an ALU error is reported. The data is incremented so that all combinations of 4 bits are tested in each ALU.

The second part of the test performs multiply and divide

operations on the ALU's with all values of data, from 0 to 64K. This part of the test, in addition to testing the ALU's will find and test the longest delay path thru the system.

5.5 TEST 5 -- MICROCODE TEST

This test is designed to exercise all microcode in the 990/10 Computer. The microcode is tested by executing instructions and addressing modes which will cause all microcode states to be executed at least once.

Some microcode states may not be tested in this test for the following reasons:

1) Unable to test due to system restrictions (i.e. CRU instructions).
2) Extensive testing in this test (i.e. interrupts).
3) Extensive testing in another diagnostic (i.e. Map instructions).

All tested microcode states and untested states are documented in the program listing (PN: 937753-9002). In order to determine the cause of an error, the microcode flowcharts are necessary.

6.0 MESSAGES

6.1 ERROR MESSAGES

REGISTER 11 IN THE FOLLOWING PRINTOUT CONTAINS THE RELATIVE ADDRESS OF THE BRANCH TO THE ERROR ROUTINE

This message indicates how to trace the cause of the error.

PGM LISTING ADDR: XXXX

This message tells where the branch to print the messages took place. It will usually be useless in tracing errors.

WORK POINTER PC STATUS AT TIME OF ERROR WP = XXXX PC = XXXX ST = XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

XXXX XXXX XXXX XXXX XXXX XXXX XXXX XXXX

This message is a dump of all registers in use at the time of the error.

ERROR RETURN FROM SUBTEST

This message indicates that an error occurred in the subtest.

ERRORTHE PC FOR THE CURRENT TEST IS OUT OF THE ALLOWABLE ADDRESSING AREA

ADDRESS OF PRIOR AND CURRENT INTERRUPT

CURRENT WP XXXX PC XXXX ST XXXX

PRIOR WP XXXX PC XXXX ST XXXX

This message is displayed when the RTC error check detects an out-of-bounds condition.

RTC DID NOT OCCUR IN WINDOW

This message indicates the RTC interrupt did not occur in the time frame of the first interrupt (± 10).

STATUS REG. AFTER INTER. = XXXX

This message indicates the interrupt mask was incorrectly set upon receipt of the clock interrupt.

ERROR*THE REAL TIME CLOCK FAILS TO INTERRUPT

ERROR*LEVEL 2 INTERRUPT TEST

ERROR*THE REGISTER TEST HAS AN ERROR

ERROR*ALU ERROR

ERROR*MICROCODE ERROR

6.2 HEADER MESSAGES

Start test message:

AU10 990/10 ARITHMETIC UNIT TEST VERSION MM/YY/*R

Note: version is month, year, and revision level.

Start subtest messages:

START SUBTEST 1 - REALTIME CLOCK TEST

START SUBTEST 2 - LEVEL 2 INTERRUPT TEST

START SUBTEST 3 - REGISTER TEST

START SUBTEST 4 - ALU TEST

START SUBTEST 5 - MICROCODE TEST

End of subtest message:

SUBTEST COMPLETE

Count of test loops completed:

LOOPS COMPLETED = XXXX

End of test message:

TEST COMPLETE

7.0 PART NUMBERS

TITLE	PART NUMBER
Program Description	937753-2001 ROFF Source
	-9001 ROFF Output
	-9901 PD Document
	-0009 SP, Microfiche
AU10, Standard Link	-1006 FLO
	-9006 LML
	-7006 LC
AU10OS, O. S. Link	-1010 FLO
	-9010 LML
	-7010 LC
AU10A, Standard Module A	-2002 SRC
	-1002 OBJ
	-9002 LST
AU10AOS, O. S. Module A	-2003 SRC
	-1003 OBJ
	-9003 LST
AU10B, Module B	2261876-2002 SRC
	-1002 OBJ
	-9002 LST

AU10MSG, Messages	2250294-2002 SRC
	-1002 OBJ
	-9002 LST
DSRS, I/O Module	2268486-2003 SRC
	-1003 OBJ
	-9003 LST

PROGRAM DESCRIPTION

FOR AU12

990/12 CPU DIAGNOSTIC

P/N 02268483-9901

REVISION = ORIGINAL 02/79/**

TABLE OF CONTENTS

1.0 SCOPE

This document describes the operation of the 990/12 CPU diagnostic program, AU12. This diagnostic is designed to meet the following goals:

a) To provide factory test facilities with the ability to thouroughly test all features of the 990/12 AU and SMI circuit boards and isolate faults to either board or further to a specific hardware feature.

b) To provide the capability for customer engineers at the remote site to thouroughly test the 990/12 AU and SMI logic and isolate faults to either board.

2.0 REFERENCES

For information beyond the scope of this document refer to the followins:

PART NUMBER	TITLE
0943442-9701	Model 990 Computer Reference Manual
2250077-9701	Model 990/12 Computer Assembly Language Programmer's Guide
0945400-9701	Model 990 Computer Diagnostic Handbook

3.0 EQUIPMENT AND SOFTWARE REQUIREMENTS This section describes the minimum equipment requirements and the available linked object software for the diagnostic.

3.1 EQUIPMENT REQUIREMENTS Refer to the Diagnostics Handbook for a discussion of the equipment requirements for DOCS. In addition to the equipment specified in the Diagnostic Handbook, the following equipment is required:

Model 990/12 computer with 16K words of memory.

3.2 SOFTWARE REQUIREMENTS

This section lists the linked object modules available in the AU12 release. For a list of all linkable parts refer to section 7.0, Part Numbers.

Loadable test modules:	AU12P1,	2268237-1006 (FLO)
	AU12P2,	2268238-1006 (FLO)
	AU12P3,	2268239-1006 (FLO)
	AU12P4,	2268240-1006 (FLO)
	AU12P5,	2268241-1006 (FLO)
	AU12P6,	2268242-1006 (FLO)
	AU12P7,	2268243-1006 (FLO)
	AU12P8,	2268244-1006 (FLO)
	AU12P9,	2268245-1006 (FLO)
	AU12P10,	2268246-1006 (FLO)
	AU12P11,	2268247-1006 (FLO)

4.0 LOADING

Loading procedures from all available media are found in the Diagnostic Handbook.

5.0 TEST EXECUTION AND DESCRIPTION

The AU12 diagnostic is standalone. This means that as soon as it is loaded, it starts executing and displaying messages on the default message devices (a 733 ASR/KSR at CRU base >00, a line printer at CRU base >60, a 913 VDT at CRU base >C0, and a 911 VDT at CRU base >100). No keyboard interaction is allowed. Any operator inputs have to be made by patching memory with a front panel. This diagnostic does not test the following functions:

1) Mapping features (tested in MAP12 diagnostic)
   A) Mapping instructions (LMF,LDS,LDD,SLSP).
   B) Map error, execute violation, write violation
   C) Map registers
2) The LREX instruction.

This diagnostic is very large. There is a requirement that all diagnostics fit in 16K words or less. Therefore AU12 has been divided into eleven separately loadable parts. They all print the same thing when they start. The parts can be distinguished by the header messages that are displayed. When AU12 starts it always displays the following initialization messages:

```
AU12  990/12 CPU DIAGNOSTIC VERSION = 02/06/79
```

RESTART ADDRESS=0088

DISABLE THIS PRINTOUT.(0=DISABLE,-1=ENABLE). ADDR=0048. CURRENT VALUE=FFFF

OUTPUT DEVICE. (0=ASR/KSR,1=913,2=911,3=LP,-1=0,1,AND 2). ADDR=003A

CURRENT VALUE=FFFF.

ASR/KSR BAUD RATE. (0=1200 BD, ELSE 300 BD)ADDR=005A. CURRENT VALUE=0000.

ASR/KSR CRU BASE. ADDR=003C. CURRENT VALUE=0000.

913 CRT CRU BASE. ADDR=000A. CURRENT VALUE=00C0.

911 CRT CRU BASE. ADDR=000C. CURRENT VALUE=0100.

PRINTER CRU BASE. ADDR=000E. CURRENT VALUE=0060.

HEADER MSG ENABLE. (0=NO,1=YES). ADDR=0020. CURRENT VALUE=0001.

ERROR MSG ENABLE. (0=NO,1=YES). ADDR=001E. CURRENT VALUE=0001.

IDLE ON ERRORS. (0=NO,-1=YES). ADDR=0032. CURRENT VALUE=0000.

LINE FREQUENCY. (0=60 HZ,1=50 HZ). ADDR=0008. CURRENT VALUE=0000.

All addresses shown here are displacements from the load point. These messages give the information necessary to patch AU12. To get messages printed on a 733 ASR at CRU base >0000 you just modify address >0046 (header message device) to a >0000 and address >0048 (error message device) to >0000. If a 913 CRT at CRU base >00C0 is to be the output device, change address >0046 to >0002 and address >0048 to >0002. On subsequent loops of the test, this lengthy printout can be disabled by patching address >0054 to >0000. If the 733/743 ASR seems to print very slowly, it may be because the TTY/EIA interface is set for 300 baud. Patching address >0062 (ASR/KSR baud rate)

to >0001, you can make the 733/743 print at normal speed. When all tests are complete, the following message is displayed:

AU12 COMPLETE LOOP COUNT = 0001 ERROR COUNT = 0000

The entire test is then started over. The loop count and error count are not cleared on subsequent loops. Whenever an error occurs, the error code is displayed on the front panel in the lower byte of the display lights and the upper (most significant) byte is all ones. When AU12 is complete and ready to start over, the loop count is displayed on the front panel. In this case the upper byte of the front panel contains zeros.

5.1 AU12 PART 1

This part contains the first quarter of AU board microdiagnostics. A typical printout from running this part should look like this:

<test initialization messages; see 5.0 above>

STARTING FIRST QUARTER OF AU MICRODIAGNOSTICS

FIRST QUARTER OF AU MICRODIAGNOSTICS COMPLETE

AU12 COMPLETE LOOP COUNT = 0001 ERROR COUNT =0000

5.2 AU12 PART 2

This part contains the second quarter of the AU board microdiagnostics. The header messages for this section are:

<test intitialization messages; see 5.0 above>

STARTING SECOND QUARTER OF AU MICRODIAGNOSTICS

SECOND QUARTER OF AU MICRODIAGNOSTICS COMPLETE

AU12 COMPLETE LOOP COUNT=0001 ERRO COUNT=0000

5.3 AU12 PART 3

This part contains the third quarter of AU board microdiagnostics. A typical printout from running this part should look like this:

<test initialization messages; see 5.0 above>

STARTING THIRD QUARTER OF AU MICRODIAGNOSTICS

THIRD QUARTER OF AU MICRODIAGNOSTICS COMPLETE

AU12 COMPLETE LOOP COUNT = 0001 ERROR COUNT =0000

5.4 AU12 PART 4

This part contains the fourth quarter of the AU board microdiagnostics. The header messages for this section are:

<test intitialization messages; see 5.0 above>

STARTING FOURTH QUARTER OF AU MICRODIAGNOSTICS

FOURTH QUARTER OF AU MICRODIAGNOSTICS COMPLETE

AU12 COMPLETE LOOP COUNT=0001 ERRO COUNT=0000

5.5 AU12 PART 5

This part contains the first half of the SMI board microdiagnostics. The header messages from this section look like this:

<test intitialization messages; see 5.0 above>

STARTING FISRT HALF OF SMI BOARD MICRODIAGNOSTICS

FIRST HALF OF SMI BOARD MICRODIAGNOSTICS COMPLETE

AU12 COMPLETE LOOP COUNT = 0001 ERROR COUNT = 0000

5.6 AU12 PART 6

This part contains the second half of the SMI board microdiagnostics. The header messages from this section look like this:

<test intitialization messages; see 5.0 above>

STARTING SECOND HALF OF SMI BOARD MICRODIAGNOSTICS

SECOND HALF OF SMI BOARD MICRODIAGNOSTICS COMPLETE

AU12 COMPLETE LOOP COUNT = 0001 ERROR COUNT = 0000

5.7 AU12 PART 7

This part contains the following tests:

1) Interrupt test for levels 3-15
2) Workspace cache test version 1
3) Workspace cache test version 2
4) Error trace memory test
5) Breakpoint register test
6) Error interrupt register test
7) Priviledged and Illegal operations tests
8) Jump instructions test

The header messages displayed by this part are:

<test initialization messages; see 5.0 above>

STARTING INTERRUPT TEST

INTERRUPT TEST COMPLETE

STARTING WORKSPACE CACHE TEST VERSION 1

WORKSPACE CACHE TEST VERSION 1 COMPLETE

STARTING WORKSPACE CACHE TEST VERSION 2

WORK SPACE CACHE TEST VERSION 2 COMPLETE

STARTING ERROR TRACE MEMORY TEST

ERROR TRACE MEMORY TEST COMPLETE

STARTING BREAKPOINT REGISTER TEST

BREAKPOINT REGISTER TEST COMPLETE

STARTING ERROR STATUS REGISTER TEST

ERROR STATUS REGISTER TEST COMPLETE

STARTING FIRST HALF OF 990/10 INSTRUCTIONS AND HARDWARE FEATURES TEST

FIRST HALF OF 990/10 INSTRUCTIONS AND HARDWARE FEATURES TEST COMPLETE

AU12 COMPLETE LOOP COUNT = 0001 ERROR COUNT = 0000

5.8 AU12 PART 8

This part contains the following tests:

1) Real time clock test

2) 12 ms. clock test

3) MOV instructions test

4) BLWP instructions test

5) All other 990/10 instructions tests

The header messages displayed by this part are:

<test initialization messages; see 5.0 above>

STARTING REAL TIME CLOCK TEST

CLOCK INTERRUPT LEVEL = 0005

TIME BETWEEN RTC PULSES = XXXX UNITS (PLUS OR MINUS >A)

REAL TIME CLOCK TEST COMPLETE

STARTING 12 MS. CLOCK TEST

12 MS. CLOCK TEST COMPLETE

STARTING SECOND HALF OF 990/10 INSTRUCTIONS AND HARDWARE FEATURES TEST

SECOND HALF OF 990/10 INSTRUCTIONS AND HARDWARE FEATURES TEST COMPLETE

AU12 COMPLETE LOOP COUNT = 0001 ERROR COUNT = 0000

5.9 AU12 PART 9

This part contains the first third of the new 990/12 instructions. The following new instructions are tested:

1) BLSK

2) CNTO,RTO,LTO,BIND,EMD

3) CRC,DBC,BDC,DINT,EINT

4) CS,TS

5) INSF,IOF,XV,XF

6) TMB,TSMB,TCMB

The header messages displayed by this part are:

<test initialization messages; see 5.0 above>

STARTING FIRST THIRD OF NEW 990/12 INSTRUCTION SET TEST

FIRST THIRD OF NEW 990/12 INSTRUCTION SET TEST COMPLETE

AU12 COMPLETE LOOP COUNT = 0001 ERROR COUNT = 0000

5.10 AU12 PART 10

This part contains the second third of the new 990/12 instruction tests. The following new instructions are tested:

1) MOVS,MVSR

2) PSHS,POPS

3) MPYS,MOVA

4) AM,SM,ANDM,ORM,XORM

5) SLAM,SRAM

6) SLSL

7) ARJ,SRJ,SWPM

The header messages displayed by this part are:

<test initialization messages; see 5.0 above>

STARTING SECOND THIRD OF NEW 990/12 INSTUCTION SET TEST

SECOND THIRD OF NEW 990/12 INSTUCTION SET TEST COMPLETE

AU12 COMPLETE LOOP COUNT=0001 ERROR COUNT=0000

5.11 AU12 PART 11

This part contains the third third of the new 990/12 instructions. The following new instructions are tested:

1) STD,LD,DD,MD,SD,CID,AD,STR,LR,DR,MR,SR,CIR,AR

2) CER,CED,CDE,CRE,NEGD,NEGR,CDI,CRI,XIT

3) NRM,EP

4) LIM,LST,LWP,STPC

5) SEQB,SNEB

The header messages displayed by this part are:

<test initialization messages; see 5.0 above>

STARTING FIRST THIRD OF NEW 990/12 INSTRUCTION SET TEST

FIRST THIRD OF NEW 990/12 INSTRUCTION SET TEST COMPLETE

AU12 COMPLETE LOOP COUNT = 0001 ERROR COUNT = 0000

5.12 DESCRIPTION OF THE TESTS

This section contains a description of each test in AU12.

5.12.1 AU MICRODIAGNOSTICS

With the Writable Control Store feature on the 990/12, it is possible program in special-purpose, user-defined instructions. This is called micro-coding. The AU12 diagnostic makes use of about 36 special diagnostic 'instructions' that test specific buses and packages on the AU board. If this test fails in the field, then the first board (AU) should be replaced.

5.12.2 SMI MICRODIAGNOSTICS

This test is really an extension of the AU microdiagnostic. There are 24 microdiagnostic instructions that isolate problems to the second board (SMI).

5.12.3 INTERRUPT TEST

This test uses the diagnostic forced interrupt feature to test interrupts level 3-15. Note that the breakpoint register bits 15-3 are tested in the process. The test causes a level 15 interrupt and then adds one interrupt at a time until all 13 interrupts are pending. Then it returns back by clearing each interrupt one at a time. At each level the present context and return context are checked. This checks the priority decoding on interrupts

as well.

5.2.4 WORKSPACE CACHE TEST VERSION 1

This test checks the data integrity of the workspace. (i.e. if I put in something, do I get the same thing out). This test does not really test the workspace cache control logic. This test demonstrates that the workspace is basically usable.

5.12.5 WORKSPACE CACHE TEST VERSION 2

The cache controller is supposed to speed up workspace accesses by keeping the latest version of the registers in fast Bipolar memory. If the cache controller were totally bypassed, the workspace could appear to be working but with too many memory acceses occurring. This test uses Error Trace Membry to examine the memory cycles that occur during different combinations of moves that involve overlapped and non-overlapped registers, valid and non-valid registers, and reads and writes. This is done for each register. Breakpoints are used in conjunction with using the Error Trace Memory.

5.12.6 REAL TIME CLOCK TEST

This test is very much like the RTC test in AU10 and AU04. This test merely verifies that the clock is consistent. The duration of the first pulse is measured and 99 subsequent pulses are checked against it. This test prints headers to tell the operator what the

interrupt level of the RTC is and what the duration of the clock pulse is. This duration is printed out in units of loop counts (i.e the number that AU12 could count to before the next clock pulse occurred).

5.12.7 ERROR TRACE MEMORY TEST

The Error Trace memory feature saves the last 16 memory addresses generated before a level 2 interrupt occurred. This test causes the following types of level 2 interrupts: 1) read breakpoint 2) write breakpoint 3) instruction fetch breakpoint 4) any address breakpoint 5) memory parity error 6) privileged violation 7) illegal opcode 8) TILINE timeout. Each interrupt is checked to make sure it occurred properly and that the entire error trace memory is correct. Many workspace overlap cases occur in this test (including the execution of code within a workspace). The breakpoint is tested that it doesn't occur when it isn't supposed to (i.e. a read access to an address set with a write breakpoint should cause no interrupt). This tests breakpoint control and breakpoint sequencing. This test also checks level 2 interrupt processing.

5.12.8 12 MS. CLOCK RATE TEST

The 12 millisecond clock on the 990/12 is tested by comparing it with the Real Time Clock. Note that this test cannot isolate which clock is in error. It can only

prove that they are proportioanl to each within the specifications. The two clocks are measured simultaneously for 900 milliseconds.

5.12.9 BREAKPOINT REGISTER TEST

This tests the least significant 3 bits of the breakpoint register by setting breakpoints on locations >0000 and >0006. The interrupt test already tested bits 3-15 and the Error Trace Memory test checked the function of the breakpoint register.

5.12.10 ERROR INTERRUPT REGISTER TEST

This test checks that the level 2 error status register at CRU base >1FC0 can be cleared with an SBZ instruction or with a RSET instruction. Bits 15-8 can be cleared with an SBO instruction. An SBO to any of bits 0-7 causes the entire error status register to be set to ones (>FFF0). All of this is checked. SBZ, SBO, and TB instructions are checked in the process.

5.12.11 INSTRUCTION TEST

The AU12 instruction test is table driven. A typical instruction test would proceed like this:

1) Preload all source and destination registers, buffers and memory locations. This previous data is fetched from tables and placed where the instruction can use it. This protects the table from being modified so it will still be the same on

subsequent loops.

2) Preload the status register with all ones (>FEOF) or with all zeros (>0000). The next time the instruction is tested the other value is used. This tests that the proper bits are set to one from a zero and that all the status bits that should be left alone are not set to zero from a one.

3) The instruction is fetched from a table , moved in line with the test code and executed. the resulting status is compared to a value in a table.

4) The source and destination data is compared to values in tables to make sure that the instruction executed correctly.

The string instructions use breakpoints to test the reexecutabilty feature. The IDLE instruction test uses the Real Time Clock to test that you can interrupt out of the idle mode. There is a separate test called 'PRIV' which tests the privileged instructions in non-privileged mode. There is also a test called 'ILOP' which tests all 990/12 illegal opcodes. There are tests called 'AOF' and 'SOF' which test arithmetic overflow interrupt and stack overflow interrupt respectively. These 4 special tests are performed in the instruction test after all other instruction tests have been completed. The instruction test code is written using a subset of the old 990/10

instruction set. The following instructions are not tested in this test:

1) LMF,LDS,LDD,SLSP (tested in MAP12).
2) LDCR,STCR (the CRU instructions are not tested in AU12 but they are used for I/O to devices and to the front panel in AU12. The ROM selftest has some microdiagnostic checking of the CRU).
3) LREX is never tested (not testable in AU12).
4) LCS and hardware XOP are used in the Microdiagnostic Overlays section of AU12 and are not included in the table driven instruction test.
5) The CKON and CKOF instructions are tested in the Real Time Clock test.

6.0 MESSAGES

There are three kinds of messages from AU12:

1) Header messages

2) Error messages

3) Warning messages

A warning message is printed when AU12 detects what can only be be described as a 'software error'. AU12 performs many internal consistancy checks (i.e. checking the range on pointers and flags). If an error of this type is detected, then AU12 cannot know when or how the problem was caused. In all cases AU12 will attempt to continue after such an error.

NOTE

WARNING MESSAGES SHOULD BE TREATED AS ERRORS IN THE FIELD. ONE OR BOTH AU AND SMI BOARDS SHOULD BE REPLACED.

The message devices can be one or all of the following:

1) ASR/KSR

2) 912 CRT

3) 913 CRT

4) Line printer

All messages are printed in non-interrupt driven mode. Each error message is associated with one of the following three messages:

1) 'ANALYSIS: AU BOARD IS FAULTY.'

2) 'ANALYSIS: SMI BOARD IS FAULTY.'

3) 'ANALYSIS: CAN'T TELL WHICH BOARD IS FAULTY'

6.1 HEADER MESSAGES

See sections 5.0 through 5.11 for examples of the header messages printed by AU12.

6.2 ERROR MESSAGES

0001 TB instruction did not set EQ bit to correct value.

0002 MPYS or DIVS instruction result is incorrect.

0003 MPYS or DIVS instruction source operand was modified.

0004 BLSK instruction did not branch to right address.

0005 BLSK instruction top of stack pointer is incorrect.

0006 BLSK instruction status register was modified.

0007 Real time clock never occurred; real time clock test aborted.

0008 SLSL instruction source autoincrement error.

0009 SLSL instruction destination autoincrement error.

000A MOVS checkpoint register incorrect after execution.

000B MOVS instruction source autoincrement incorrect.

000C MOVS instruction destination autoincrement incorrect.

000D not used

000E The return PC was incorrect on breakpoint.

000F BUFFER MISCOMPARE: (buffer description)

ACTUAL BUFFER

XXXX YYYY YYYY YYYY YYYY YYYY YYYY YYYY YYYY .. ..

EXPECTED BUFFER

XXXX YYYY YYYY YYYY YYYY YYYY YYYY YYYY YYYY .. ..

This is called often. The value XXXX is the memory address. The values YYYY are the data at those addresses. The characters to the right of the data are the ASCII characters represented by the data. A period is printed if the data is

non-ASCII.

0010 Interrupt occurred at the wrong place in EINT/DINT instruction test.

0012 SLSL final block address incorrect.

0012 SLSL next to last block address incorrect.

0013 MOVA instruction destination result incorrect.

0014 RSET instruction didn't clear the interrupt mask.

0015 Interrupt mask not=1 in level 2 interrupt handler.

0016 Level 2 interrupt was expected but the wrong type received.

0017 Unexpected level 2 interrupt received.

0018 Unexpected interrupt level 3-15 occurred.

0019 Error in writing ones to workspace registers.

001A Ones in workspace registers not written to memory on context switch

001B Error in writing zeros to registers.

001C Writing ones to odd-numbered registers caused incorrect result in registers.

001D Writing ones to odd-numbered registers caused incorrect result in memory after context switch.

001E Privileged interrupt did not occur.

001F No interrupt occurred during ILOP or AOF or SOF test.

0020 not used

0021 Return PC incorrect during ILOP or SOF or AOF test.

0022 not used

0023 TSMB or TCMB or TMB instruction autoincrement incorrect.

0024 NRM destination result incorrect.

0025 RTWP affected more than status bits 0-6 when in privileged mode.

0026 not used

0027 12 ms. clock rate error.

0028 The error status register could not be cleared properly using the three methods (SBZ,SBO,RSET).

0029 A buffer miscompare occurred where the entire buffer was expected to consist of a single byte of data repeated.

002A Expected breakpoint never occurred.

002B Incorrect return PC during privileged instruction test.

002C RSET instruction didn't clear the clock interrupt.

002D An SBO to one of the bits 0-8 at CRU base >1FC0 did not cause the error status register to be set to >FFF0 for diagnostic purposes.

002E BLSK instruction didn not branch

002F not used

0030 ACTUAL CHECKPOINT REGISTER =XXXX EXPECTED=XXXX.

0031 CRC instruction partial sum incorrect

0032 CRC instruction source autoincrement incorrect.

0033 Actual destination autoincrement REG=XXXX EXPECTED=XXXX (SEQB) instruction).

0034 ACTUAL STATUS=XXXX EXPECTED STATUS=XXXX

0035 XF or XV instruction destination result incorrect.

0036 XF or XV instruction deferred count register (R0) was modified.

0037 ARJ or SRJ jump occurred when it should not.

0038 ARJ or SRJ jump did not occur when it should have.

0039 ARJ or SRJ register is incorrect

003A ARJ or SRJ modified the status register.

003B MOV instruction modified the source address.

003C MOV instruction destination result is incorrect.

003D MOV instruction source autoincrement incorrect.

003E MOV instruction destination autoincrement incorrect.

003F EP instruction incorrect source autoincrement.

0040 EP instruction incorrect destination autoincrement.

0041 BLWP instruction status incorrect.

0042 LST instruction no level 2 interrupt occurred.

0043 Software error: unable to find instruction text in FINDOP buffer.

0044 B or BL instruction did not branch.

0045 BL instruction incorrect link register value.

0046 No interrupt occurred during EINT/DINT instructions test.

0047 Demand fill with pending overlap failed. Prefetched JMP instruction ignored.

0048 Demand fill with pending overlap failed. The MOV did not complete.

0049 not used

004A not used

004B not used

004C Byte string autoincrement failed on a stack instruction.

004D Checkpoint register incorrect after stack instruction.

004E Top of stack pointer is incorrect when TS=0 for PSHS or POPS instructions.

004F Return PC after RTC interrupt incorrect

0050 Return WP after RTC interrupt incorrect

0051 Current WP after RTC interrupt incorrect

0052 not used

0053 TS autoincrement incorrect.

0054 Real time clock is not consistant in duration to within 5%.

0055 MOV instruction S field or D field decode problem.

0056 CRC checkpoint register incorrect.

0057 CRC destination autoincrement incorrect

0058 Real time clock occurred once but not twice. The length of the pulses is indeterminate.

0059 Unexpected software XOP occurred.

005A MOVS instruction failed when the destination register and the checkpoint register are the same..

005B LWP instruction modified the status register.

005C LWP instruction result workspace pointer incorrect.

005D LST instruction resultant status incorrect.

005E MPY or DIV modified the source operand.

005F not used

0060 CNTO, RTO, or LTO result destination incorrect.

0061 CS instruction index register incorrect.

0062 BIND Instruction did not branch.

0063 BIND instruction modidied the status register.

0064 BIND instruction branched to the wrong address.

0065 Breakpoint occurred when none was expected.

0066 The level 3-15 interrupt test failed to reach the level 3 interrupt handler.

0067 SLSL final block address after reexecution from breakpoint is incorrect.

0068 SLSL next to last block address after reexecution from breakpoint is incorrect.

0069 Interrupt 3-15 test; the workspace pointer is incorrect at one of the interrupt levels.

006A Interrupt 3-15 test; the interrupt mask is incorrect at one of the interrupt levels.

006B Interupt 3-15 test; the return workspace pointer is incorrect for one the interrupt levels.

006C Interrupt 3-15 test; the return program counter is incorrect for one of the interrupt levels.

006D Interrupt 3-15 test; the return interrupt mask is incorrect for one of the interrupt levels.

006E Stack descriptor pointer autoincrement on a stack instruction failed.

006F STPC instruction incorrect source result.

0070 SEQB or SNEB instruction source operand modified.

0071 ACTUAL SOURCE AUTOINC REG=XXXX EXPECTED=XXXX (SEQB instruction).

0072 One of the old 990/10 instruction set jump instructions modified the status register.

0073 One of the 990/10 jump instructions jumped when it was not supposed to or didn't jump when it was supposed to.

0074 COC, CZC, or XOR instruction modified the mask.

0075 COC or CZC or XOR instruction destination result incorrect.

0076 Incorrect result from one of the old 990/10 shift instructions.

0077 Incorrect register result for one of the old 990/10 immediate instructions (LI,AI,ANDI,ORI,CI,LIMI,STST).

0078 STWP or LWPI instruction resulted in incorrect WP address.

0079 STWP or LWPI instruction modified the status register.

007A A floating point instruction result is in error.

007B Incorrect destination result for one of the Format 1 instructons (SZC,SZCB,S,SB,C,CB,A,AB,MOV,MOVB,SOC,SOCB).

007C not used

007D MPY or DIV instruction result incorrect.

007E ABS instruction source result incorrect.

007F CLR instruction source result incorrect.

0080 DEC instruction source result incorrect.

0081 DECT instruction source result incorrect.

0082 INC instruction source result incorrect.

0083 INCT instruction source result incorrect.

0084 INV instruction source result incorrect.

0085 NEG instruction source result incorrect.

0086 SETO instruction source result incorrect.

0087 SWPB instruction source result incorrect.

0088 Status not cleared by EMD.

0089 WP changed by EMD.

008A RTWP did not branch to PC.

008B RTWP result status incorrect.

008C RTWP result WP incorrect.

008D EMD did not clear error status latch.

008E OVERLAY FAILURE R0=XXXX R1=XXXX OVERLAY POINTER=XXXX WP=XXXX ST=XXXX.

This message appears when a microdiagnostic fails. The analysis message will always say 'AU BOARD IS FAULTY'. This is true only for AU12 parts 1 through 4 (AU microdiagnostics). For AU12 parts 5 and 6 (SMI microdiagnostics) the recommended action is to replace the SMI board. The microdiagnostic passes back the Overlay Number in R0; this can be looked up in the Fault Dictionary for microdiagnostics. Register one (R1) contains the loop count where the error occurred. The 'OVERLAY POINTER' is merely another way of finding out which overlay failed without relying on the overlay itself telling us. It is only meaningful if AU12 listings are available. The Workspace pointer (WP) and the status register (ST) are printed out because sometimes a microdiagnostic will destroy the return status for AU12.

6.3 WARNING MESSAGES

This section contains the warning messages that are incorporated into AU12. Warnings were originally intended as software checkout aids.

1) 'WARNING: UNKNOWN BUFFER TYPE'

This message is associated with error 000F. It occurs when a table is searched, but a table entry is not found. This should never occur, and if it does happen then it means that the table was destroyed or that the normal flow of program control in AU12 was altered.

2) 'TEST SKIPPED BECAUSE RTC INTERRUPT LEVEL UNKNOWN'

This message is printed in Part 8 during the IDLE, RSET, BLWP, and RTWP tests only if the real time clock interrupt level is unknown. The interrupt level is known from the Real Time Clock test. If the RTC test was unable to cause an RTC interrupt, then the interrupt level for the RTC will be unknown.

3) 'WARNING: OCCUR FLAG NOT CLEARED'

The level 2 interrupt handler has a couple of flags that make sure that every interrupt is accounted for and that no extra interrupts occur. This message should be accompanied by error messages 0016 and/or 0017.

4) 'WARNING: ERR FLAG NOT CLEARED'

See paragraph 6.3.3

7.0 PART NUMBERS

TITLE	NUMBER	
PROGRAM DESCRIPTION	2268483-2001	ROFF Source
	-9001	ROFF Output
	-9901	PD Document
FICHE KIT	-0009	SP

The following dash table applies to the following linked test parts:

-1006	FLO	Fully Linked Object
-7006	LC	Link Control
-9006	LML	Link Map Listing

Linked tests:

AU12P1	2268237-
AU12P2	2268238-
AU12P3	2268239-
AU12P4	2268240-
AU12P5	2268241-
AU12P6	2268242-
AU12P7	2268243-
AU12P8	2268244-
AU12P9	2268245-
AU12P10	2268246-

AU12P11 2268247-

The following dash number table applies to the following test modules and linkable parts:

-1003 OBJ

-2003 SRC

-9003 LIST

Linkable test parts:

AUTENT	- Main Driver	2268248-
BBE	- Inst test	2268249-
BLSK	- Inst test	2268375-
BLWP	- Inst test	2268376-
BRKPNT	- Breakpoint test	2268377-
CACHE	- WP Cache test	2268378-
CKRATE	- 12 ms clock	2268379-
CRCDBDNT	- Inst test	2268380-
CS	- Inst test	2268381-
ERROR	- Output driver	2268382-
FIELD	- Inst test	2268383-
FMT1	- Inst test	2268384-
FMT379	- Inst test	2268385-
FMT5	- Inst test	2268386-
FMT6	- Inst test	2268387-
FMT8	- Inst test	2268388-
FP	- Inst test	2268389-

INIT	- Initialization	2268390-
INSERR	- Error test	2268391-
INSTAB	- Inst Driver	2268392-
INTEST	- Interr. test	2268393-
JMP	- Inst test	2268394-
JUMPSWPM	- Inst test	2268395-
MEM	- Inst test	2268396-
MICROD	- Overlay driver	2268397-
MOVS	- Inst test	2268398-
MPYSMOVA	- Inst test	2268399-
MSGP1	- Part 1 messages	2268400-
MSGP2	- Part 2 messages	2268401-
MSGP3	- Part 3 messages	2268402-
MSGP4	- Part 4 messages	2268403-
MSGP5	- Part 5 messages	2268404-
MSGP6	- Part 6 messages	2268405-
MSGP7	- Part 7 messages	2268406-
MSGP8	- Part 8 messages	2268407-
MSGP9	- Part 9 messages	2268408-
MSGP10	- Part10 messages	2268409-
MSGP11	- Part11 messages	2268410-
MSGMOD	- General Messgs	2268411-
MULT	- Inst test	2268412-
PREC	- Inst test	2268413-
REG	- Inst test	2268414-

RTCMOV - Real time clock 2268415-

SEQB - Inst test 2268416-

SHFT - Inst test 2268417-

SLSL - Inst test 2268418-

STAK - Inst test 2268419-

TRCTST - Error trace mem 2268420-

XFP - Inst test 2268421-

OVRMSG - Overlay messges 2268422-

The following are the microcode modules used in AU12:

USTST 2268423-

CNDTST 2268424-

BUSTST 2268425-

GENREG 2268426-

CNTTST 2268427-

ARTST 2268428-

PRTST 2268429-

CLTST 2268430-

DLTST 2268431-

SCTST 2268432-

SLTST 2268433-

TSTDTST 2268434-

DPTST 2268435-

PCTST 2268436-

MCTST 2268437-

ALU7TST 2268438-

ALU1TST	2268439-
ALU3TST	2268440-
AUL4TST	2268441-
ALU5TST	2268442-
ALU6TST	2268443-
ALU8TST	2268444-
ALU2TST	2268445-
COTST	2268446-
ABUSTST	2268447-
BTPTST	2268448-
MDOTST	2268449-
BYTTST	2268450-
STLTST	2268451-
STATST	2268452-
ALU10TST	2268453-
CRCTST	2268454-
CACHE	2268455-
LNGCNST	2268456-
WCS800	2268457-
EOITST	2268458-
E2TST	2268459-
CRUTST	2268460-
MDUMOD	2268461-
HTSIE	2268462-
BRKPT	2268463-

INT2	2268464-
INT3UP	2268465-
MF0E	2268466-
MF1E	2268467-
MF2E	2268468-
MF3E	2268469-
MF0L	2268470-
MF1L	2268471-
MF2L	2268472-
MF3L	2268473-
MF0B	2268474-
MF1B	2268475-
MF2B	2268476-
MF3B	2268477-
TRACEINT	2268478-
TILINE	2268479-
MEMTRC	2268480-
TLHOLD	2268481-
IORESET	2268482-

8.0 ALGORITHMS

This section contains the algorithms in METACODE for all parts of AU12. Since AU12 is table driven and modular the code follows these algorithms very closely.

```
************
Begin
    Until CONTENTS(LOAD.POINT) EQ ENTRY.POINT Perform AU12
End
************
Procedure AU12                  ;990/12 CPU DIAGNOSTIC
        Begin
            Perform INIT
            Perform MICROD
            Perform INTEST
            Perform CACHE
            Perform NCACHE
            Perform RTC
            Perform TRCTST
            Perform CKRATE
            Perform BRKPNT
            Perform RSTCLR
            Perform INSTAB
        End
************
Procedure INIT                  ;INITIALIZATION ROUTINE
        Begin
            Turn on fault light
            Disable mapping
            Initialize breakpoint
            Set up interrupt traps
            Do short test of load/execute in WCS
                 and reading Trace Memory
            Turn off fault light
            Print first heading
            Print addresses and values for patch locations
        End
************
```

```
************
Procedure MICROD                ;MICRODIAGNOSTICS OVERLAY DRIVER
        Until OVERLAY.TABLE.POINTER EQ END.OF.TABLE Do
            Load OVERLAY(OVERLAY.TABLE.POINTER) into
                 WCS at location >800
            Perform overlay (at location >800)
            Increment OVERLAY.TABLE.POINTER
        End
************
Procedure INTEST                ;LEVELS 3-15 INTERRUPT TEST
        Begin
            Set up interrupt trap vectors
            Until INTERRUPT.NUMBER EQ 2 Do
                Cause interrupt LEVEL(INTERRUPT.NUMBER)
                Check for proper trap                    **CHKINT
                Increment INTERRUPT.NUMBER
            End
        End
************
Procedure CACHE                 ;WORKSPACE CACHE TEST
        Begin
            Clear workspace1 memory
            Load WP with workspace1 pointer
            Clear workspace2 memory
            Write ones to R0-R15 (WP1)
            Compare R0-R15 to ones (WP1)
            Load WP with workspace2 pointer
            Compare WP1 memory to ones
            Compare R0-R15 to zeros (WP2)
            Write ones to odd numbered registers
            Compare odd numbered registers to ones
            Load WP with workspace1 pointer
            Compare memory in WP2 to alternate ones and zeros
        End
************
Procedure NCACHE                ;DEMAND FILL CACHE TEST
        Until REGISTER.TEST.POINTER EQ END Do
            Initialize buffers
            Set breakpoint          **SETBP
            Execute register test(REGISTER.TEST.POINTER)
            Compare Trace buffer    **COMPBF
        End
************
```

```
************
Procedure RTC                   ;REAL TIME CLOCK TEST
        Begin
            Disable TILINE CACHE memory
            Determine clock interrupt level   **DETCIL
            Set up RTC interrupt vector
            Enable interrupts
            Calculate time interval
            Test subsequent trials for stability
        End
************
Procedure TRCTST                ;ERROR TRACE MEMORY TEST
        Until TABLE.POINTER EQ END.OF.TABLE Do
            Set breakpoint          **SETBP
            Test breakpoint         **TESTBP
            Increment TABLE.POINTER
        End
************
Procedure CKRATE                ;12 MS. CLOCK RATE TEST
        Begin
            Get RTC interrupt level
            Set up trap vector
            Get time constant
            Enable 12 ms. clock
            Compare counter to RTC counter
        End
************
Procedure BRKPNT          ;TESTS BITS 0-3 OF INTERRUPT REGISTER
        Begin
            Enable breakpoint interrupts
            Set breakpoint    **SETBP
                 (bits 0-3 Set to zero)
            Test breakpoint   **TESTBP
            Set breakpoint    **SETBP
                 (bits 0-3 Set to one)
            Test breakpoint   **TESTBP
            Set breakpoint    **SETBP
            Reset       (Test that reset clears breakpoint)
            Test breakpoint   **TESTBP
        End
************
```

```
************
Procedure RSTCLR                ;ERROR INTERRUPT LATCH TEST
        Begin
            Disable all interrupts
            Until COUNT1 EQ 15 Do
            Begin
                Execute SBO(COUNT1)
                Test LATCH                (SHOULD BE ALL ONES)
                Until COUNT2 EQ 12 Do
                     Execute TB(COUNT2+3)
                End
            End
        End
************
Procedure INSTAB
        Until I.POINTER EQ IEND Do
            Until SEQUENCE(S.POINTER) EQ -1 Do
                Get S.POINTER
                Perform SEQUENCE(S.POINTER)
                Increment S.POINTER
            End
            Increment I.POINTER
        End
************
Procedure (Instruction Test) ;GENERALIZED INSTRUCTION TEST FORMAT
        Begin
            Get INSTRUCTION.TABLE.POINTER
            Get STATUS.BITS.TABLE.POINTER
            Get PARAMETER.TABLE.POINTER
            Until PARAMETER.TABLE.POINTER EQ END
            Get PARAMETERS (SOURCE, DESTINATION, ETC)
            Execute instruction          **STATCH
                  and check results
        End
************
```

A

APPLICATION	
NEXT ASSY	USED ON
	7506

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

REV

SHEET

REV STATUS OF SHEETS — REV / SHEET

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL STD 100

DWN — DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group *Dallas, Texas*

PD, COMM05, 990/5 COMMUNICATIONS PORT TEST -990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	2267316-9901

SCALE | REV. | SHEET 1 of 26

TI-0923 C

TABLE OF CONTENTS

1.0 SCOPE.

This Program Description defines the operation of COMM05, a diagnostic designed to test the 3 external Communications Ports of the Texas Instruments 990/5 Computer. The details of the test operation are outlined in addition to troubleshooting procedures based on the error indications produced by the test. A comprehensive list of error messages and their meanings is also included. Lastly, the Algorithms defining the COMM05 program organization are presented.

It is assumed that the user has previously run the AU05 Diagnostic to verify that the 990/5 CPU instructions and the communications port's TMS990X internal test mode functions are operating properly.

2.0 DOCUMENTATION

To cause as little duplication as possible, many sections of the HSTSLV documentation reference the AU05 PD. The user needs to have this document for proper failuer correction. For other information beyond the scope of this document reference the following:

TITLE	PART NUMBER
990 COMPUTER REFERENCE MANUAL	943442-9701
990 ASSEMBLY LANGUAGE MANUAL	943441-9701
DIAGNOSTIC HANDBOOK	945400-9701
AU05 PROGRAM DESCRIPTION	2267307-9901

3.0 EQUIPMENT AND SOFTWARE REQUIREMENTS.

This section describes the minimum hardware requirements defining the operating environment for operation of the Diagnostic and the availible linked object modules.

3.1 HARDWARE REQUIREMENTS:

In addition to the equipment described in the diagnostic handbook, the following equipment is required:

A 990/5 Computer with 16K words of memory

EIA Loopback Connector P/N 944550-0001

3.2 SOFTWARE REQUIREMENTS.

The COMM05 Diagnostic is designed to run on the Texas Instruments 990/5 Computer and must operate under the control of DOCS (Diagnostic Operational Control System). The object modules of the subtests listed below are linked together to form the Fully Linked Object module (FLO) that is the COMM05 Diagnostic:

LINKABLE PARTS:

COMMR5 -- Main Control Module

COMALL -- Common Tests to all ports

COM902 -- Tests Unique to the 9902 (Ports 1 & 2)

COM903 -- Tests Unique to the 9903 (Port 3)

COMMSG -- Messages

The fully linked test module is COMM05, 2267316-1006 (FLO).

4.0 LOADING.

Loading procedures from all available media are found in the Diagnostic Handbook (P/N 945400-9701).

5.0 TEST EXECUTION AND DESCRIPTION.

The COMM05 Diagnostic is an Interactive Test. When the diagnostic has been loaded by DOCS, the following message will be output to the specified interactive device:

"COMM05 = 990/5 COMMUNICATON PORT LOOPBACK TEST. VERSION 4/79 **"

5.1 IT Verb -- Initialize Test:

The Diagnostic must be initialized before any tests are performed The 'IT' verb is used to perform this initialization. When the 'IT' verb is executed, it will display an informational message and then ask the operator the operating environment configuration questions that follow. If the displayed default is correct, the operator need only press the return key. If a different parameter is required, the operator should enter that value and then press the return key.

"THIS TEST REQUIRES THE LOOPBACK CONNECTOR P/N 944550-1 TO BE PLACED ON THE 990/5 COMM PORT CONNECTORS ACCORDING TO THE TABLE BELOW:

PORT	CONNECTOR	LOCATION	NOTE:
1	P4	FRONT	THE CONNECTORS ARE LOCATED ON THE
2	P5	MIDDLE	RIGHT SIDE OF THE TOPMOST BOARD.
3	P6	REAR	

AT THE START OF EACH SUBTEST, THE PROPER PORT WILL BE ASKED TO BE LOOPED BACK, FOLLOWED BY A PAUSE. PRESS THE RETURN KEY WHEN READY."

"IS A TMS9903 INSTALLED AT LOC UV05? (DEF = 1 -"
"LINE FREQ = 60HZ (1) OR 50HZ (0)? (DEF = 1) -"
"PORT 3 MODEM CLOCK = INTERNAL (1) OR EXTERNAL (0)? (DEF = 1) -"

5.2 EA Verb -- Execute All Tests:

When this verb is entered, the 4 subtests are executed sequentially. Test 1 will execute without any operator interaction. Tests 2, 3 and 4 will prompt the operator to place the loopback connector on the appropriate port. At the start of each test, a test started header message is output. At the end of each test, a test complete message is also displayed.

NOTE: During the execution of all the tests, a "blip" character is output at frequent intervals to signify to the operator that the test is progressing. This blip is represented by the right arrow character ">". Tests 1, 2 and 3 output 3 blips during their execution, while test 4 outputs 28 blips.

5.3 E1 Verb -- Execute Test 01 (Common Functions):

This test requires no operator interaction. The common internal functions of the 3 ports that do not require the loopback connector are tested. These are primarily the interval timer operation of th TMS9902 & TMS9903 Integrated C rcuits. The interupt functioning of the three ports on the proper 990/5 levels is also checked. The following are the functions tested:

-Timer countdown and resetability.

-Timer interupt generation and reset.

-Timer error overun and reset.

-Timer accuracy with respect to generating consistent intervals.

-Timer accuracy relative to the line frequency (within 6 percent).

-Timer ranges.

This test takes approximately 3 seconds to complete successfully. 3 "blips" (>) are output during the execution of this test.

5.4 E2 Verb -- Execute Test 02 (Port 1 Operation):

This test requires that the loopback connector be placed on the Port 1 input/output connector - P4. This test will verify the basic operational performance of the TMS9902 and it's associated circuitry. The functions tested are:

-Character lengths (5,6,7,8 bits/char).

-Odd and even transmit parity generation for all character patterns.

-Transmission data rates.

-Proper functioning of the testable data set interface signals (RTS, CTS, DSR, DSCH, DSCINT).

This test takes approximately 2 seconds to excute properly and will output 3 blips.

5.5 E3 Verb -- Execute Test 03 (Port 2 Operation):

This test operates identically to Test 3, with the exception that Port 2 is being tested. Thus, the loopback connector is required to be placed on P5.

5.6 E4 Verb -- Execute Test 04 (Port 3 Operation):

This test requires that the loopback connector be placed on the Port 3 interface connector, P6. The operation of the TMS9903 Integrated Circuit is verified as well as it's interface to the Data Set and other support circuitry. If a NO (0) answer is returned in response to the initialization question "IS A TMS9903

INSTALLED AT LOC UV05? (DEF = 1) -", then Test E4 is bypassed completely and control is passed back to DOCS. Since Port 3 has an internal clock generator, the test can be run with the loopback connector which does not provide external clocks as well as with an external Data Set.

The following tests are performed:

-Basic chip in place test.

-Data Set interface circuits (RTS, CTS, DCD, DSR, DTR, SRTS, SDCD, RING, off board communications enable).

-Mode 0 character length operation (5, 6, 7, 8, and 9 bits/char).

-Odd and even parity generation and detection.

-Async data transmission (all 8 bit chars and 4 other patterns).

-Bisync data transmission (all 8 bit chars and 4 other patterns).

-Transmission rate of the internal Port 3 clock generator relative to the line frequency.

This test takes approximately 20 seconds to complete in internal clock mode) and will output 28 blips.

5.7 L1 Verb -- Loop on Test 01

This verb causes Test 01 to be repeated continuously until the operator presses ', CMD, or HELP.

5.8 L2 Verb -- Loop on Test 02

This verb causes Test 02 to be repeated continuously until the operator presses ', CMD, or HELP.

5.9 L3 Verb -- Loop on Test 03

This verb causes Test 03 to be executed continuously until the operator presses ', CMD, or HELP.

5.10 L4 Verb -- Loop on Test 04

This verb causes Test 04 to be executed continuously until the operator presses ', CMD, or HELP.

5.11 PE Verb -- Print Errors.

This verb causes the current error count to be displayed. The current error count is also displayed automatically at the end of each test verb. The error count is reset during execution of the IT verb as well as at the start of execution of the other verbs.

6.0 ERROR MESSAGES.

When an error occurs in the execution of any part of the tests, an error message will be output to the error output device, if the print error flag in DOCS is set. The error message indicates which error has occurred. The error number will also appear on the programmer panel. In addition to the error message number, the Port number under test is also displayed, since an identical error message may originate from any of the 3 ports. In general, six or less error messages will be output as a result of a repetive test section failure, before that test section is aborted and another test section is invoked. For example, if during the 256 character Async data test, there were 6 errors, then the balance of the test is skipped to eliminate having to wait for all 256 errors to occur and be printed. 6 errors are enough to cause a clear error indication

6.1.1 ERROR MESSAGES - TYPE 1.

There are 2 different types of error messages. The first type will state that a specific status signal is either missing or was detected when it was not expected. Thus, there are 2 different error messages that can be associated with a given signal. An example of the 2 different type 1 error messages is:

"*PORT X RBRL MISSING (LISTING = NNNN)."

or;

"*PORT X RBRL UNWANTED (LISTING = NNNN)."

The meaning of the different parts of the error message is:

"*PORT X" refers to the Port under test.

"RBRL" is the name of the tested signal.

"MISSING" means that the signal was expected to be present but was not detected.

"UNWANTED" means that the signal was not supposed to be present but was detected.

"(LISTING = NNNN)" defines the relative memory location from the beginning of the diagnostic of the test of the signal.

Error numbers 1 thru 19 are Type 1 error outputs indicating signals as either Missing or Unwanted. These errors are usually symptomatic of a malfunctioning Data Set circuit if they repeat on a specific port, or an inoperative TMS990X if a multiplicity of errors occurs.

NUMBER (in hex)	SIGNAL NAME AND FUNCTION
1	RCVERR - Summary Receive error
2	RPER - Receive Parity Error
3	ROVER - Receive Overrun
4	RFER - Receive Framing Error
5	RFBD - Receive Full Bit Detect

6	RSBD - Receive Start Bit Detect
7	RIN - Receive Data Input
8	RBINT - Receive Buffer Loaded Interupt
9	XBINT - TRansmit Buffer Loaded Interupt
A	TIMINT - Timer Expired Interupt
B	DSCINT - Data Set Signal Change Interupt
C	RBRL - Receive Buffer Loaded
D	XBRE - Transmit Buffer Emptied
E	XSRE - Transmit Shift Register Emptied
F	TIMERR - Interval Timer Error
10	TIMELP - Interval Timer Expired
11	RTS - Request To Send (Data Set Signal)
12	DSR - Data Set Ready " " "
13	CTS - Clear To Send " " "
14	DSCH - Data Set Status Change
15	FLAG - Any Control Flag Set
16	INT - Summary Interupt
17	DCD - Data Carrier Detect (Data Set Signal)
18	RING - Ring Indicator " " "
19	SDCD - Secondary Data Carrier Detect (Data Set Signal)

NOTE: Signals that are associated with the Data Set have EIA drivers and receivers between the TMS990X IC and the EIA connector. If there are failures concerning these signals, the most likely failure mode is the driver or receiver.

6.1.2 ERROR MESSAGES - TYPE 2.

Type 2 error messages describe failure mechanisms that do not require further explanation:

NUMBER (In Hex)	ERROR MESSAGE TEXT AND REASON
1A	"*PORT X ERROR. 990/5 DID NOT RECEIVE 990X GENERATED INTERRUPT" This message indicates that the diagnostic detected via CRU status that an interrupt was generated but that it was not processed by the 990/5.
1B	"*PORT X ERROR. UNEXPECTED INTERRUPT RECEIVED." This message indicates that an unexpected interrupt was received on the interrupt level under test during a test period when the port under test was not supposed to be generating an interrupt.
1C	"*PORT X ERROR. INTERVAL TIMER ERROR, TIMER REG = ZZ." This message indicates that the TMS990X interval timer did not generate a consistent interval period for 50 consecutive tests.
1D	"*PORT X CHAR LENGTH ERROR. XMIT = XX, RECV = YY." this error indicates that the TMS990X is not operating properly.

1E "*PORT X XMIT RATE ERROR. RATE REG = XXXX."

This error indicates that the TMS9902 internal clock rate is not within the proper tolerance.

1F "*PORT X EVEN PARITY ERROR. XMIT CHAR = XX."

This error indicates that the TMS9902 did not generate the proper even parity on transmit. Character with the wrong parity = XX.

20 "*PORT X ODD PARITY ERROR. XMIT CHAR = XX."

Same as above except odd parity.

21 "*PORT X DATA ERROR. XMIT = XX, RECV = YY."

This error indicates that the received data did not agree with the transmitted data.

22 "*PORT X TIMER ERROR (LINE FREQ ACCURACY TEST)."

This error indicates that the interval timer accuracy was not within a 6 % tolerance with respect to the line frequency. Note that if the timer is operating properly and the line frequency is in error, than this error output will be given.

23 "*PORT X TIMER ERROR (TEST MODE TEST)."

This error indicates that the interval timer did not go 32 times faster in the test mode.

24 "*PORT X DATA ERROR. (MODE0, ASYNC, BISYNC).

This error indicates that the received data did not agree with the transmitted data.

25 "*PORT X STATUS ERROR. (MODE0, ASYNC, BISYNC).

This error indicates that a receive status error was detected (parity, overrun or framing).

26 "*PORT X RECEIVE TIME OUT ERROR. (MODE0, ASYNC, BISYNC).

This error indicates that there were no characters received. (Normally due to an open connection in the loopback or Data Set signal path).

27 "9903 CHIP IN TEST FAILED."

This error indicates that the basic test of the TMS990. failed and it is likely that the chip is missing or dead. It is the first test made of port 3 and will be the first error output.

28 "*PORT 3 INTERNAL CLOCK RATE ERROR (RATE = ZZZZ)."

This error indicates that the 990/5 internal clock generator was not within 6 % of the line frequency.

6.2 ERROR ANALYSIS.

It is possible for a specific type of error to occur on one port and not another, since all 3 ports are usually tested for the same functions. Thus an error message can be generated by any of the 3 ports. Since the primary functions that are being tested are the TMS9902 and TMS9903 multifunction integrated circuits, it is common that an IC will either pass the test with no errors or fail just about all the tests. Thus if a test is run and only a few errors are logged, then the loopback circuitry as well as the 990/5 is suspect.

If the loopback function is missing, then there will be a multiplicity of errors.

If there are intermittant data errors, then it is likely that a Data Set driver or receiver is faulty.

If the same signal is repeatedly missing, try swapping the loopback function to another port to see if the error follows and thus can be attributed to the loopback function.

If a signal is repeatedly in error, then it is likely that the Data Set interface circuitry is faulty, rather then the TMS990X Integrated Circuit (See note after signal errors).

7.0 PART NUMBERS.

COMM05:	2267316-9901	PD, PROGRAM DESCRIPTION
	-2001	DATA, PROGRAM DESCRIPTION ROFF SOURCE
	-9001	PD, ROFF OUTPUT FILE OF PD
	-0009	SP, MICROFICHE LISTING KIT
	-9006	LML, LINK MAP LISTING
	-7006	LC, LINK CONTROL
	-1006	FLO, FULLY LINKED OBJECT
COMMR5:	2267316-9003	LIST, ASSEMBLY LISTING
	-2003	SRC, SOURCE
	-1003	OBJ, OBJECT
COMALL:	2267317-9003	LIST, ASSEMBLY LISTING
	-2003	SRC, SOURCE
	-1006	OBJ, OBJECT
COM902:	2267318-9003	LIST, ASSEMBLY LISTING
	-2003	SRC, SOURCE
	-1006	OBJ, OBJECT
COM903:	2267319-9003	LIST, ASSEMBLY LISTING
	-2003	SRC, SOURCE
	-1006	OBJ, OBJECT
COMMSG:	2267320-9006	LIST, ASSEMBLY LISTING
	-2003	SRC, SOURCE
	-1006	OBJ, OBJECT

8.0 METACODE PROGRAM DESCRIPTION.

This section contains the algorithms for the verbs and subroutines that comprise the COMM05 test. The algorithms are written in Metacode.

Global variables:

```
      TSTCRU - CRU base address for the port under test.
      TSTLVL - Interupt level for the port under test.
      LINE60 - Line frequency is 60 HZ, else 50 HZ.
      IN903  - TMS9903 IC is installed, test Port 3.
      CLKINT - Test port 3 using the internal 990/5 clock generator.
      TSTPRT - Port under test.
```

IT Verb - Initialize Test.

```
Local variables:
    EALL - Execute All Tests
    ERRCNT- Count of test errors

*----
Begin
  Clear EALL,ERRCNT
  Turn off front panel fault light
  Output header message "THIS TEST REQUIRES ..ETC"          HMOUT
  Request IN903                                             SSIRP
  Convert to ASCII and update message
  Initialize TMS9903 IC                                     INIT03
  Request LINE60                                            SSIRP
  Convert to ASCII and update message
  Request CLKINT                                            SSIRP
  Convert to ASCII and update message
  Return to VERB                                            VDENT
End
*----
```

EA Verb - Execute All Tests.

```
Begin
  Set EALL
  Clear ERRCNT
  Perform E1
  Perform E2
  Perform E3
  Perform E4
  Output "ALL TESTS COMPLETE"                                    HMOUT
  Clear EALL
  Return to VERB
End
```

```
E1/L1 Verb - Test 01 -- Common Functions.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Local variables:
    LPCNTB - Count of test loops
    LOOPT1 - Loop test 1 flag

-----
L1
Begin
  Clear LPCNTB
  Set LOOPT1
  Go to E11
End
E1
Begin
  Clear LOOPT1
E11
  Output header - "SUBTEST 1 STARTED"                            HMOUT
E11A
  Set TSTPRT,PRTNUM = 1
  Set TSTCRU for port 1 cru address
  Set TSTLVL for port 1 interrupt level
E12
  Perform common function test                                   COMALL
  If TSTPRT = 1
    then Begin
      Set TSTPRT,PRTNUM = 2
      Set TSTCRU for port 2 cru address
      Set TSTLVL for port 2 interrupt level
      Go to E12
    End
  If TSTPRT = 2
    then Begin
      Set TSTPRT,PRTNUM = 3
      Set TSTCRU for port 3 cru address
      Set TSTLVL for port 3 interrupt level
      Go to E12
    End
  Else output trailer - "SUBTEST 1 COMPLETE"                     HMOUT
  Output error count                                             PECALL
  If EALL .NE. 0
    Then Return
    Else If LOOPT1 .NE. 0
      Then update and output loopcount                           LOOP
      Go to E11A
    End
  Return to VERB                                                 VDENT
End
-----
```

```
COMALL - Basic Common Function Test.
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
-----
Begin
  Move TSTCRU to R12
  Output blip message                                          HMOUT
  Test basic functions                                         TBASIC
  Test interval timer functions                                CKINT
  Test interval timer accuracy                                 CKACUR
  Test interval timer ranges                                   CKTIM
  Test interval timer test mode                                CKTSTM
  Return
End
-----

E2 Verb - Test 02 -- Port 1 (9902).
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

LOOPT2 - Loop test 2 flag

-----
L2
Begin
  Clear LPCNTB
  Set LOOPT2
  Go to E21
End
E2
Begin
  Clear LOOPT2
E21
  Set TSTPRT,PRTNUM = 1
  Set TSTCRU = port 1 cru address
  Set TSTLVL = port 1 interrupt level
  Output "SUBTEST 2 STARTED"                                   HMOUT
  Request operator to install loopback conn                    ITVERB
E22
  Perform test 1                                               COM902
  Output "SUBTEST 2 COMPLETE"                                  HMOUT
  Output error count                                           PECALL
  If EALL .NE. 0
    Then Return
  Else If LOOPT2 .NE. 0
    Then Update and output loopcount                           LOOP
    Go to E22
    End
  Return to  VERB                                              VDENT
End
-----
```

COM902 - 9902 Test.

```
Begin
  Perform Initialization                                    INIT02
  Set up 990/5 communications port                          COMOFF
  Test character lengths                                    TSTLEN
  Output "blip"                                             HMOUT
  Test data rates                                           TSTRAT
  Output "blip"                                             HMOUT
  Test Data Set interface                                   TSTDCS
  Output "blip"                                             HMOUT
  Test parity circuits                                      PARITY
  Return
End
```

E3/L3 Verb - Test 03 -- Port 2 (9902).

This verb is identical to E2/L2 with the exeception that Port 2 parameters are used.

E4 Verb - Test 04 -- Port 3 (9903).

This verb operates similarly to E2/L2 with the exception that Port 3 parameters are used and the test of the port is accomplished by the subroutine COM903.

COM903 - Test Port 3 -9903.

```
Begin
  Test for chip presence                                    CHIPIN
  Initialize the 9903                                       INIT
  Test the Data Set interface                               TSTMDM
  Output "blip"                                             HMOUT
  Set baud rate = 4800
  Test character lengths                                    TLEN
  Output "blip"                                             HMOUT
  Test asynchronous mode                                    TSTASY
  Output "blip"                                             HMOUT
  Test bisync mode                                          TSTCOP
  Output "blip"                                             HMOUT
  Test internal clock rates                                 TSTCLK
  Return to caller
End
```

APPENDIX I - EIA/RS-232 LOOPBACK SIGNALS.

The loopback connector (P/N 945550-001) provides the following connections to the 25 pin EIA Output/Input Signal path:

PINS CONNECTED	FUNCTION		
2 -> 3	Transmit Data	->	Receive Data
4 -> 5	Request To Send	->	Clear To Send
4 -> 8	Request To Send	->	Data Carrier Detect
20 -> 6	Data Terminal Ready	->	Data Set Ready
11 -> 12	Secondary RTS	->	Secondary DCD
14 -> 22	"	->	Ring Indicator

These signals can be looped back using manual jumpers or a Data Set if the loopback connector is not availible.

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																																
SHEET																															
REV STATUS OF SHEETS	REV																															
	SHEET																															

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± .010
2 PLACE DECIMAL ± .02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD 100

DWN	DATE
CHK	
ENGR	
QA	
APVD	
CONTR NO.	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

PD, CRCOMM, COMMUNICATIONS INTERFACE MODULE TEST-990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	2250100-9901

SCALE	REV. A	SHEET 1 of 19

TI-9923-C

CRCOMM

COMMUNICATIONS INTERFACE MODULE TEST-990

PROGRAM DESCRIPTION

2250100-9901

TABLE OF CONTENTS

1.0 SCOPE

This document describes the Communications Interface Module Test (2250100). This module is designed to operate under DOCS (Diagnostic Operational Control System). The test is designed to assure full performance of and/or diagnose problems associated with the COMMUNICATION INTERFACE MODULE (PN 946105). The subgroup of tests to be performed is indirectly under the control of the operator in that the operator specifies the testing mode of the module. The module may be tested with or without a modem (synchronous or asynchronous) attached. Those tests which cannot be meaningfully performed without either the test connector or modem attached are skipped and so indicate.

2.0 REFERENCES

2.1 DOCUMENTATION

For information beyond the scope of this document reference the following:

PART NUMBER	TITILE
945410-9701	Model 990 Computer Communication Interface Module Depot Maintenance Manual
945409-9701	Model 990 Computer Communication System Installation and Operation
945400-9701	Diagnostic Handbook
946109-9901	Spec - Comm Interface Module

2.2 TEST MODULE

CRCOMM LINK 2250100-1006

LINKED PARTS: CRCOMM
COMSUB
TSTPK1
TSTPK2
TSTPK3
TSTPK4
CRCMMSG

3.0 EQUIPMENT AND SOFTWARE REQUIREMENTS

CRCOMM runs on a computer system with the following requirements:(1) 990 computer with 12K of memory,(2) an appropriate interactive device,(3) test connector, part number 948550-0001 (optional), and (4) communications interface module (either standard interface or the 2741 interface).

CRCOMM runs under the DOCS (Diagnostic Operational Control System). Refer to Section III of Diagnostic Handbook for a detailed explanation of the DOCS versions.

4.0 LOADING

Loading procedures for all media are contained in the Diagnostic Handbook (945400-9701).

5.0 TEST EXECUTION AND DESCRIPTION

CRCOMM consists of 22 tests and supplies the verbs listed below:

IT - Initialize the program

ET - Execute Test

EA - Execute All Tests

LT - Loop on Test

LA - Loop on All Tests

DS - Display Status

SP - Select Pattern (Used by Test 06)

5.1 TEST INITIALIZATION (IT VERB)

In addition to the questions required by the DOCS, CRCOMM also asks the questions listed below:

ENTER COMM INTERFACE CRU BASE DEFAULT =0140

LOOP BACK CONNECTOR INSTALLED? DEFAULT =01

ENTER COMM INTERFACE INTERRUPT LEVEL(>02)DEFAULT =09

IS COMM INTERFACE CONNECTED TO A MODEM?DEFAULT =01

IS THE MODEM SYNCHRONOUS? DEFAULT =00

IS THIS COMM INTERFACE A 2741? DEFAULT =00

ID SWITCH VALUE? DEFAULT =007F

IS THE COMPUTER LINE CURRENT 60 HZ? DEFAULT =01

5.2 EXECUTE TEST (ET VERB)

After ET verb is entered, the operator is asked to enter a test number between 1 and 16 in hexadecimals.

5.2.1 TEST 1

This test toggles RES MODEM LD OUT (OUTPUT WORD 4 - Bit 3) and SYNC MODE (OUTPUT WORD 3 - Bit 5) and tests the corresponding input bits for agreement.

5.2.2 TEST 2

This test sets DTR and then toggles SRTS to determine if

SDCD toggles in agreement. Then it checks to see if NSF (New Status Flag) and INTSUM are set and cleared properly by this condition.

5.2.3 TEST 3

This test checks the 250 millisecond timer (TIMEXP) for proper setting and clearing and checks the timing to within the design tolerance. It also checks INTSUM for proper setting and clearing.

5.2.4 TEST 4

This test reads the ID switches on the Module and prints no message (except TEST 4 COMPLETE) unless the switches are set to other than >7F (that is switches 1-7 all closed ("ON")). In the event there is a discrepancy, it prints the switch ID as read and indicates an error.

5.2.5 TEST 5

This test checks to ensure that the interrupt logic on the Module properly interrupts the processor when enabled, and not otherwise.

5.2.6 TEST 6

This test writes an alternating pattern of bytes to OUTPUT WORDS 2 and 3, strobing each to the ASTRO. It then reads the ASTRO registers thus written back through the I/F to verify correct operation. Then it swaps bytes and repeats process.

5.2.7 TEST 7

This test writes a "WALKING 1" pattern to OUTPUT WORDS 2 and 3, strobing each to the ASTRO. It then reads the ASTRO registers thus written back through the I/F to verify correct operation. Then it writes a "WALKING 0" pattern and repeats until a "1" and a "0" have been walked through all 8 bit positions.

5.2.8 TEST 8

This test checks the correct operation of the 1 millisecond pulse used as Carrier Detect Reset on an asynchronous modem and Newsync on a synchronous modem. The PULSED MODEM LD (OUTPUT WORD 4 - Bit 4) bit is toggled various times to check proper setting of the RING INDICATOR input and timing is checked to within the tolerances.

5.2.9 TEST 9

This test checks the CDR/NEWSYNC EIA driver and the RING EIA receiver by toggling PULSED MODEM LD and checking RING for agreement.

5.2.10 TEST A

This test toggles DTR and checks the toggleing of DSR. It also checks to see if DSR setting causes NSF to set, when enabled.

5.2.11 TEST B

This test toggles RTS and checks the toggling of CTS and DCD. It also checks to see if DCD setting causes NSF to set, when enabled.

5.2.12 TEST C

This test toggles SRTS and checks the toggling of SDCD. It also checks to see if SDCD changes cause NSF to set, when enabled.

5.2.13 TEST D

This test sets up the conditions which should cause WRQ to set, and then checks to see if WRQ sets when enabled, and resets when disabled. It also checks to see if

INTSUM is set when WRQ=1.

5.2.14 TEST E

This test enables the half-duplex bit (HALF-DUPLEX) which masks the receive data to a space and verifies that no RRQ interrupts are generated.

5.2.15 TEST F

This test forces a line break condition (transmitted data a constant space) and tests proper operation of the BREAK bit. Since the breaking condition also forces a parity error and framing error, these bits (PE and FE) and RCVERRSUM are also tested for proper setting.

5.2.16 TEST 10

This test performs a 1 byte data transfer in SELF-TEST, ASYNCHRONOUS mode to verify data transfer capability.

5.2.17 TEST 11

This test sets up a data transfer, but with RRQ disabled, thus forcing an OVERRUN condition. It tests for the OVERRUN bit being set and RCVERRSUM being set. It then enables RRQ and checks to see if RRQ goes from a

zero to a one. It also receives the character transmitted and verifies that the transfer was done correctly.

5.2.18 TEST 12

This test writes the ASTRO'S SYNC and DLE registers, and then uses the transparent idle fill feature to transmit a DLE/SYNC sequence. This constitutes a transmitter UNDERRUN condition; so, the UNDERRUN bit is tested for setting. Then the receiver is enabled and the DLE/SYNC sequence's proper transmission and reception is verified.

5.2.19 TEST 13

This test performs a series of 200-byte data transfers in SELF-TEST, SYNCHRONOUS mode to verify data transfer capability. The transfer is performed 4 times and must be executed successfully (no error conditions) all four times in order to pass.

5.2.20 TEST 14

This test performs a series of 200-byte data transfer in ASYNCHRONOUS mode, and at a different baud rate from TEST 12. Other test parameters are essentially the same

as in TEST 12.

5.2.21 TEST 15

This test performs a series of 200-byte data transfers in SYNCHRONOUS mode, and at a different baud rate from TEST 13. Other test parameters are essentially the same as in TEST 13.

NOTE: Tests 13 and 15 use a clock-recovery method for synchronizing the data (unless a synchronous modem is used). Thus although the data format is synchronous the clocking scheme is essentially asynchronous.

5.2.22 TEST 16

This test performs a series of 200-byte data transfer in SELF-TEST, SYNCHRONOUS mode to verify data transfer capability. The transfer is performed 4 times without resetting the module to verify that the module strips synchronous between blocks.

NOTE: Some tests are performed several times, but if the test fails, any remaining loops are bypassed and the next test is executed. This eliminates the same error message being printed 4 times or more in a test.

6.0 MESSAGES

In addition to the messages for verb 'IT' (see para 5.1), CRCOMM also displays messages for other verbs listed below and error messages (see para 6.1).

'TEST ** TEST NOT AVAILABLE **'

'PAT1 ='

'PAT2 ='

'*** STATUS ***'

'IW0 ='

6.1 ERROR MESSAGES

The following error messages may be printed out and are considered self-explanatory in all cases, assuming some familiarity with the Module specification. The natation IWm-n implies Input Word m, bit n; and OWm-n implies Output Word m, bit n, as defined in said specification.

NO.	MESSAGE
01	RSVOUT NOT SET (IW1-3) ERROR
02	RSVOUT NOT RESET (IW1-3) ERROR
03	MODE NOT SET (IW1-9) ERROR
04	MODE NOT RESET (IW0-9) ERROR
05	MODE NOT RESET (IW1-9) ERROR
06	MODE NOT RESET (IW0-9) ERROR

07 SDCD NOT SET (IW1-2) ERROR

08 NSF SET WHEN NOT ENABLED (IW1-12) ERROR

09 SDCD NOT RESET (IW1-2) ERROR

0A NSF NOT BEING SET (IW1-12) ERROR

0B INTR SUM NOT SET (IW1-14) ERROR

0C NSF NOT RESET (IW1-12) ERROR

0D INTR SUM NOT RESET (IW1-14) ERROR

0F TIMEXP NOT RESET (IW1-13) ERROR

10 TIMER TOO SHORT (IW1-13) ERROR

11 TIMER NOT BEING SET (IW1-13) ERROR

12 TIMER TOO LONG (IW1-13) ERROR

13 TIMER TOO LONG AFTER RETRIGGER (IW1-13) ERROR

14 TIMER SHORT AFTER RETRIGGER

15 INTERRUPT SUMMARY NOT SET BY TIMEXP (IW1-14) ERROR

16 ID SWITCH READ XXXX

EXPECTED XXXX (IW4-0-6) ERROR

17 UNEXPECTED INTERRUPT AT LEVEL XXXX ERROR

18 UNEXPECTED INTERRUPT FROM COMM INTERFACE ERROR

19 COMM INTERFACE DOES NOT INTERRUPT CPU ERROR

1A RING NOT SET (IW1-4) ERROR

1B RING NOT RESET (IW1-4) ERROR

1C PMODLD TIMING LONG ERROR (OW4-4)

1D PMODLD TIMING SHORT ERROR (OW4-4)

1E

	WROTE	READ	XOR
OW2	XXXX	XXXX	ZZZZ

```
        OW3     XXXX     XXXX    ZZZZ
1F      *** ERROR RWBUSY NOT RESET
20      DSR NOT BEING SET (IW1-0) ERROR
21      DSR NOT RESET (IW1-0) ERROR
22      CTS NOT BEING SET (IW1-0) ERROR
23      DCD NOT RESET (IW1-5) ERROR
24      CTS NOT RESET (IW1-1) ERROR
25      DCD NOT RESET (IW1-5) ERROR
26      SDCD NOT BEING SET (IW1-2) ERROR
27      WRQ SET WHEN NOT ENABLED (IW1-15) ERROR
28      WRQ NOT BEING SET (IW1-15) ERROR
29      RRQ SET WHEN NOT ENABLED (IW1-8) ERROR
2A      OVERRUN NOT SET (IW0-14) ERROR
2B      UNEXPECTED FRAMING ERROR (IW1-13) ERROR
2C      UNEXPECTED PARITY ERROR (IW0-12) ERROR
2D      XMIT DATA = XX  ADDRESS = aaaa
        RCV  DATA = XX  ADDRESS = bbbb
        ERROR COUNT = YY
2E      *** ERROR IN DATA TRANSFER ***
        IW0 = XXXX
        IW1 = XXXX
        IW2 = XX
        IW3 = XX
        XMIT BUFF ADDRESS = XXXX BYTE CNT = XXXX
        RCV  BUFF ADDRESS = XXXX BYTE CNT = XXXX
```

2F	RRQ GENERATED WHEN HALF DUPLEX SELECTED (OW4-6) ERROR
30	RECVERR NOT BEING SET (IWO-15) ERROR
31	PE NOT BEING SET (IWO-12) UPON BREAK (OW4-6) ERROR
32	FE NOT BEING SET (IWO-12) UPON BREAK (OW4-6) ERROR
33	*** TIMED OUT ***

7.0 PART NUMBERS

FICHE KIT	2250100-0009 (SP)
CRCOMM, Linked Test	-1006 (FLO)
	-9006 (LML)
	-7006 (LC)
CRCOMM, Main Test Module	-1003 (OBJ)
	-9003 (LIST)
	-2003 (SRC)
COMSUB, Subroutines	0939485-1001 (OBJ)
	-9001 (LIST)
	-2001 (SRC)
TSTPK1, Test Package 1	0939486-1001 (OBJ)
	-9001 (LIST)
	-2001 (SRC)
TSTPK2, Test Package 2	0939487-1001 (OBJ)
	-9001 (LIST)
	-2001 (SRC)
TSTPK3, Test Package 3	0939488-1001 (OBJ)
	-9001 (LIST)
	-2001 (SRC)
TSTPK4, Test Package 4	0939489-1001 (OBJ)
	-9001 (LIST)

	-2001 (SRC)
CRCMMSG, Test Message	2250264-1001 (OBJ)
	-9001 (LIST)
	-2001 (SRC)

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																														
SHEET																														

REV STATUS OF SHEETS	REV																														
	SHEET																														

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ± 1°
3 PLACE DECIMAL ± 010
2 PLACE DECIMAL ± 02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL STD 100

	DWN	DATE
	CHK	
	ENGR	
	QA	
	APVD	
	CONTR NO	
	DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

PD, DDFLOP, TILINE FLOPPY DISK TEST-990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	2250139-9901

SCALE	REV. C	SHEET 1 of 62

TI-0003 C

TABLE OF CONTENTS

1.0 SCOPE

This document describes the TILINE floppy disk test (DDFLOP). The structure of the test is such that the operator may run the complete test or separate portions. The test has some utility routines which aid the operator in troubleshooting the TILINE floppy problems.

There are five test modules, TFCONT1 - TFCONT5 and a message module TFMSGS, in addition to the DOCS modules.

2.0 REFERENCES

For information beyond the scope of this document refer to the following:

PART NUMBER	TITLE
0943442-9701	MODEL 990 COMPUTER REFERENCE MANUAL
0943441-9701	MODEL 990 ASSEMBLY LANGUAGE MANUAL
0945400-9701	MODEL 990 COMPUTER DIAGNOSTIC HANDBOOK
2261897-9901	PROGRAM DESCRIPTION, DOCS
2261639-9901	SPECIFICATION FOR TILINE FLOPPY CONTROLLER
2261886-9701	MODEL FD1000 FLEXIBLE DISK SYSTEM INSTALLATION AND OPERATION

3.0 EQUIPMENT AND SOFTWARE REQUIREMENTS

This section describes the minimum equipment requirements and the available linked object software for each of the test versions.

3.1 EQUIPMENT REQUIREMENTS

In addition to the equipment specified in the Diagnostic Handbook, the following equipment is required:

a) TILINE Floppy Kit P/N 2261685-0001

b) DD Diskettes P/N 2261687-0001

(1 write protected,1 nonprotected)

c) SD Diskette P/N 0945965-0001

(1 nonprotected)

3.2 SOFTWARE REQUIREMENTS

The following describes the linked object modules available in the DDFLOP release:

DDFLOP linked test containing:

TFCONT1

TFCONT2

TFCONT3

TFCONT4

TFCONT5

TFMSGS

DDFLOP burn in link containins:

TFCONT1

TFCONT2

TFMSGS

4.0 LOADING

Loading procedures for all available media are found in the Diagnostics Handbook or the DOCS Program Description (PD).

5.0 TEST EXECUTION AND DESCRIPTION

The TILINE floppy diagnostic (DDFLOP) tests the TILINE floppy controller (TFC) and the associated floppy drive units. DDFLOP is composed of the following parts:

PART #	FUNCTION
1	Functional check of the controller
2	Random read and head motion
3	Addressing of all tracks and sectors read after write
4	Zeros, ones, and worst case data transfer
5	Command timeout, seek incomplete, fill zero sector, and busy flag test
6	Head switch test for dual density or track switch for single density and head load stabilization test
7	Memory addressing test
8	Interactive test # 1: door lock, write protect, not ready, unsafe, off line, and single and dual sided sensor test
9	Interactive test # 2: data integrity after power cycle

5.1 PARTS

The following sections describe Part 1 through Part 9 of the diagnostic. The commands issued in these parts are listed immediately after a general description of the test. The format used to describe the commands is (CMD=0,1,2,3,4,5,6,7). This shows the 8 word command issued to the controller. The definitions for these words can be found in the H/W design document or the following manuals: Model FD1000 Flexible Disk System Installation and Operation, and the FD1000 Depot Maintenence Manual. In the part descriptions that follow all numbers not preceded by the '>' are decimal. All numbers preceded by the '>' and all numbers in the 8 word commands are hexidecimal. All labels in the 8 word commands are variables which are described with the command. A brief description of some of the command parts can be found in section 5.2.1.12 DC verb.

5.1.1 CONTROLLER TEST-PART 1

Part 1 runs a performance test on the controller. A drive need not be present because none is selected. A failure in this part indicates a faulty controller or faulty installation and the rest of the tests probably will not run correctly. Part 1 tests for the following:

1. The ability to write ones and zeros into bits 12-15 of controller register 0.
2. Tests the ability to communicate with bits 6,7,8,9,10, and 11 of controller registers 1,2,3,4,5,6,7.
3. Issues an I/O reset and verifies register 7 status is correct.
4. Issues a store registers command and checks the results against a known parameter table (CMD=0,0,>100,0,6,P1SDRT,0,0 where P1SDRT=the address to store 3 words).
5. Issues a store registers command for 0 ,1 ,2 , and 3 words and verifies the correct number of words were transfered (CMD=0,0,>100,0,X,P1SDRT,0,0 where X=byte count (0, 2, 4, or 6) and P1SDRT=address to store the words transfered).
6. Takes over the floppy's interrupt trap and tests for interrupt ability for register 7 and for the 4 attention line interrupts.
7. Issues a store registers command to an illegal TILINE

address and tests for a TILINE timeout (CMD=0,0,>100,0,2,>FBFE,>1F,0).

8. Sets all unit select lines in word 6 to zero by restore command and then checks for the correct status in registers 0 and 7 (CMD=0,>700,>100,0,0,0,0,0).

9. Tests slave logic by writing and reading bits in the TILINE floppy controller under mask.

5.1.2 RANDOM READ AND HEAD MOTION TEST-PART 2

Part 2 performs 150 random reads of >32 words by generating new random cylinder numbers for each read (CMD=0,>200,>100,X,>64,TEND,Y,0 where X=cylinder #, TEND=read buffer, and Y=selected unit). Next part 2 does a head motion test by reading one word from each cylinder starting at the maximum cylinder. The test reads both heads then decrements the cylinder count by 1 until it reaches cylinder 0. The test then reverses directions, again reading both surfaces (if present) of the cylinder before increasing the cylinder count by 1 until the maximum cylinder is reached (CMD=0,Z,>100,X,2,ACBUF,Y,0 where Z=read command plus head #(>200 or >201) X=cylinder #,ACBUF=read buffer, and Y=selected unit).

5.1.3 ADDRESSING TEST -PART 3

Part 3 checks the ability of the controller and the drive to select each track and each sector. This is accomplished by writing 40 words of data to every third sector while incrementing head number, and cylinder number. After writing the pattern to one third of all sectors on both surfaces of all the cylinders, the test reads all sectors written to, and verifies the data written by doing data compares of what was written (expected) and what was read (recieved) (CMD=0,Z,>100+S,X,>80,IO,Y,0 where Z=read or write command plus head # (>200,>201,>300,>301), S=sector #, X=cylinder #, IO=buffer address for reads or writes, and Y=selected unit).

5.1.4 ZEROS, ONES, AND WORST CASE DATA TEST -PART 4

Part 4 is divided into 2 parts. Part P4A writes a sector of zeros to sector 0, and a sector of ones to sector 1 of cylinder >4C on surface 0. The test then reads these sectors and compares the expected and recieved data. The test then increments to the next head (surface) if available and repeats part P4A (CMD=0,Z,S,>4C,>120,TEND,Y,0 where Z=command+head # (>200,>201,>300,>301), S=sectors per record + sector # (>100,>101), TEND=read and write buffer, and Y=selected unit #). Part P4B writes the worst case data pattern(>DB67 for dual, >AAAA for single density) to cylinder >4C sector 1 head 0. The test then reads the same sector and verifies the data by doing a data compare of expected and actual data. The test then increments to the next head (surface) and repeats part P4B (CMD=0,Z,>101,>4C,>120,TEND,Y,0 where Z=command + head # (>200,>201,>300,>301), TEND=the read and write buffer, and Y=selected unit).

5.1.5 CT, SI, ZERO FILL, AND BUSY FLAG TEST -PART 5

Part 5 is broken into four parts, A, B, C, and D.

A. Part A verifies that the controller can generate a command timeout. It does this by issueing a read command with an illegal sector # and checking that the command is completed within 3 seconds and the proper error code has been generated in register 7 (>A004) (CMD=0,>200,>11C,0,>10,TEND,Y,0 where TEND=read buffer, Y=selected unit #).

B. Part B verifies that the controller can generate a seek incomplete (SI) in register 0 and a unit error (UE) in register 7 by attempting to write past the last sector,last cylinder, and last surface (CMD=0,Z,S,C,>1F4,0,Y,0 where Z=write command + maximum surface # (dual=>301,single=>300), S=sectors/record + maximum sector # (>119), C=maximum cylinder # (>4C), and Y=selected unit #). The test then checks registers 0 and 7 for the expected error status(0=>0400,7=>A001).

C. Part C insures that data is not destroyed if a read or write command with a word count of zero is issued. This is done by formating surface 0, cylinder 0, sector 0 with a known data pattern(>E5E5) (CMD=0,>100,>100,0,>120,P5CFD,Y,0 where P5CFD=location of data pattern (>E5E5), and Y=selected unit #). The test then reads the formatted sector, with a word count=0, into a read buffer which has been initialized with zeros (CMD=0,>200,>100,0,4,P5CRD,Y,0

where P5CRD=read buffer which has been cleared, and Y=selected unit #). The read buffer is then checked to verify it has not been changed. The test then issues a write command to the formatted sector, with a word count=0, and a write buffer with a known data pattern (>1234) (CMD=0,>300,>100,0,0,P5CZW,Y,0 where P5CZW= write buffer with a known data pattern (>1234), and Y=selected unit #). The test then verifies that the diskette was not written on by reading the first word of the formatted sector and verifying that it contains the correct data pattern (>E5E5) (CMD=0,>200,>100,0,2,P5CRD,Y,0 where P5CRD=read buffer, and Y=selected unit #).

D. Part D checks for normal operation of the busy flag (IDLE bit). This is done by writing the slave registers to diskette. The idle bit should not be set since the controller is doing the write command (CMD=0,>300,>100,0,>10,TFC,Y,0 where TFC=address of TILINE control space(slave registers),Y=selected unit # + MSB of TILINE address). The test then reads the 8 words written, to verify that register 0 bit 0(idle bit) was not set during write command (CMD=0,>200,>100,0,>10,P5DRD,Y,0 where P5DRD=read buffer, and Y=selected unit #).

5.1.6 HEAD SWITCH TEST-PART 6

Part 6A checks the ability of the unit to switch from head zero to head one and from head one to head zero for a double density diskette. If a single density media is installed the test becomes a track switch test checking the ability of the unit to switch from one track to the next. This is done by writing 216 words of a known pattern to the last sector of cylinder 0 surface 0 (CMD=0,>300,S,0,>1B0,TEND,Y,0 where S=sectors/record + maximum sector # (>119), TEND=write buffer with sequential data pattern 216 decrementing to 0, and Y=selected unit #). The test then reads 216 words starting at the last sector of cylinder 0 surface 0 and compares the two buffer areas for data compares (CMD=0,>200,S,0,>1B0,TEND+>1B0,Y,0 where S=sectors/record + maximum sector #,TEND+>1B0=read buffer, and Y=selected unit #). This process is continued by writing to the last sector of cylinder 0 surface 1 if present (CMD=0,>301,S,0,>1B0,TEND,Y,0 where S=sectors/record + maximum sector #, TEND=write buffer,Y=selected unit #).Then reading the last sector of cylinder 0 surface 1 and comparing the data of the read and write buffer areas.(CMD=0,>201,S,0,>1B0,TEND+>1B0,Y,0 where S=sectors/record + maximum sector #, TEND+>1B0=read buffer, and Y=selected unit #).

Part 6B is a head load stabilization test. This part first

issues a write controller memory command to decrease the load delay(CMD=0,>8500,0,>80C2,2,P6BUNL,Y,0 where P6BUNL=address of pattern (>FFFE) to write to controller memory, and Y=selected unit #). The test then loops on the following three insructions: 1. write command (CMD=0,>300,>100,>4C,>900,TEND,Y,0 where TEND=write buffer, and Y=selected unit #). 2. read command (CMD=0,>A00,>100,>4C,>900,TEND,Y,0 where TEND=read buffer, and Y=selected unit #). 3. extended command (CMD=0,>8700,0,>D,WDCT,0,Y,0 where wdct=word delay counts, and Y=selected unit #).

5.1.7 MEMORY ADDRESSING TEST - PART 7

Part 7 checks the ability of the disk controller to address all memory present in the machine. The test finds memory available after loading the diagnostic and uses it as a test buffer for addressing in part A. If less than 4K words are available the test is skipped. Part 7A determines if there is at least 4K of memory after the program by attempting to write each word's address into itself until it gets a TILINE timeout or it reaches the 32K boundary. It then writes this whole area to diskette (CMD=0,>300,>100,0,P9BYTE,P9ADDR,Y,0 where P9BYTE=the number of bytes found between the end of the program and the 32K boundary, P9ADDR=address of 1st word past end of program, and Y=selected unit #). The test then clears the whole buffer area found between the end of the program and the 32K boundary. After the buffer area has been set to 0 the test reads the data written to diskette back to the buffer area and verifies each word has its address written into it (CMD=0,>200,>100,0,P9BYTE,P9ADDR,Y,0 where P9BYTE=the number of bytes to read, P9ADDR=address of read buffer (1st word past program), and Y=selected unit #).

Part 7B checks the mapped memory which may be present. It does this in >80 byte increments using the same process used in part A. If possible it writes each words address to itself in the >80 byte buffer area, writes the buffer area to diskette, clears the buffer area, reads the data from

diskette back to the buffer area, and verifies the words in the buffer area contain their own address (15 LSB's of address). If there is no mapped memory the test determines this and skips part 7B.

Part 7C tries to determine if the controller can recognize more than one address as its TILINE control space. It does this by issueing a nop command with the command words in a known pattern. The test then reads all of mapped memory in 4K blocks looking for the data pattern in six consecutive words followed by the correct error status. If there is no mapped memory the test skips part 7C. (CMD=>5544,>5544,>5544,>5544,>5544,>5544,>5544,>5544)

5.1.8 INTERACTIVE PART 1-PART 8

Part 8 checks the operation of the door lock, write protect, unsafe, not ready, off line, and the dual and single density media sensors. Operator intervention is required so there is no loop on capability (L8 or C8). It requires the use of both a single and a double density diskette. To run properly, the double density diskette must be formatted. The test is run by asking the operator to perform certain tasks and checking for the correct status by doing a store registers command after each task is completed (CMD=0,0,>100,0,6,SRDAT,Y,0 where SRDAT=address to write three words to, and Y=selected unit #). The following information and instructions are printed on the interactive device:

```
PART 8 START

INSTALL WRITE PROTECTED DISKETTE AND PRESS RETURN KEY -

INSTALL A NON-PROTECTED, SINGLE DENSITY DISKETTE
LEAVING THE DOOR OPEN AND PRESS THE RETURN KEY -

CLOSE THE DOOR AND PRESS RETURN KEY -

INSTALL A DOUBLE DENSITY DISKETTE AND PRESS RETURN KEY -

YOU NOW HAVE 20 SECS TO CHECK THE DOOR LOCKED
DO NOT PRESS RETURN UNTIL THE NEXT QUESTION IS ASKED

WAS THE DOOR LOCKED? (0=NO,1=YES) DEFAULT=1 -

WOULD YOU LIKE TO TRY AGAIN? (0=NO,1=YES) DEFAULT=0 -

PART 8 DONE
```

5.1.9 INTERACTIVE PART 2

Part 9 is a power fail data integrity test. Operator intervention is required so there is no loop on capability (L9 or C9). It requires that a formatted diskette be installed, that a battery pack be on the system, and the power be cycled. The test takes over the power up trap (WP=0000,PC=0002) and when power is cycled the integrity of the diskette is checked by doing a read command with the transfer inhibit bit set (CMD=0,>A00,>100,0,>9000,TEND,Y,0 where TEND=dummy read buffer, and Y=selected unit #). The following information and questions are printed on the interactive device:

```
PART 9 START
IS THE DISKETTE FORMATTED? (0=NO,1=YES) DEFAULT=0 -
DOES THIS SYSTEM HAVE BATTERY SUPPORT? (0=NO,1=YES) DEFAULT=0 -
HAS SYSTEM POWER BEEN CYCLED? (0=NO,1=YES) DEFAULT=0
PART 9 DONE
```

5.2 OPERATOR INTERFACE

DDFLOP runs under the control of DOCS. Refer to the section on DOCS in the Diagnostic Handbook for a detailed description of DOCS.

5.2.1 AVAILABLE VERBS

In addition to the verbs supplied by the DOCS package, DDFLOP also provides the verbs listed in Table 1.

TABLE 1

VERB	FUNCTION
IT	Initialize Test
E1	Execute Part 1
.	.
.	.
E9	Execute Part 9
L1	Loop on Part 1
.	.
.	.
L7	Loop on Part 7

NOTE

There are no verbs to loop on Parts 8 or 9

C2	Loop on Part 2 for all available units
.	.
.	.
C6	Loop on Part 6 for all available units
C7	Loop on Part 7 for all available units
CA	Loop on Parts 1-7 for all available units

NOTE

No unit is required for the execution of Part 1 but it is included in CA to insure the exercising of the controller.

EA	Execute Parts 1-9 on the unit under test
LA	Loop on Parts 1-7 on the unit under test
PE	Print error count
IC	Issue command
IM	Issue multiple commands
LO	Loop on multiple commands
DC	Display controller status
ST	Execute controller self test Part (n)
FD	Format diskette
OT	Execute off track data test
RM	Read controller memory
WM	Write controller RAM

AL	Disk alignment test
RD	Read formatted diskette
RS	Reset H/W
MT	Margin test

5.2.1.1 IT VERB

The Initialize verb is used to initialize the floppy parameters: TILINE address (default = F800), floppy interrupt level (default = D), unit under test (default = 0), and other available units (default = FF(none)). The following shows the output from the IT verb:

```
FLOPPY TILINE ADDR. DEFAULT = F800 -
FLOPPY INTERRUPT LEVEL DEFAULT = 0D -
UNIT TO BE TESTED DEFAULT = 00 -
OTHER AVAILABLE UNITS (0 - 3 TERM. WITH FF)
DEFAULT = FF -
```

5.2.1.2 E1 THROUGH E9

The Execute verbs are used to execute Part 1 through Part 9 of the diagnostic (one part per verb).

5.2.1.3 L1 THROUGH L7

The Loop verbs are used to loop on Part 1 through Part 7 of the diagnostic (one part per verb). The "L" verbs will loop until the operator terminates the execution by the entry of an at (@) sign on the interactive device. If header messages are enabled, the loop count will be printed after each loop of the "L" verb.

5.2.1.4 C2 THROUGH C7

The Change verbs are used to loop on Part 2 through Part 7 (one part per verb) changing the unit under test to the next available unit after each loop. After the last available unit is tested the loop starts over on the original unit. To terminate the test the operator must enter an at (@) sign on the interactive device.

5.2.1.5 EA VERB

The Execute All verb is used to execute all nine parts of the diagnostic in order E1 - E9.

5.2.1.6 LA VERB

The Loop on All verb is used to loop on parts one through seven in order. The loop count is printed after each loop of all seven parts is completed. To stop the execution of this verb the operator enters an at (@) sign on the interactive device.

5.2.1.7 CA VERB

The Change All verb is used to loop on parts one through seven changing the unit under test to the next available unit after each loop. To stop execution the operator should enter an at (@) sign on the interactive device.

5.2.1.8 PE VERB

The Print Errors verb is used to print the present DATA, TIMING, and STATUS error counts. It should be used by the operator to determine the number of errors that occurred in the execution of the last verb. The format is:

```
ERROR COUNTS
DATA = 0000 TIMING = 0000 STATUS = 0000
```

5.2.1.9 IC VERB

The Issue Command verb can be used to issue a command to the TILINE floppy controller (TFC) from the memory address input by the operator. The command is eight words long.

FORMAT: VERB? -IC - ADDR - XXXX - CHECK STATUS?(0=NO,1=YES) DEFAULT=1 -

ACTION: Issue the command located at address ADDR. If a status check is desired and if error printing is enabled, print any errors that occur.

5.2.1.10 IM VERB

The Issue Multiple commands verb can be used to issue multiple commands to the TFC from memory. The commands require eight words each and must be stored in consecutive locations.

FORMAT: VERB? -IM - ADDR-XXXX - #-X - CHECK STATUS?(0=NO,1=YES)DEFAULT=1 -

ACTION: Issue the # of commands specified starting at the location ADDR. If a status check is desired and if error printing is enabled, print any

errors that occur.

5.2.1.11 LO VERB

The Loop On commands verb is identical to the IM verb except that at the completion of the last command the test will go back to the first command and loop on the command list until the operator enters an at (@) sign on the interactive device. The loop count will be displayed on the Front Panel.

FORMAT: VERB? -LO - ADDR-XXXX - #-X - CHECK STATUS?(0=NO,1=YES)DEFAULT=1 -

ACTION: Issue the # of commands specified starting at the location ADDR. If a status check is desired and if error printing is enabled, print any errors that occur. Loop on specified commands until interrupted by an '@' sign on the interactive device.

5.2.1.12 DC VERB

The Display Controller status verb is used to display the status of the TILINE Floppy Controller (TFC). It reads the TFC slave registers, formats them, and then outputs them to the interactive device. The following shows the format and explanation of the DC verb output:

VERB? - DC

```
CONTROLLER STATUS
FLOPPY STAT   COMD    HD    S/R  R A   CYL A   BYTE C   MEM AD   SEL   CONT STATUS
  XXXX         XX     XX     XX  XX     XXXX     XXXX   XXXXXX   XX        XXXX
```

Status report explanation (DC verb and status errors).

FLOPPY STAT - This word gives the selected drive status and is obtained from TFC register 0.

ILLEGAL	-CXXX	***Any of these
NOT READY	-DXXX***	bits may be
WRITE PROT	-2XXX***	combined in the
UNSAFE	-1XXX***	status returned.
SEEK INCOMP	-X4XX***	

COMD - Command number:

00	-	store registers
01	-	write format
02	-	read data
03	-	write data
04	-	read id
05	-	nop=command timeout
06	-	seek
07	-	restore
80	-	extract interface factor
81	-	write interlaced sector format
82	-	read delete
83	-	write delete
84	-	read unformatted
85	-	write to controller memory
86	-	read controller memory
87	-	extended test commands

HD - Head number: 0 - bottom

1 - top

S/R - Sectors per record constant: 01

RA - Record address, sector number: 0 to >19

CYL A - Cylinder address: 0 to >4C

BYTE C - Byte count for data transfers: >80 for single density

>120 for double density

MEM AD - TILINE memory address for data transfers

CONT STATUS - Controller status (TFC register 7)

5.2.1.13 ST VERB

The Self Test verb is used to execute a particular portion of the controller extended self test (CMD=0,>8700,>100,X,0,0,Y,0 where X=the number of the portion of the selftest to run, and Y=selected unit #). Tests 1-8 are included in the controller power up or h/w reset selftest. Tests 9-C are special selftests and must be set up accordingly if they are to run without errors. If,test 1-8 will not run without errors the controller board is probably bad. The tests and their numbers which may be performed include:

0 = Microprocessor selftest
1 = On board RAM test
2 = ROM test
3 = PIA test
4 = Slave register test
5 = TILINE master/busy slave
6 = Data generator/separator
7 = Disk drive interface test
8 = TILINE interrupts test
9 = Extended disk I/F test(loop back)
A = VCO adjust test
B = Drive head test
C = Tap-Tap test
D = Variable delay command

FORMAT: VERB? -ST - SELF TEST # -X - CHECK STATUS?(0=NO,1=YES)DEFAULT=1 -

ACTION: Execute self test part (n),

where n may be from 0-D.

NOTE

For scope loops on individual Self Tests add >C000 to the test number.

5.2.1.14 FD VERB

The Format Diskette verb is used to format the whole diskette. The FD verb performs a diskette surface analysis followed by initialization of all tracks with the worse case data pattern (CMD=0,>100+H,>100,CYL,WC,FDFG,Y,0 where H=head #, CYL=cylinder #, WC=maximum bytes per sector, FDFG=address of the worse case data pattern (>DB67 for double sided, and >AAAA for single sided diskettes), and Y=selected unit #). The status check flag is set and if an error occurs it will be printed. It takes about two minutes for a single density and about four minutes for a double density diskette to be formatted.

WARNING

ANY INFORMATION PREVIOUSLY STORED ON THE DISKETTE WILL BE LOST WHEN THE FD VERB IS EXECUTED

Because this verb writes over the entire diskette the operator is required to answer the following question:

FORMAT: VERB? - FD - ARE YOU SURE?(0=NO,1=YES)DEFAULT=0

ACTION: The diskette is formatted. The status checker is on and any errors will print out if error reporting was turned on during DOCS initialization. After the diskette has been formatted the following message will be printed.

FD COMPLETE

5.2.1.15 OT VERB

The Off Track verb is a special verb used with a specially prepared off track data diskette (P/N 2267260-0001). This verb verifies head alignment by being able to read data correctly when it is at the limits of specification. The test disables retries before it begins the reads (CMD=0,>8500,>100,>80C6,2,RTKILL,Y,0 where RTKILL=address of data pattern (>FFFF) that turns off retries, Y=selected unit #). There are only two written tracks on the diskette (>47 and >49). The user should specify track >47 until diskette wears out, then use track >49 for the reads (CMD=0,>200+H,>100+S,CYL,BYTE,TEND,Y,0 where H=head #, S=sector #, CYL=cylinder #, BYTE=maximum bytes per sector, TEND=read buffer, and Y=selected unit #). The diskette must be write protected before loading it.

FORMAT: VERB?- OT

ENTER TEST TRACK NUMBER(EITHER 47 OR 49 HEX) -

ENTER THE DESIRED # OF LOOPS -

ACTION: The test executes printing any errors that occur(if error print on). The loop count is printed after each loop through the test.

5.2.1.16 RM VERB

The Read Memory verb is used to read the contents of the controller memory (ROM or RAM). The operator may view from 8 to 64 words (in blocks of 8 words) with each execution (CMD=0,>8600,>100,CONADD,BYTES,TEND,0,0 where CONADD=controller address to start reading from, BYTES=total number of bytes to transfer (16 x # of word blocks chosen), TEND=address of read buffer).

FORMAT: VERB? - RM -

CONTROLLER STARTING ADDRESS? - XXXX

OF 8 WORD BLOCKS (1 - 8)? - X

ACTION: Display the specified number of eight word blocks on the interractive device.

NOTE

The RM verb may be used to display an internal controller command trace, by specifying the controller address = 8140. Up to twelve commands from the last I/O reset are stored in this controller trace buffer and may be viewed with this verb.

5.2.1.17 WM VERB

The Write Memory verb is used to write from TILINE memory to controller RAM. The operator may change the contents of one location or as many as all 256 locations with one execution (CMD=0,>8500,>100,CONADD,BYTE,TEND,0,0 where CONADD=controller address to start writing to, BYTE=number of bytes to write to controller memory, TEND=address of data to start writing to controller memory).

FORMAT: VERB? - WM

CONTROLLER STARTING ADDRESS? - XXXX

ENTER THE TILINE STARTING ADDR - XXXX

ENTER # OF WORDS (1 - >100) - XXXX

ACTION: Move the specified # of words from the specified TILINE starting address sequentially to the specified controller RAM address.

5.2.1.18 AL VERB

The Alignment verb is a verb used to read unformatted from the specified cylinder and head so that disk alignment may be accomplished with the use of a specially prepared alignment diskette (P/N 2267259-0001). (CMD=0,>8400+H,0,CYL,6,ENDDIG,Y,0 where H=head #, CYL=cylinder #, ENDDIG=address of read buffer, and Y=selected unit #). The test will loop on the reads until the operator enters an at sign (@) on the interactive device.

FORMAT: VERB? - AL

NUMBER OF DIFFERENT READS? 1 OR 2?- X

READ FROM CYLINDER NUMBER? - XX

HEAD NUMBER? - X

READ FROM CYLINDER NUMBER? - XX

HEAD NUMBER? - X

ACTION: Loop on unformatted read from location specified. If two locations are specified it will alternate between the two. Disk status is not checked. Data is not checked.

5.2.1.19 RD VERB

The Read Diskette verb allows diskettes formatted on other drives to be read on the selected drive. This verifies the compatability between drives. The reads may be done with retries enabled or disabled. (CMD=0,>8500,>100,>80C6,2,RDINH,Y,0 where RDINH=address of word to write to controller memory(pattern=>FFFF), and Y=selected unit #). In either case however the transfer inhibit bit is set for the read (CMD=0,>A00+H,>100,CYL,MAXTRK,0,Y,0 where H=head #, CYL=cylinder #, MAXTRK=maximum bytes per track, and Y=selected unit #).

FORMAT: VERB? - RD

START READ DISKETTE

DO YOU WANT TO READ WITH RETRIES?(0=NO,1=YES)DEFAULT=00-

ERROR COUNTS

DATA = 0000 TIMING = 0000 STATUS=0000

READ DISKETTE COMPLETE

ACTION: Starting with head 0, cylinder 0, read the whole diskette with data transfer inhibited, and retries enabled if specified.

5.2.1.20 RS VERB

The Reset verb issues a H/W I/O reset. It then waits for a required 2.5 seconds for the controller to complete the selftest. This verb may be used after interrupting the self test verb (ST) to reinitialize the H/W before continueing testing.

FORMAT: VERB? - RS

ACTION: A H/W reset is issued which forces the controller to do its selftest. After waiting for selftest to complete the following message is printed.

CONTROLLER HAS BEEN RESET

5.2.1.21 MT VERB

The Margin Test verb first asks several initialization questions shown below. It then inhibits controller retries and inhibits precompensation, if the defaults were taken, by using the write controller memory command (CMD=0,>8500,>100,CADD,2,RDINH,Y,0 where CADD=controller address to write to (>80C6=retries,>80CE=precompensation), RDINH=address of pattern (>FFFF=retries,0=precompensation) to write to controller memory, and Y=selected unit #). The test then formats both surfaces of the diskette tracks 0 and >4C with a data pattern (>DB67) (CMD=0,>100+H,>100,CYL,BYTE,MTFD,Y,0 where H=head #, CYL=cylinder #(0 or >4C), BYTE=maximum # of bytes per track, MTFD=address of data pattern to write (>DB67), and Y=selected unit #). The test then reads the formatted tracks with the transfer inhibit bit set for the number of loops selected. It first reads all heads of cylinder 0 for the specified loops, and then reads all heads of cylinder >4C for the specified # of loops (CMD=0,>A00+H,>100,CYL,BYTE,0,Y,0 where H=head #, CYL=cylinder # (0 or >4C), BYTE=maximum # of bytes per track, and Y=selected unit #).

FORMAT: VERB - MT

START MARGIN TEST

********** WARNING **********
ERRORS MAY BE EXPECTED IF THIS TEST IS RUN WITH BOTH RETRIES
AND PRECOMPENSATION DISABLED.

HOW MANY TIMES DO YOU WISH TO LOOP ON READ? (HEX DEFAULT=80) -

DO YOU WANT TO READ WITH RETRIES? (0=NO,1=YES)DEFAULT=00 -

DO YOU WANT WRITE PRECOMPENSATION?(0=NO,1=YES)DEFAULT=00 -

ERROR COUNTS

DATA = 0000 TIMING = 0000 STATUS=0000

MARGIN TEST COMPLETE

ACTION: After inhibitting the controller retries and
controller precompensation if selected the
test formats both surfaces of tracks 0 and >4C.
The test reads the formatted tracks with the
transfer inhibit bit set for the specified
amount of loops.

5.2.2 STANDARD DOCS INITIALIZATION

Reference Diagnostic Handbook or DOCS Program Description (PD).

5.2.3 PROGRAMMER PANEL DOCS INITIALIZATION

When running DDFLOP in the Front Panel Mode, besides asking the question IDLE ON ERRORS?, the IT verb questions are asked.

NOTE

As only the EA verb is run in the Programmer Panel test, take the default value (>FF) for available units in IT verb (this mode tests only the selected unit).

After all initialization questions are answered by the operator the test will start executing. The first question asked is:

IS THE DISKETTE FORMATTED?(0=NO,1=YES)DEFAULT=0

If the answer is no the test will abort. Therefore, if a formatted diskette is not installed, install one and answer the question yes. The test will run approximately 5 minutes before operator interaction is required. Of course this

time will be longer if any errors occur. Should an error occur the error number will appear on the front panel. When the test starts Part 8 operator interaction is required. The 990 will go into the idle mode with a >0000 on the front panel. At this point the operator is being prompted:

INSTALL A WRITE PROTECTED DISKETTE AND PRESS RETURN KEY

After doing this the operator depresses HALT, CLEAR, then RUN. If an error occurs the error # will be displayed otherwise the 990 will idle with a >0000 on the front panel. At this time the operator is being prompted:

INSTALL A NON-PROTECTED SINGLE DENSITY DISKETTE
LEAVING THE DOOR OPEN AND PRESS THE RETURN KEY

After doing this the operator depresses HALT, CLEAR, then RUN. If an error occurs the error # is displayed. The 990 idles with a >0000 displayed The operator is being prompted:

CLOSE THE DOOR AND PRESS RETURN KEY

After doing this the operator depresses HALT, CLEAR, then RUN. If an error occurs the error # will be displayed otherwise the 990 will idle with >0000 on the front panel. At this time the operator is being prompted

INSTALL A DOUBLE DENSITY DISKETTE AND PRESS RETURN KEY

NOTE

The diskette must be properly formatted for correct operation.

After doing this the operator depresses HALT, CLEAR, then RUN. If an error occurs the error # will be displayed. The operator now has 20 seconds to check the door locked. After about 20 seconds the 990 idles with >0000 displayed. The operator is being asked:

WAS THE DOOR LOCKED? (0=NO,1=YES) DEFAULT=1 -

If the operator answers no the question is asked:

WOULD YOU LIKE TO TRY AGAIN?(0=NO,1=YES)DEFAULT=0 -

If the operator answers yes the test loops back and locks the door for another 20 seconds. This loop may be repeated as many times as desired. If the operator answers yes to the door being locked (or does not try again) the 990 starts executing Part 9. The 990 idles displaying a >0000 on the front panel asking the operator:

IS THE DISKETTE FORMATTED?(0=NO,1=YES)DEFAULT=0 -

Answering no to this question will cause the test to abort. If the operator gets this far, a formatted diskette is installed. The question is intended for use in normal

operations where the E9 verb may be run by itself.

DOES THIS SYSTEM HAVE BATTERY SUPPORT?(0=NO,1=YES)DEFAULT=0

If the answer is no the test aborts. If the answer was yes the 990 idles displaying a >0000 asking the operator:

HAS THE POWER BEEN CYCLED?(0=NO,1=YES)DEFAULT=0

If the answer is no the 990 idles displaying a >0000 prompting the operator to:

TURN OFF SYSTEM POWER FOR 30 SECONDS.

If the operator answered yes to power being cycled or if he turns off power for 30 seconds and then back on the test is executed.

6.0 ERROR MESSAGES AND ERROR NUMBERS

When an error occurs in the execution of any part of the test (if error message printing was enabled in .IS), an error message will be printed on the selected error message output device. The error message indicates which error has occurred. The error message number will also appear on the front panel. The error messages are listed in table 2.

TABLE 2

NUMBER(HEX)	ERROR MESSAGE
	PART 1
11	1-1 NO COMMUNICATIONS WITH CONT REG XX CHANGE THE CONTROLLER

Part 1 could not write and read back bits to the controller register printed. This indicates a bad controller board.

12	1-2 NO SET/RESET BIT XX OF CONT REG XX MAY BE THE CONTROLLER OR TILINE

Part 1 could not set and/or reset the indicated bit of the controller register printed. This indicates a bad controller board or the TILINE is not working correctly.

13	1-3 NO SET/RESET CONT REG 5 BIT XX MAY BE CONTROLLER OR TILINE

Part 1 could not set and/or reset the indicated bit of controller register 5. This indicates a bad controller board

or the TILINE is not working correctly.

14 1-4 NO SET/RESET CONT REG 0 BIT XX
MAY BE CONTROLLER OR TILINE

Part 1 could not set or reset the indicated bit of controller register 0. This indicates a bad controller board or the TILINE is not working correctly.

15 1-5 STORE REGISTERS ERROR
WORD1 WORD2 WORD3
XXXX XXXX XXXX

Part 1 has done a store registers command and the words written do not compare with a table which contains the correct words for each type of diskette.

16 1-6 NO UNIT SELECTED,
REG 0 SHOULD=C0X0 REG 7 SHOULD=A001
REG 0 = XXXX REG 7 = XXXX

Part 1 issued a restore command with no unit selected, the expected error status from registers 0 and/or 7 was not recieved.

17 1-7 FAIL SELF TEST CONTROLLER REG 2 = XXXX
EXPECTED REG 7 = A100 AFTER I/O RESET
REG 0 = XXXX REG 7 = XXXX

Part 1 issued an I/O reset, the expected error status in register 7 was not set.

NUMBER(HEX) ERROR MESSAGE

18 1-8 TILINE FLOPPY CONTROLLER REG DATA TEST
WAS EXP REG.
XXXX XXXX XX

Part 1 writes various patterns under various masks to all controller registers and then reads the registers. The register indicated did not return the expected data.

19 1-9 FAILURE IN REG 7 INTERRUPT LOGIC

Part 1 tried and failed to generate an interrupt by setting the interrupt enabled bit in register 7.

1A 1-A FAILURE IN REG 0 INTERRUPT LOGIC

Part 1 tried and failed to generate an interrupt by setting each of the four attention interrupt bits in register 0.

1B 1-B TILINE TIMEOUT ERROR
R7 STATUS EXPECTED=A020
REG 0 = XXXX REG 7 = XXXX

Part 1 attempts to cause a TILINE timeout by writing to TILINE address 1FFBFE. Register 7 did not return with a TILINE timeout status.

1C 1-C ERROR DID STORE REG WITH WC = XX
WORDS TRANSFERED = XX

Part 1 issued a store register command with the word count shown in the message. The number of words transfered however was not equal to the word count in the command.

NUMBER(HEX)	ERROR MESSAGE

1D 1-D NO TILINE TIMEOUT IN 3 SECONDS

Part 1 attempts to cause a TILINE timeout by writing to TILINE address 1FFBFE. If no TILINE timeout occurs within 3 seconds this error message followed by the controller status message will be printed.

NONE ** CONTROLLER MALFUNCTION **

This message will be printed after part 1 completion if there are any data, status, or timing errors in any of the nine subtest of Part 1.

NONE INTERRUPT TIMEOUT ERROR
FLOPPY INTERRUPT DID NOT OCCUR IN 8 MS

This message may be printed with errors 19 or 1A. If during the floppy interrupt test in Part 1 the floppy interrupt does not occur this message will be printed with either message 19 or 1A depending on how the interrupt is being set.

NOTE

Parts 2, 3, and 4 have no error messages of their own. However, they may output error messages from the command issuer, status checker, reset, interrupt, and compare data subroutines.

PART 5

NUMBER(HEX) ERROR MESSAGE

51 5-1 ERROR PART 5 SI TEST
REG0 STATUS EXP=04XX REG7 STATUS EXP=A001
REG 0 = XXXX REG 7 = XXXX

Part 5 attempts to set the seek incomplete bit in register 0, and the unit error bit in register 7 by trying to write past the last surface, last cylinder, and last sector. The status returned in register 0 and/or register 7 was not the expected status.

52 5-2 ERROR PART 5C
DID WRITE WITH WC=0 BUT DATA CHANGED
EXPECTED DATA = E5E5 BUT GOT DATA = XXXX

Part 5 after formatting the diskette to data pattern E5E5 did a write command with the word count set to zero. The data pattern expected had changed when the diskette was read.

53 5-3 ERROR PART 5 BUSY FLAG SET
DATA (MSB SHOULD BE 0) XXXX

Part 5 has issued a write command using the status registers as the write buffer. The test then reads these words back and verifies that bit 0 of register 7 is not set. This is the idle bit which should not have been set before the write command completed. This message prints the register 7 status that was stored on diskette.

54 5-4 PART 5A TIMEOUT ERROR
NO COMMAND TIMEOUT IN 500 MS

Part 5 issued a read command which did not complete in the time allowed.

55 5-5 COMMAND TIMEOUT ERROR
REG 7 STATUS EXP= A004
REG 0 = XXXX REG 7 = XXXX

Part 5 issued a read command with an illegal sector # trying to set the command timeout bit in register 7. The status of register 7 after the read did not have the expected data.

56 5-6 DID A READ WITH WORD COUNT = 0
EXPECTED DATA = 0 BUT GOT DATA = XXXX

Part 5 did a read command with a word count of zero into a read buffer with a known data pattern. The expected data in the read buffer had changed after the read.

NUMBER(HEX) ERROR MESSAGE

PART 6

61 6-1 FAILURE TO SWITCH HEADS 0 TO 1, CYL 0
REG. 7 STATUS EXP=C000
REG 0 = XXXX REG 7 = XXXX

Part 6 attempts to do a head or track switch failed. The expected status of register 7 did not set set.

62 6-2 FAILED TO SWITCH HEADS FROM 1 TO 0 CYL.0
REG. 7 STATUS EXP=C000
REG 0 = XXXX REG 7 = XXXX

Part 6 attempts to do a switch of head or track failed. the expected status of register 7 did not set set.

63 6-3 PART 6 HEAD SWITCH FAIL ON DATA COMPARE

Part 6 writes a known pattern to diskette switching heads during the write. The test then reads the data written and compares it to the known pattern. This message is printed if the data does not compare between the write and the read.

NONE PART 6 HEAD SWITCH FAILURE

This message may be printed just before error messages 61 and 62. The message simply indicates a failure in Part 6.

PART 7

71 7-1 DATA NOT EQUAL TO ADDRESS
ADDRESS=XXXX
DATA=XXXX

Part 7A attempts to address all unmapped memory by writing the address of every word of memory into itself. This message is printed if a memory word in the first 32K of memory does not contain its own address.

72 7-2 COMPARISON ERROR IN MAP MEMORY
MAP ADDRESS=XXXX
MAP BIAS=XXXX
TILINE ADDRESS=XXXXXX
EXPECTED DATA=XXXX
DATA READ=XXXX

Part 7B tests mapped memory using the same process as Part 7A on 80 byte blocks at the beginning of each 4K boundary. This message is printed if the data read is not equal to the least significant bits of the data word's address in any of mapped memory.

73 7-3 CONTROLLER ADDRESSING ERROR
THE TFC MAY ALSO BE ADDRESSING XXXXXX
AS ITS TILINE CONTROL SPACE ADDRESS

Part 7C searches mapped memory looking for 8 words which the floppy controller may be addressing as a control space address in addition to the real control space address. This message is printed if the diagnostic determines the controller is addressing more than one set of 8 words as its

NUMBER(HEX)	ERROR MESSAGE
	tiline peripheral control space.

NUMBER(HEX) ERROR MESSAGE

PART 8

81 8-1 ERROR WRITE PROTECT TEST EXPECTED REG 0 = 20X0
REG 0 = XXXX REG 7 = XXXX

Part 8 asks for a write protected diskette to be installed. After the response the status checked indicated a write protected diskette had not been installed.

82 8-2 ERROR OFF LINE, NOT READY UNSAFE TEST
EXPECTED REG 0 = D0X0
REG 0 = XXXX REG 7 = XXXX

Part 8 asks that a single sided diskette be installed with the door left open. After the response the status is checked for the off line bit, not ready bit, and unsafe bit in register 7 being set. This message is printed if one of those bits is not set.

83 8-3 ERROR IN STORE REGISTERS VALUES

	EXPECTED	
XXXX	XXXX	XXXX
	RECIEVED	
XXXX	XXXX	XXXX

Part 8 uses a store registers command to determine if a diskette is single or double sided diskette by comparing the returned words with tables containing the proper single or double sided parameters. This message will be printed if the returned words do not compare with the expected data in the correct table.

NUMBER(HEX)	ERROR MESSAGE

PART 9

91	9-1 DATA INTEGRITY TEST FAILED. MAY BE DUE TO A BADLY FORMATTED DISKETTE

Part 9 tests that data will not change after a power cycle. This message prints if there is bad data on the diskette after the power cycle. This may be caused by having a diskette which was formatted incorrectly before the power cycle or by data actually changing due to the power cycle.

NOTE

COMMON SUBROUTINE ERROR MESSAGES

The following error messages are generated by subroutines which are used by all or some of Parts 1 thru 9. The following subroutines contain at least one error message: Command Issuer, Status Checker, Reset, Floppy Interrupt, Compare Data, and the Off Track Disc test (OT verb).

C-1 UNIT=X REG.7 = XXXX
FLOPPY STATUS UNIT=X REG.0 = XXXX

STATUS ERROR CONTROLLER STATUS COMP=X ERR=X IDLE=X

T.F.C. REG.

FLOPPY STAT	COMD	HD	S/R	R A	CYL A	BYTE C	MEM AD	SEL	CONT STATUS
XXXX	XX	XX	XX	XX	XXXX	XXXX	XXXXXX	XX	XXXX

COMMAND ISSUED

FLOPPY STAT	COMD	HD	S/R	R A	CYL A	BYTE C	MEM AD	SEL	CONT STATUS
XXXX	XX	XX	XX	XX	XXXX	XXXX	XXXXXX	XX	XXXX

The Status Checker subroutine generates this error message if the status in register 7 has the error bit set, or the complete bit is not set. This message will only be printed if the status check flag has been set (contains nonzero data). This message contains several error messages not associated with an error number. Section 5.2.1.12 of this PD defines parts of this status error message.

NUMBER(HEX)	ERROR MESSAGE

C2 C-2 NR BIT FOR UNIT XX RESET TOO SOON

The Status Checker subroutine generates this error message if the not ready bit in register 0 is set for the unit indicated. This message will not print unless the status check flag has been set.

C3 C-3 UNEXPECTED FLOPPY INTERRUPT AT XXXX
REG 0 = XXXX REG 7 = XXXX

The Floppy Interrupt subroutine generates this message if a floppy interrupt occurs that was not expected. The address given is the address of the next instruction to be executed after the interrupt subroutine completes.

C4 C-4 THE TFC DID NOT RESPOND IN 10 SECONDS

The Command Issuer subroutine generates this error if a command was issued and was not completed within 10 seconds.

C5 C-5 THE ATTN LINES DID NOT SET IN 10 SECONDS

The Command Issuer subroutine generates this error if a command was issued without the register 7 interrupt enabled bit set and none of the attention lines in register 0 gets set within 10 seconds.

NUMBER(HEX) ERROR MESSAGE

C6 C-6 IDLE BIT SET WITHOUT A COMPLETE OR ERROR
REG 0 = XXXX REG 7 = XXXX

The Command Issuer subroutine generates this err if after issuing a command the idle bit in register 0 is set but the complete and/or error bits in register 7 do not get set.

C7 C-7 IDLE DID NOT OCCUR WITHIN 2.5 SECONDS
AFTER AN I/O RESET
REPLACE THE CONTROLLER

The Reset subroutine issued an I/O reset. This forces the controller to do its selftest. This message is printed if after waiting 2.5 seconds for the selftest to complete the idle bit in register 0 is not set. This indicates that the selftest did not pass and therefor the controller board is probably bad and needs to be replaced.

C8 C-8 DATA COMPARE ERROR

CYL	SECT	HEAD	WCNT	EXPT	RECD
XXXX	00XX	00XX	XXXX	XXXX	XXXX

The Print Data Compare Error subroutine generates this error if the expected data and the actual data do not compare.

7.0 PART NUMBERS

PROGRAM DESCRIPTION	2250139-9901 (PD)
SRC, PD ROFF SOURCE	-2001 (SRC,PD)
LIST, PD ROFF LISTING	-9001 (PD)
FICHE KIT	-0009 (SP)

DDFLOP LINKED TEST	-1006 (FLO)
	-9006 (LML)
	-7006 (LC)
DDFLOP BURN IN LINK	-1009 (FLO)
	-9009 (LML)
	-7009 (LC)
TFCONT1, MAIN MODULE	-1003 (OBJ)
	-9003 (LST)
	-2003 (SRC)
TFCONT2	2250976-1001 (OBJ)
	-9001 (LST)
	-2001 (SRC)
TFCONT3	2250977-1001 (OBJ)
	-9001 (LST)
	-2001 (SRC)
TFCONT4	2250978-1001 (OBJ)
	-9001 (LST)
	-2001 (SRC)

TFCONT5	2250979-1001 (OBJ)
	-9001 (LST)
	-2001 (SRC)
TFMSGS, MESSAGE MODULE	2250575-1001 (OBJ)
	-9001 (LST)
	-2001 (SRC)

A

APPLICATION	
NEXT ASSY	USED ON
	7506

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

REV

SHEET

REV STATUS OF SHEETS — REV / SHEET

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± 010
2 PLACE DECIMAL ± 02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL STD 100

DWN — DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group — Dallas, Texas

PD, DS10PD, TEN MEGA-BYTE DISK DIAGNOSTIC-990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	2250113-9901

SCALE — REV: B — SHEET 1 of 39

TI-0023-C

TABLE OF CONTENTS

1.0 SCOPE

This document describes DS10PD, the TILINE controller and ten mega-byte disk test. This test was written to verify the correct operation of the ten mega-byte disk controller and the associated disk unit. This test also has some utility routines which aid the operator in trouble shooting. The structure of this test is such that the operator may run the complete test, separate parts of the test, or any individual test. The diagnostic also has some unique utility routines which aid the operator in trouble shooting the controller and disk drive.

2.0 REFERENCES

In addition to the 990 documents listed in the Diagnostic Handbook the following documents should be referenced.

Ten Mega-byte Disk Controller Specification P/N 937503-9901

Ten Mega-byte Disk Installation
And Operation Manual P/N 946261-9701

Ten Mega-byte Disk Depot Maintenance Manual P/N 946262-9701

3.0 EQUIPMENT AND SOFTWARE REQUIREMENTS

3.1 EQUIPMENT REQUIREMENTS

In addition to the equipment specified in the Diagnostic Handbook the following equipment is required.

Ten Mega-byte Disk Kit P/N 937500

990 Computer with a minimum of 12K words of memory

3.2 SOFTWARE REQUIREMENTS

The diagnostic consists of the following modules.

Linkable Parts : DS10M1
DS10M2
DS10M3
DS10MS

DS10PD, Linked Test P/N 2250113-1006 (FLO)

4.0 LOADING

Loading procedures for all media are contained in the Diagnostic Handbook (945400-9701).

5.0 TEST EXECUTION AND DESCRIPTION

5.1 IT VERB Initialize Test Parameters

The test parameters should be initialized before any other verb is executed. The IT verb may be called at any time to verify or change the parameters. The only other verb that can change any of the test parameters is the CM verb which can change the unit under test.

5.2 EA VERB Execute All Parts

The EA verb executes all parts of the diagnostic on the specified unit. The parts are:

PART 1	QUICK CONTROLLER TEST
PART 2	DISK AND CONTROLLER TEST
PART 3	ADDRESSING TEST
PART 4	MEMORY ADDRESSING TEST
PART 5	MEDIA TEST
PART 6	INTERACTIVE PART

5.2.1 LA VERB Loop on All except interactive part

The LA verb may be used to loop on all parts except the interactive part.

5.3 E1 THROUGH E6 VERBS

Each part of the diagnostic may be executed separately by using the verb corresponding to the part number to be executed. When the specified part has completed, the diagnostic returns to DOCS.

5.4 L1 THROUGH L5 VERBS

Each part of the diagnostic except the interactive part may be looped on.

5.5 ET VERB Execute Subtest

Each part may contain sevral subtest. To execute a subtest only the ET verb may be used. When the ET verb is entered the the verb will then ask for the subtest number. When the subtest has completed control is then returned to DOCS. The subtests associated with each part are as follows.

PART 1	SUBTEST 1-3
PART 2	SUBTEST 4-19
PART 3	SUBTEST 20-22
PART 4	SUBTEST 25
PART 5	SUBTEST 23
PART 6	SUBTEST 24

5.5.1 LT VERB Loop On Subtest

Each subtest may be looped on using the LT verb. When the LT verb is entered the verb will then ask for the subtest number. This does not include part 6.

5.6 CM VERB Check Multiple Units

The CM verb may be used to execute an LA VERB on all available units.

5.7 UTILITY VERBS

The following verbs have been included in the disk diagnostic to aid in finding problems.

VERB	FUNCTION
IC	ISSUE COMMAND
IM	ISSUE MULTIPLE COMMANDS
LO	LOOP ON MULTIPLE COMMANDS
DC	DISPLAY LAST COMMAND
SR	STORE REGISTERS
CD	COMPARE TWO BLOCKS OF DATA
FD	FORMAT DISK
DT	DISPLAY TRACE
AL	DISK ALIGNMENT VERB

5.7.1 IC VERB ISSUE COMMAND

The IC verb may be used to issue a disk command which the operator has created outside the diagnostic or a disk command found in the diagnostic. The only information required by this verb is the memory address of the command and whether the status should be checked after the command has completed.

5.7.2 IM VERB ISSUE MULTIPLE COMMANDS

The IM verb may be used to issue more than one command. The commands must be stacked in memory in the order in which they are to be issued. The information required by this verb is the number of commands, the memory address of the first command and whether status should be checked after each command has completed.

5.7.3 LO VERB LOOP ON MULTIPLE COMMANDS

The LO verb will issue multiple commands the same as the IM verb however when the specified number of commands have been issued the verb starts over with the first command instead of returning to DOCS.

5.7.4 DC VERB DISPLAY LAST COMMAND

The DC verb may be used to display the last command issued with the status returned by the disk and the disk controller.

5.7.5 SR VERB STORE REGISTERS

The SR verb does a Store Registers command then prints out the information returned by the command.

5.7.6 CD VERB COMPARE TWO BLOCKS OF DATA

The CD verb may be used to compare two blocks of data. The required inputs for this verb are the two starting addresses for the compare and the number of words to compare. When a miscompare is found it is reported and the compare continues until the next miscompare is found or the required number of words have been compared.

5.7.7 FD VERB FORMAT DISK

The FD verb may be used to format the whole disk. The selected disk is formatted with the sectors/record = 1 and the word count = maximum number of words allowed when the sectors/record = 1.

5.7.8 DT VERB DISPLAY TRACE

A trace table is kept by the command issuer subroutine of the last 10 disk commands. The DT verb will print the last 10 commands when executed. The commands will be displayed as they were given to the command issuer. The commands are listed in the order they are executed, oldest command at the top of the list.

5.7.9 AL VERB DISK ALIGNMENT VERB

The AL verb will do unformatted reads from the cylinder and head specified. If two read locations are specified the verb will alternate between the two.

5.8 SUBROUTINES

The diagnostic has subroutines which may be called by any test.

5.8.1 COMMAND ISSUER SUBROUTINE

All commands are issued by calling the command issuer subroutine. (CISUER) This subroutine can issue commands with or without interrupts. After the command is issued to the disk controller, the CIS (Command Issuer Subroutine) spins for 20 sec. Or until a disk interrupt occurs.

When the C.I.S. finishes issuing the commands and if the status check flag is set, it calls the status check subroutine (SCHKER) which checks the controller status and the disk status to see if any error bits are set.

5.9 PARTS 1 THROUGH 6

5.9.1 PART 1 QUICK CONTROLLER TEST

Part 1 is a quick test of the controller. If this part of the test fails there is a problem with the controller and the rest of the test will probably not run correctly.

5.9.1.1 SUBTEST 01

All of the bits in the slave logic that can be written and read are tested by writing and reading them under a mask. The 16 patterns are 0, 1111, 2222, ...,FFFF.

5.9.1.2 SUBTEST 02

All of the unit select lines in word 6 of the U.T.C slave registers are set to 0. The disk status is then checked. It should be 'C000' (off line and not ready). A reset command in the 990 is executed then the U.T.C. status is checked. It should be 'A100' (idle, error, and abnormal completion.

5.9.1.3 SUBTEST 03

A store registers command with address 57FE is issued to the U.T.C. and the status is checked after the command. The descriptor parameters read from the U.T.C. are then checked against the descriptor values for the disk. If the descriptor words read do not compare with the list, an error is printed.

5.9.2 PART 2 CONTROLLER AND DISK TEST

Part 2 is a test of the disk and controller.

5.9.2.1 SUBTEST 04

This test issues seek commands to cylinders 0,10,20,30,and 40 and does a restore after each seek. The seeks are not done by the controller but a complete and idle status should be returned. The restores should be done by the controller.

5.9.2.2 SUBTEST 05

This test issues 32 write unformatted commands to different cylinders each followed by a read unformatted and a restore. The data is checked for errors.

5.9.2.3 SUBTEST 06

This test first formats the entire disk, then does an unformatted read, checking the read format against the correct format. Ten reads are performed.

5.9.2.4 SUBTEST 07

This test formats a cylinder, writes data to it, reads the data, and then checks the data. This entire operation is performed on ten different cylinders. Also, a retry check is run ten times. This test is performed by timing a write data command which writes data to all twenty-four sectors of one track. The track is formatted at the max word count.

5.9.2.5 SUBTEST 08

The transfer inhibit bit of the command word is tested by issuing a read command with this bit on and checking that no data was actually transferred. The track is formatted first to insure that this test is independent of other tests.

5.9.2.6 SUBTEST 09

This test causes an ID (header) error (IE) by doing a write format and a read unformatted to get a good header. The test then changes each of the three words of header and CRC one at a time, verifying that the IE bit is set by the controller upon trying to do a write to that address.

5.9.2.7 SUBTEST 10

This test creates two search errors (SE) conditions and checks to see if they are reported by doing a write format of two sectors/record. Do a read of sector one where no header is. A search error should be reported. Then do a write and a read unformatted to eliminate the data sync character. Do a read of sector zero. Again a search error should be reported.

5.9.2.8 SUBTEST 11

This test creates a data error (DE) by doing a write format command of 80 bytes, reading the header and data back,changing the word count to two, and then doing a write unformatted to put the data back on the disk. The test then issues a read command, for two words. A CRC error should result and the DE bit should be set.

5.9.2.9 SUBTEST 12

This test checks the timeout error bit. This test causes a disk operation timeout by attempting to do a read from the maximum sector address plus two. The status word is checked for the TO bit being set.

5.9.2.10 SUBTEST 13

This test checks the seek incomplete. To check the seek incomplete error bit in the disk status do the following:

1. Format the last track on the disk with the sectors/record=maximum and the word count=100 words.
2. Do a write to the last track with the word count=101 words. Check the controller and disk status. The controller status should be A901. Disk status should be 04XX where XX is don't care.
3. After the write a restore is issued to clear the disk.

5.9.2.11 SUBTEST 14

A write command is done with the word count=0 to verify that no data are written and that all of the words in the record are set=0. The commands are:

1. Format track 0 with sectors/record=1 and word count=2.
2. A write is done with the word count=0. (Data=1234)
3. A read with word count=2. The data read back is checked. It should be 0.

5.9.2.12 SUBTEST 15

This test checks the busy flag. In order to test the busy flag a write is done with its TILINE address equal to the address of the controller. Since the controller should be busy while the write is being done, the most significant bit should be set. The commands done are:

1. Track 0 is formatted with word count=12.
2. A write is done with the word count=8 and the tiline address equal to the address of the controller.
3. Eight words are read back. The data is checked. It should be zero.

5.9.2.13 SUBTEST 16

TILINE timeout test does store registers to test for TILINE timeout by using an illegal memory address. This test does not use the command issuer subroutine.

5.9.2.15 SUBTEST 18

The write amplifier test formats the disk for a write of 4096 words (cause the controller to switch heads). Write 4096 words then check for errors.

5.9.2.16 SUBTEST 19

The cylinder switch test verifies that the disk controller does an auto increment from cylinder >FF to cylinder >100. First format both tracks. Then issue a write unformatted command to clean track 0. Then do a write to track >FF, head one, of enough words to cause an auto increment. Check status.

5.9.3 PART 3 ADDRESSING TEST

5.9.3.1 SUBTEST 20

Test 20 verifies that the controller can address all of the sectors on one track correctly. To do this it:

1. Formats track 0 with sectors/record=1 and word count=8.
2. Eight words of data are written to each sector on the track. The write is done to the last sector first then the sector address is decremented to 0. The reason for this is that if a write modifies the next sector the next write wont cover up the error. The data word used is the sectors/record and sector address. The data is read back and checked. The read starts at the first sector on track 0.
3. Sestor 15 (21 decimal) is addressed to check for improper sector selection in the disk drive or an incorrect cartridge.

5.9.3.2 SUBTEST 21

Test 21 verifies that the controller can address every track on the disk. To do this it:

1. Formats the whole disk with sectors/record=1 and words/record=8.
2. Eight words of data are written to the first record on each track. The data is equal to the cylinder address.
3. The data is read back and checked. In reading the data back, the test starts at the middle cylinder, then increments by one decrements by two, increments by three, etc., until all cylinders are read.
4. 1000 hexidecimal random seeks are performed on each head.

5.9.3.3 SUBTEST 22

Test 22 verifies that the controller can do variable size sectors per record and auto increments. To do this it:

1. Formats the first 25 tracks with the sectors/record=x (where x goes from 1 to the maximum allowed) and the word count is 1 word per record. The data used is the sectors/record.
2. A read is then done with the word count equal to 50 hex words and the data read is checked. If an error is found a message is printed.
3. The sectors/record count is then incremented and the test is done again until the sectors/record count=maximum.

5.9.4 PART 4 MEMORY ADDRESSING TEST

This part contains only one test (24).

5.9.4.1 SUBTEST 25

This test checks the ability of the disk controller to address every memory location on the TILINE. Part 4A uses memory that is not used for storing the diagnostic as its buffer. If there is less than 4K words left after loading the diagnostic the test would be meaningless, therefore the test will output a message of 'INSUFFICENT MEMORY' and the test will be skipped. If enough memory is available then the test writes the address at the address in each buffer location. The buffer is then written to cylinder 0 sector 0 of the disk. The buffer is cleared. Then the data is read back into the buffer from the disk. Each location of the buffer is then verified that it contains its address.

The map memory addressing (PART 4B)is tested in the same manner as above except that the buffer is only 80 bytes long on each 4K word boundary and the test data is the processor address and map bias used by the processor to address the buffer. If part 4B is skipped then it means that the computer does not have the memory mapping option.

5.9.5 PART 5 MEDIA TEST

5.9.5.1 TEST 23

This test uses the CRC character to verify all of the data of the disk. The way it does this is to:

1. Formats the whole disk with the sectors/record=maximum (1 record/track) and the word count=maximum.
2. A read with the word count=1 is done from each track. The word that is read is checked and if a bit is bad anywhere on the track a status error should be reported with the DE error bit set. If the data word that is read back is incorrect the read is executed 10 times and the number of failures in 10 tries is added to the data error count. The reason that all of the words are not just read back from the disk and checked is because it would take a very long buffer to accomodate the data. The disk is formatted and read 4 times with different data each time. The 4 data patterns used are 0000, FFFF, AAAA, and 5555. When the disk is being formatted each track gets different data pattern from the track before. The data used goes as follows:

LOOP	TRACK	DATA
1	0	0
	1	FFFF
	2	AAAA

	3	5555
	4	0000
	:	:
	MAX	
2	0	FFFF
	1	AAAA
	2	5555
	3	0000
	4	FFFF
	:	:
	MAX	
3	0	AAAA
	1	5555
	2	0000
	3	FFFF
	4	AAAA
	:	:
	MAX	
4	0	5555
	1	0000
	2	FFFF
	3	AAAA
	4	5555
	:	:
	MAX	

5.9.6 PART 6 INTERACTIVE TEST

5.9.6.1 TEST 24

This test requires the operators intervention so there are no verbs to loop on this test. This test uses the data from test 23 so it is necessary to have run test 23 before test 24. When the test starts execution it asks the operator if part 5 (test 23) has been run. If the operator enters a 0 (no) a message will be output asking him to run part 5. If a 1 (yes) is entered the test will output a message asking if the power has been cycled. If the answer is 0 (no) the test will output a message telling the operator to cycle power. To cycle power first turn off power to the computer then turn off power to the disk then turn power on to the disk and then turn power on to the computer. If the answer is 1 (yes) the test verifies the data on the disk has not been modified by the power cycle.

6.0 MESSAGES

'DSK10MBT 990 TEN MEGA-BYTE DISK TEST VERSION MO/YR RV'

The first message printed by the diagnostic contains the month and year of the last revision.

6.1 HEADER MESSAGES

'MESSAGE' COMMENTS

6.1.1 MESSAGES TO INITIALIZE THE TEST

'DISK TILINE ADDRESS D=F800' Disk tiline address.

'DISK INTERRUPT LEVEL D= 0D' Disk interrupt level.

'UNIT TO TEST 00' Disk unit number.

'OTHER AVAILABLE UNITS(0-3 TERMINATE WITH FF)FF' Other units.

'MAX ERROR COUNT EQUAL - FFFF' Maximum error number.

'INSURE THAT ALL DISC PACKS ARE WRITE' Reminder message.
'PROTECTED OR ARE SCRATCH PACKS'

6.1.2 MESSAGES FOR IM VERB

'ADDR' Used by this and other verbs to get an address.

' COMMANDS' Get number of commands.

6.1.3 MESSAGES FOR LC VERB

' COMMANDS' Get the number of commands.

'CHECK ST? ' Do you want to check the disk and controller status.

6.1.4 MESSAGES EXECUTE TESTS

'TEST - XXXX' Get test number to loop on.

'TEST DONE' Trailer message.

'UNIT XX UNDER TEST ' The CM verb is now testing another unit.

'LOOP COUNT =XXXX' Output after each pass through loop.

'ERROR COUNTS' Output after each pass through loop and at end.

'DATA =XXXX TIMING=XXXX STATUS=XXXX'

6.1.5 MESSAGES FOR COMPARE DATA VERB (CD)

'MISCOMPARE AT XXXX AND XXXX'

'TOTAL WORD COUNT= XXXX'

6.1.6 MESSAGE FOR STORE REGISTERS VERB (SR)

'SECT/TRACK =XX OVERHEAD/RECORD =XX'

'TRACKS/CYL =XX CYL/DRIVE =XXXX'

6.1.7 MESSAGE FROM SELF TEST

'SELF TEST STATUS = XXXX'

6.1.8 MESSAGES FROM VERB TO ISSUE COMMANDS (IC)(IM)

'ISSUEING COMMAND NOW'

6.1.9 HEADER AND TRAILERS FOR EACH PART

'PART 1'

'PART 1 DONE '

'PART 2'

'PART 2 DONE '

'PART 3'

'PART 3 DONE'

'PART 4'

'INSUFFICENT MEMORY' Part aborted end message.

'PART 4 DONE '

'PART 4A DONE'

'PART 4B SKIPPED ' No mapped memory found to test.

'PART 5'

'PART 5 DONE '

'PART 6'

'PART 6 DONE '

'WAS PART 5 RUN LAST?'

'RUN PART 5'

'HAS SYSTEM POWER BEEN CYCLED? '

'CYCLE POWER '

6.1.10 MESSAGES FOR ALIGNMENT VERB

'NUMBER OF DIFFERENT READS? 1 OR 2?'

'CHECK STATUS? '

'SEEK TO CYLINDER NUMBER?'

'SURFACE NUMBER? '

6.2 ERROR MESSAGES

When an error occurs in execution of any part of the test, an error message will be printed out, providing that the print error flag has been set. The error mesage will describe what type of error and also inform the operator the test number in which the error occured. In some cases errors will be incurred while using the utility verbs. These verbs cause the test number in the error message print-out to become all 'F's.

NUMBER 'ERROR MESSAGE' COMMENTS

0 'CONTROLLER STATUS '

'DISK STAT COMM SA S/R R A (continued)

'XXXX XX XX XX XX (continued)

CYL A BYTE C MEMADD SEL CONT STAT'

XX XXXX XXXXXX XX XXXX '

The above message is a continuation to many error messages where the specific disk command and the status of the disk or controller is needed. The data listed in this message is taken from the tiline peripheral control space after the disk instruction terminates.

01 '*ERROR* THE UTC DID NOT RESPOND IN 20 SEC ' The command issuer subroutine timed out waiting for a response from the disk controller.

02 'UNEX DISK INT. AT XXXX' The disk interrupt level did not match the level given in the IT verb.

03 'ERROR IN TEST - XXXX'
'CONTROLLER STATUS COMP=X ERR=X IDLE= X'

04 'MAX ERROR COUNT EXCEEDED, STOP' The maximum error count specified in the IT verb has been reached.

05 'TEST =XXXX IS NOT CURRENTLY AVAILABLE ' The operator has specified a test that the diagnostic does not contain.

06 'UTC REG DATA TEST '
'WAS EXP REG.'

07 'XXXX XXXX XX' Device register on the disk controller failed to retain data stored in it.

08 'ERROR R6=0 STATUS SHOULD BE COXO WAS XXXX' With no unit selected the disk status should report the disk off line and not ready.

09 'R7 SHOULD BE A100 WAS XXXX AFTER IO RESET' Expected disk status; idle, error, and abnormal comltetion because of IO reset.

10 'STORE REG ERROR ' This message will be continued with a report of the command and error status.

11 'SELF TEST ERROR '
'REG TWO SHOULD READ 0000'
'REG TWO WAS --XXXX'
'PLEASE CONSULT MAINTENANCE MANUAL '
'BEFORE CONTINUEING THE DIAGNOSTIC '

12 'LOCK OUT NOT SET AFTER 2 READS ' The disk controller failed to set the lockout bit of the controller status after the status had been read while the controller was idle.

13 'DATA ERROR'
'DATA READ WAS - XXXX'
14 'DATA SHOULD HAVE READ - XXXX'

```
15 'WRITE FORMAT ERROR'
   'FORMAT READ WAS - '
16 'XXXX XXXX XXXX XXXX'
17 'FORMAT SHOULD HAVE READ - '
16 'XXXX XXXX XXXX XXXX'

18 'RETRY ERROR - RETRIED WHEN NOT REQUIRED'

19 'TRANSFER INHIBIT ERROR '
   'DATA  RECIEVED  WAS  XXXX  ,  SHOULD  BE  XXXX'  Data
   transferred  on  a  read  instruction  with  inhibit  in
   force.

20 'ID TEST ERROR'
   'REG 7 SHOULD READ A810 BUT WAS - XXXX'

21 'SE TEST ERROR'
   'REG 7 SHOULD READ A802 BUT WAS -XXXX'

22 'DE TEST ERROR'
   'REG 7 SHOULD READ A840 BUT WAS -XXXX'

23 'ERROR PART 2 XX TEST '
   'STATUS EXP= XXXX STATUS REC=XXXX'
```

```
24 'ERROR PART 2 SI TEST '
   'R0   STATUS  EXP=04XX  REC=  XXXX  R7  STATUS  EXP=A801
   REC=XXXX'

25 'ERROR PART 2 '
   'DID WRITE WITH WC=0 BUT DATA NOT 0'

        26 'ERROR PART 2 BUSY FLAG SET '
   'DATA (MSB SHOULD BE 0)'

27 'XXXX'

28 'TILINE TIMEOUT ERROR'
   'R7 STATUS EXP " A820 REC= XXXX'

29 EXPECTED RATE ERROR, DID NOT RECIEVE IT'
   'R7 STATUS EXP= A808 REC = XXXX'

30 'WRITE AMP FAILURE '
   'R7 STATUS EXP = C800 REC = XXXX'

31 'ERROR PART 2N CONTROLLER DID NOT SWITCH'
   'FROM MAX HEAD CYL FF TO HEAD 0 CYL 100'
```

32 'DATA ERROR IN SECTOR - XXXX'

'DATA READ WAS - XXXX'

33 'SECTOR COUNT ERROR'

'ACCESSED SECTOR GREATER THAN 20' Disks with more than 20 sectors will fit this drive. Insure that the disk cartridge is the correct one for this drive.

34 'ERROR IN CYL ADDRESSING TEST'

'AT CYL -XXXX RECEIVED XXXX FOR CYL ADDR'

35 'ERROR IN VARIABLE FORMAT TEST'

'AT -XXXX SECTORS PER SECOND'

'RECIEVED - XXXX AS DATA'

36 'PART 4 DATA NOT EQUAL TO ADDRESS'

'ADDRESS=XXXX'

'DATA= XXXX Part A of the memory addressing test uses the address as data to test memory addressing.

37 'COMPARRISON ERROR IN MAP MEMORY '

'MAP ADDRESS=XXXX'

'MAP BIAS= XXXX'

'TILINE ADDRESS= XXXXXX'

'EXPECTED DATA=XXXX'

'DATA READ=XXXX' Part B of the memory addressing test uses the address and the bias as data to test map memory addressing.

38 'XX ERRORS IN A TRIES '

39 'ERROR PART 5 '

'DID FORMAT WITH:'

'WC= XXXX S/R= XX & DATA=XXXX DATA READ= XXXX'

'FROM CYL=XXXX SURF=XX'

40 'PART 5 LOOP ERROR '

41 'ERROR DATA READ XXXX EXP XXXX FROM SURF XX CYL XXXX'

46 '**STATUS ERROR'

7.0 PART NUMBERS

PROGRAM DESCRIPTION	2250113-9901 (PD)
PROGRAM DESCRIPTION ROFF SOURCE	-2001 (SRC,PD)
PROGRAM DESCRIPTION ROFF OUTPUT	-9001 (ANSI,PD)
Fiche Kit	-0009 (SP)
DS10PD, Linked Test	-1006 (FLO)
	-9006 (LML)
	-7006 (LC)
DS10M1,TEST MODULE ONE	-1003 (OBJ)
	-9003 (LIST)
	-2003 (SRC)
DS10M2,TEST MODULE TWO	939661-1003 (OBJ)
	-9003 (LIST)
	-2003 (SRC)
DS10M3,TEST MODULE ONE	939662-1003 (OBJ)
	-9003 (LIST)
	-2003 (SRC)
DS10MS,TEST MESSAGE MODULE	2250262-1003 (OBJ)
	-9003 (LIST)
	-2003 (SRC)

A

APPLICATION	
NEXT ASSY	USED ON
	7506

REVISIONS

LTR	DESCRIPTION	DATE	APPROVED

REV
SHEET

REV STATUS OF SHEETS — REV / SHEET

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± 010
2 PLACE DECIMAL ± 02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL STD 100

DWN DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group *Dallas, Texas*

PD, EMU900, 9900/9980 EMULATOR TEST-990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	2250229-9901

SCALE	REV.	SHEET

TI-0023 C

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

TABLE OF CONTENTS

1.0 SCOPE

The 9900/9980 Emulator and Buffer Diagnostic (P/N 2250211-1006) tests the 9900/9980 Emulator (P/N 941555-0001) and one of the following buffer board combinations: TMS9900-1 Buffer Board (P/N 949973-0001) or Buffer Module Assembly (P/N 941560-0001); SBP9900A Buffer Board (P/N 9949972-0001) or Buffer Module Assembly (P/N 941550-0001); or the TMS9980A/81 Buffer Board (P/N 949974-0001) or Buffer Module Assembly (P/N 941565-0001) depending on which one is connected. The test exercises all functions of the 9900/9980 Emulator and its associated Buffer Board.

The test also includes a verb ('EM') which completely tests the Expansion Memory Board when used.

1.1 Software Description

The 9900/9980 Emulator Diagnostic is a test that runs under control of DDCS. If I/O is through the front panel, the error message number will be displayed on the front panel.

Note, that the Burn-In version of this diagnostic does not test the Expansion Memory Board. This test is available as EXPRAMBI (P/N 2250237-1003). In addition, the Burn-In version does not test the Target Cru Interface, or the Trace Module/Emulator Interface.

2.0 REFERENCES

990 Diagnostics Handbook P/N 945400-9701

Specification, 990 Diagnostic Standards

Specification, 9900/9980 Emulator Module P/N 941552-9901

Specification, TMS9900 Buffer Module P/N 941557-9901

Specification, SBP9900 Buffer Module P/N 941547-9901

Specification, TMS9980/81 Buffer Module P/N 941562-9901

Specification, Trace Module

3.0 EQUIPMENT AND SOFTWARE REQUIREMENTS

3.1 Suggested Equipment

FS990/4 or FS990/10 system

9900/9980 Emulator Module, P/N 941555-0001

Trace Module, P/N 949910-0001

One of the following is also required:

TMS 9900-1 Buffer Module, P/N 949973-0001

SBP 9900A Buffer Module, P/N 949972-0001

TMS9980A/81 Buffer Module, P/N 949974-0001

3.2 Linking Information

EMU900, Standard Link: 2250229-1006 (FLO)

Linkable Parts: UNIEMU

EMUMSG

UNIACT

ACTMSG

UNIINT

INTMSG

UNIMEM

MEMMSG

UNISPD

SPDMSG

UNIBIT

BITMSG

```
UNIREG
REGMSG
UNILUB
LUBMSG
UNIBPR
BPRMSG
UNITRC
TRCMSG
UNIBPC
BPCMSG
UNICPU
CPUMSG
DMSTRT
AU04A
DMEND
UNIETI
ETIMSG
UNITAR
TARMSG
UNIEXP
EXPMSG
DBGVRB
DBGMSG
UNIEND
```

4.0 LOADING AND INITIALIZATION

4.1 Loading

Loading instructions for available load media may be found in the 990 Diagnostics Handbook.

4.2 Initialization

4.2.1 The initialization questions asked by the diagnostic are listed below. For DOCS initialization, refer to the 990 Diagnostics Handbook.

```
ENTER THE TRACE MODULE BASE ADDRESS  DEFAULT=0100 -
ENTER THE EMULATOR CRU BASE ADDRESS  DEFAULT=0140 -
****  WARNING  ****
EMULATOR AND I/O DEVICE INTERRUPT LEVELS MUST NOT BE THE SAME
ENTER THE EMULATOR INT. LEVEL  DEFAULT=0006 -
```

4.3 After the initialization questions are answered, the Diagnostic then performs some initialization procedures. The following messages appear before each procedure is performed.

```
SELECTING EMULATOR CLOCK, EXECUTING RESTART
INITIALIZING EMULATOR W/"IE"
```

5.0 9900/9980 TEST VERBS

Please note that only the EMULATOR and a BUFFER BOARD should be connected for execute verbs E1-E8, and loop verbs L1-L8. The TRACE MODULE should not be connected during these tests or error messages will result. The TRACE MODULE should be connected only during the E9 Verb (Emulator/Trace Module Interface Test) and its associated loop verb, L9. For the TS and associated loop verb L0, only an EMULATOR, BUFFER BOARD, and TARGET SYSTEM should be connected. Also, for the EM and associated LE verbs, only an EMULATOR, BUFFER BOARD and EXPANSION MEMORY BOARD should be connected.

In the following verb descriptions the following definitions apply:

Prototype Memory refers collectively to either the on-board Emulator 8KB RAM, or to the Target system RAM.

Control Memory refers strictly to the Emulator 512 byte word on-board RAM.

In many of the following tests, in order to check out a particular Emulator function, a small test procedure must be loaded into Prototype memory (either Target or Emulator) and executed. If for any reason the test does not complete execution or the Emulator does not indicate completion an error message such as the following might

be output: EMULATION FAILED or EMULATOR FAILED TO INDICATE IDLE EXECUTION. The reason for the latter message is that many test procedures end with an IDLE instruction.

HLTACK- Halt Acknowledge: Bit 16 on the HOST/EMU CRU-IN interface

EMREDY- Emu Ready: Bit 17 on the HOST/EMU CRU-IN interface

HSTRED- Host Ready: Bit 17 on the HOST/EMU CRU-OUT interface

COMCMP- Command Complete: Bit 18 on the HOST/EMU CRU-IN interface

COMMND- Command bit: Bit 18 on the HOST/EMU CRU-OUT interface

IDLE- Idle Bar bit: Bit 21 on the HOST/EMU CRU-IN interface

INTPEN- Interrupt Pending: Bit 23 on the HOST/EMU CRU-IN interface

INTENA- Interrupt Enable: Bit 23 on the HOST/EMU CRU-OUT interface

The following is a list of all Interactive and Debug Verbs contained in this diagnostic.

IT-INITIALIZE TEST

L1-LOOP ON HOST/EMULATOR INTERFACE TEST

L2-LOOP ON MEMORY TESTS EXCEPT SPEED TESTS

L3-LOOP ON BITS REGISTER TEST

L4-LOOP ON LOWER AND UPPER BOUNDS HARDWARE TEST

L5-LOOP ON BREAKPOINT HARDWARE REGISTER TEST

L6-LOOP ON TRACE FUNCTION AND HARDWARE TESTS

L7-LOOP ON BREAKPOINT FUNCTION TESTS

L8-LOOP ON CPU TEST

L9-LOOP ON EMULATOR/TRACE MODULE INTERFACE TEST

LO-LOOP ON BUFFER/TARGET SYSTEM INTERFACE TEST

LE-LOOP ON EXPANSION MEMORY TEST

LW-LOOP ON EXPANSION MEMORY WRITE PROTECT REGISTER TEST

S1-LOOP ON PROTOTYPE MEMORY SPEED TEST

S2-LOOP ON CONTROL MEMORY SPEED TEST

LA-LOOP ON PARTS 1-8

E1-EXECUTE HOST/EMULATOR INTERFACE TEST

E2-EXECUTE MEMORY TESTS

E3-EXECUTE BITS REGISTER TEST

E4-EXECUTE LOWER AND UPPER BOUNDS HARDWARE TEST

E5-EXECUTE BREAKPOINT HARDWARE REGISTER TEST

E6-EXECUTE TRACE FUNCTION AND HARDWARE TESTS

E7-EXECUTE BREAKPOINT FUNCTION TESTS

E8-EXECUTE CPU TEST

E9-EXECUTE EMULATOR/TRACE MODULE INTERFACE TEST

TS-EXECUTE BUFFER/TARGET SYSTEM INTERFACE TEST

EM-EXPANSION MEMORY BOARD TEST

EW-EXPANSION MEMORY WRITE PROTECT REGISTER TEST

EA-EXECUTE PARTS 1-8

HT-HALT EMULATOR

RS-RESET EMULATOR

SC-SELECT CLOCK

GC-GENERATE COMMAND

XT-EXECUTE COMMAND TABLE

LT-LOOP ON EXECUTE COMMAND TABLE

SL-SHOW COMMAND TABLE LIST

IE-INITIALIZE EMULATOR

SM-SELECT MEMORY (EMULATOR OR TARGET)

CK-READ CLOCK SELECTION

RR-READ PROTOTYPE CPU REGISTERS

WR-WRITE PROTOTYPE CPU REGISTERS

RM-READ MEMORY

WM-WRITE MEMORY

RC-READ TARGET SYSTEM CRU FIELD

WC-WRITE TARGET SYSTEM CRU FIELD

DT-DEFINE TRACE CONFIGURATION

DB-DEFINE BREAKPOINT CONFIGURATION

RT-READ TRACE SAMPLES

RE-RUN EMULATION

CP-RUN PROGRAM IN CONTROL RAM

PC-SELECT PROTOTYPE OR CONTROL MEMORY

PE-PRINT ERROR COUNT

DS-DUMP EMULATOR STATUS

LB-LOAD CRU BASE

LD-LOAD CRU

ST-STORE CRU

SO-SET CRU BIT TO ONE

SZ-SET CRU BIT TO ZERO

BT-TEST CRU BIT

BS-BIT SAMPLE

HD-CONVERT HEX TO DECIMAL

DH-CONVERT DECIMAL TO HEX

BP-SET BREAKPOINT

CB-CLEAR BREAKPOINT

BL-LOOP ON SET BIT

LO-LOOP ON ONE

LZ-LOOP ON ZERO

5.1 IT Verb

The IT Verb initializes the Emulator by setting up both the Trace Module and Emulator base addresses; setting up the Emulator interrupt level; selecting Emulator internal clock as the system clock; executing a forced Restart; and initializing the Emulator with a "IE" (Initialize Emulator) command which also selects Emulator 8KB Memory. The Diagnostic continues to use Emulator 8KB as its memory throughout all the tests until the user changes to Target memory with the "SM" command (Select Memory) as described in Section 5.12. If the Emulator

cannot successfully select Emulator 8KB Memory, a warning message is output.

5.2 Loop Verbs

5.2.1 L1 Verb

L1 loops on Part 1 (HOST/EMULATOR INTERFACE TEST).

5.2.2 L2 Verb

L2 loops on Part 2 (MEMORY TESTS) except speed tests.

5.2.3 L3 Verb

L3 loops on Part 3 (BITS REGISTER TEST).

5.2.4 L4 Verb

L4 loops on Part 4 (LOWER AND UPPER BOUNDS HARDWARE REGISTER TESTS).

5.2.5 L5 Verb

L5 loops on Part 5 (BREAKPOINT HARDWARE REGISTER TEST).

5.2.6 L6 Verb

L6 loops on Part 6 (TRACE FUNCTION AND HARDWARE TESTS).

5.2.7 L7 Verb

L7 loops on Part 7 (BREAKPOINT FUNCTION TESTS).

5.2.8 L8 Verb

L8 loops on Part 8 (CPU TEST).

5.2.9 L9 Verb

L9 loops on Part 9 (EMULATOR/TRACE MODULE INTERFACE TEST).

5.2.10 L0 Verb

L0 loops on Part 10 (BUFFER/TARGET SYSTEM INTERFACE TEST).

5.2.11 LE Verb

LE LOOPS ON THE EXPANSION MEMORY TEST.

5.2.12 LW Verb

LW LOOPS ON THE EXPANSION MEMORY WRITE PROTECT register test.

5.2.13 S1 Verb

S1 loops on Part 2A (PROTOTYPE MEMORY SPEED TEST).

5.2.14 S2 Verb

S2 loops on Part 2B (CONTROL MEMORY SPEED TEST).

5.2.15 LA Verb

LA executes Parts 1, 2, 3, 4, 5, 6, 7 and 8 including the memory speed tests in that order and loops on that sequence.

5.3 EXECUTE VERBS

5.3.1 E1 Verb

E1 executes test Part 1. Part 1 tests the 16 Bit Data Field on the Host/Emulator CRU interface; interrupt capability; entry into and exit from run mode; and all control lines associated with each of the above operations. This test does not check out the 8 Bit Command Field.

The test begins by executing a RESTART command. If the control program is operating properly the pattern >4869 (Ascii representation for "Hi") will appear on the Host CRU-IN Field. This is the first and most basic check of the Control Program and the CRU interface. Next an invalid command opcode is output to the Control Program to see whether it will properly respond by raising COMCMP

(Command Complete) high which indicates an error condition. Then a RESTART command is executed to force the Emulator to the "top" (meaning the beginning) of the Control Program, the communications protocall is then partially executed by raising HSTRED (Host Ready) high and waiting for EMREDY (Emu Ready) to go low. For a detailed explanation of the Control Program and the communications protocall refer to the 9900/9980 EMULATOR HARDWARE SPECIFICATION P/N 941552-0001. In this partially executed mode the Control Program will echo any pattern that it sees on the Host CRU-OUT back to the Host CRU-IN. The test uses this feature and writes patterns of >FFFF, >8000, >4000, >2000, >1000, >800, ...>0000 to the CRU-OUT and checks for them on the CRU-IN.

The next portion of the test checks out the interrupt processing; the IDLE bit present at CRU-IN; and whether or not the Emulator can enter into and exit the RUN mode successfully. First of all, INTPEN (Interrupt Pending) is cleared so that the emulator will recognize a new interrupt. If the INTPEN bit will not clear, an error message is printed and the the entire test will be aborted. Once this is successfully accomplished, HSTRED is set high which causes EMREDY to go low. When EMREDY goes low this causes an interrupt to be generated. The interrupt should be present at INTPEN although in this

case this will not create a CRU interrupt since CRU interrupts have been previously masked off with INTENA (Interrupt Enable). If INTPEN does not indicate an interrupt present an error message will be printed out. Next, an interrupt is again generated using EMREDY but this time the CRU interrupts are not masked off. This will cause the test to process the interrupt through its interrupt processor and if for any reason the interrupt is not processed, an error message will be printed. Next, PROTOTYPE memory is filled with JMP $ instructions and an IDLE instruction is placed at address >500. The PC is then set to >500 and the program is run. During this operation, HLTACK (Halt Acknowledge), and IDLE (IDLE Bit) are checked for proper operation. Finally, the interrupt feature is checked once more to see that HLTACK creates an interrupt when the Emulator switches from RUN mode to CONTROL mode. This concludes the E1 Interface Test.

5.3.2 E2 Verb

E2 executes test Part 2. Part 2 tests Emulator or Target Memory (depending on which has been previously selected with the "SM"-Select Memory command) and Control Memory. Both sets of memory are tested with a Pattern Test, an Address Test and a Speed Test. The verb requests beginning and ending test addresses and uses these bounds

in the Pattern and Address tests. This feature of the Memory test can be used to check out the Expansion Memory Board. The initial beginning and ending bounds at startup of the test are from >0000 to >1FFE which would include only 8KB of Prototype memory, either Emulator or Target. Once the bounds are modified they remain as modified throughout all loop and execute verbs using these memory tests. The Speed tests do not use the same bounds as the Pattern and Address tests but instead check memory in blocks of either 0-8KB, 0-16KB or 0-64KB depending on whether the Buffer connected is a TMS9980A/81 and depending on whether the beginning address is above a certain 8KB boundary.

The Pattern Test is the same for both Prototype (EMU 8KB or Target) and Control Memory. It consists of writing and reading specific patterns (>8000, >4000, >2000 ,...., >0000, >FFFF) to each separate memory location. Each pattern will be written to all memory in one operation and then read back in one operation until all patterns have been used. If any bit errors occur writing or reading memory or if there are any Data Bus problems, this test should find them.

The Address Test will be the same in both Control and Prototype memory. In this test each memory location is loaded with values that are equal to the address being

accessed. In this way the address lines are tested to see if a value is written into the wrong location. If an error occurs, the address of the error, expected data and actual data are returned.

The type of procedure used for the Speed Test is the same for both Control and Prototype memory. The test consists of downloading a test procedure into lower memory which successively writes and reads alternating "1's" and "0's" to each memory location, testing all memory locations one at a time. The procedure starts out by filling memory with >AAAA'S, then, beginning with the first available memory location it writes >5555 to that location and immediatedly reads it back, shifts it left by one and writes it again to the same location, reads it back again, shifts it to the right this time and writes it once again to the same location. The final result should be that memory will be filled with >5555's. If any speed errors occur during this test they will propagate through and the final value will be different from >5555. After the test is through running the Diagnostic reads all memory and reports any errors along with the address of the error and the actual and expected values.

5.3.3 E3 Verb

E3 executes test Part 3. Part 3 tests the Hardware Bits Register by using two test subroutines (RIPTST and

WRTPAT) and by exercising all the verb commands which set and reset each of the bit locations on the register.

RIPTST (Ripple Pattern Test) is a subroutine used in this and subsequent E Verbs which writes and reads various patterns (>8000, >4000, >2000, >1000, ..., >0000) to a selected hardware register, reporting any errors and displaying both expected and actual data patterns. This test should locate a bad bit location in a register.

WRTPAT (Pattern Test) is another subroutine which writes and reads a set of patterns to a selected hardware register. The patterns used in this test are >FFFF, >AAAA, >5555, and >0000. This subroutine reports errors in a manner similar to that used in RIPTST.

Following these tests each individual bit position on the register is set and checked using the particular control program command associated with it. For instance, the "DT" (Define Trace) command is used to set bits 12-15 on the register, and the "DB" (Define Breakpoint) command is used to set bits 5-7 depending upon the conditions specified by the command. Using both these hardware and software methods of testing, the complete operation of the Bits Register can be checked out.

Note: When executing this test with Expansion Memory plugged into the Emulator, an error will be returned that says that the EMU 4K Select Bit has failed. This is

perfectly normal and should be ignored. This error should not appear when Expansion Memory is not being used.

5.3.4 E4 Verb

E4 executes test Part 4. Part 4 tests the Lower and Upper Trace Bounds Registers. Both registers are tested identically. The registers are first tested with "RIPTST" (Ripple Test) and "WRTPAT" (Pattern Test) as described in Section 5.3.3 above. Finally the patterns >55AA and >AA55 are written and read successively to each of the registers using the "DT" (Define Trace) command. This is accomplished by first setting up the "DT" command and then executing a "RE" (Run Emulation) command. As soon as the "RE" command is executed the Control Program sets up all necessary hardware registers including both Upper and Lower Bounds Registers with the previously specified values. In this way both the hardware and the software functions of the registers are tested.

5.3.5 E5 Verb

E5 executes test Part 5. Part 5 tests both Breakpoint Hardware Registers A and B. Both registers are tested identically. The registers are first tested with "RIPTST" (Ripple Test) and "WRTPAT" (Pattern Test) as described in Section 5.3.3 above. Finally the patterns >55AA and >AA55 are written and read successively to each

of the registers using the "DB" (Define Breakpoint) command. This is accomplished by first setting up the "DB" command and then executing a "RE" (Run Emulation) command. As soon as the "RE" command is executed the Control Program sets up all necessary hardware registers including both Upper and Lower Bounds Registers with the previously specified values. In this way both the hardware and the software functions of the registers are tested.

5.3.6 E6 Verb

E6 executes test Part 6. Part 6 tests out the complete operation of the tracing functions of the Emulator with the exception of the Trace Module/Emulator interface. This test consists of five parts: a trace counter test, bounds comparison circuitry test, trace memory address and data lines test, diagnostic memory test, and diagnostic memory speed test.

The trace counter test checks to see that the Emulator will make simple traces in Prototype memory in the MA mode. Memory is filled with JMP $ instructions; an IDLE instruction is placed in memory and a "RE" is executed. The result should be that the IDLE bit should become active. When this occurs the test halts the Emulator and reads the value in the trace counter which should be equal to the original value loaded-1. This same test is

repeated using trace counter values of >01, >02, >04,...,>80. When a value of >01 is used the trace counter will decrement through 0 and will immediately halt. This, therefore, tests out the Emulator's ability to halt when the trace memory full (TMF) condition is met.

The bounds comparison circuitry test consists of three parts which together check out completely the bounds comparison circuitry. For the first part, a test procedure is downloaded into Prototype memory after filling memory with JMP $ instructions. The procedure accesses one memory location starting with >8000->0000 dividing the memory value by 2 each time. In this way all bits of the compare registers are checked out. After a memory location is accessed the Emulator IDLE'S, is halted and the value in the trace counter is compared to its expected value. The trace counter is always initialized at >FF, therefore, the expected value will always be >FE. The downloaded procedure is executed in the "MA" mode with the upper bound=lower bound=>8000, >4000, >2000,...,>0000, and the workspace pointer is always set equal to the upper bound=lower bound. The trace counter is always reset to >FF. The second part of the test checks out the other half of the output lines (A>B, address>bound) on the lower bound address

comparators. It does so by setting the lower bound to >30, the upper bound to 0 and placing the test procedure in the excluded range (0-30), then a procedure is executed which accesses memory above the lower bound of >30 thus causing the A>B lines to toggle. The fact that the code executes beneath the lower bound range and above the upper bound range automatically causes the A<B outputs on the lower bound comparators and the A>B outputs on the upper bounds comparators to toggle so that there is no need for a separate test for those output lines. In a similar manner the third part of the test checks out the A<B lines on the upper bound comparators by accessing memory below the upper bound.

The trace memory address and data lines test checks out the address and data lines of the 512 byte trace RAMS. In the address lines test, memory is first filled with >FFFE'S and then a 0 is placed at one location in memory. Next the entire trace RAM is read to see if the 0 appears anywhere else in memory. This is done for each location in memory. The data lines test consists of executing a type of pattern test using patterns of >8000, >4000, >2000,...,>0000.

The diagnostic memory address and data lines test checks out the address and data lines of the diagnostic RAM along with four of the input lines of the 2-to-1 data

selector, by outputing the patterns 0, 2, 4, >10, >14, >A, and >1E. These patterns are used instead of 0, 5, >A and >F because this particular RAM and multiplexer traces data lines 11 through 14 instead of 12 through 15 which means that the test patterns must be shifted one place to the left in order to appear as 0, 5, >A, and >F. The test also checks to see that the Emulator will function in the PC and EX modes. This is accomplished by downloading and executing a program which traces a known number of samples in each of these modes. After the procedure is executed the trace counter is checked for the correct final value.

The diagnostic memory speed test uses the two control lines IAQ and DBIN. This test checks the ability of the Emulator to trace the processor's control lines IAQ and DBIN which feed into the trace memory through the 2-to-1 data selector. It does so by loading and executing a test procedure which toggles these control lines at a high rate of speed. The test procedures contain instructions selected because of their particular read and write machine cycles. For instance, in testing the DBIN line an instruction is needed that has a read cycle followed closely by a write cycle.

5.3.7 E7 Verb

E7 executes test Part 7. Part 7 tests out the Breakpoint

Mode Functions and Breakpoint Comparator circuitry. The Breakpoint Comparator circuitry is checked first by loading a test procedure into Prototype memory which accesses workspace register 0 one time with a MOV R0,R1 instruction. The workspace pointer and either Breakpoint A or B depending on which is being test is set to one of a group of values (>8000, >7800, >4000, >2000, >1000, >800, >7C0, >400, >200, >100, >80, >40, >3E, >20, >10, >8, >4, >2, >0) and the halt on breakpoint function is enabled so that when the downloaded procedure is executed it should cause the Emulator to halt. If the Emulator halts too soon or if it does not halt at all, an error message is output. The above values will successively test each address bit of the compare registers. The reason for the values >7800, >7C0, and >3E is that they set all the bits high on each of the MSB-LSB comparator packages respectively thus guarenteeing that if any of the input lines to these packages are bad the user will be able to pinpoint more accurately which of the packages is at fault.

Next, all possible Breakpoint Mode Combinations are checked by executing a procedure out of Prototype memory which generates each of the possible cycles used by the Emulator to qualify the halt condition (memory read, memory write, and instruction acquisition). The test

checks not only to see that the Emulator will halt on a breakpoint but that it will also not halt if the "Halt On Breakpoint" feature is disabled. Both sets of Breakpoint Compare circuitry A and B are tested identically with the same Verb.

5.3.8 E8 Verb

E8 executes test Part 8. Part 8 loads and executes the standard AU04 Standalone CPU Test in Emulator memory starting at address >0100. If the test encounters any CPU errors, it will place an error code in lower Prototype memory depending on the type of error (error codes start at address >0000). After the test completes and IDLEs, the Diagnostic checks for any error codes returned and reports all commands that failed. At the end of the error list the AU04 test places an end-of-list pointer (>FFFF). If no errors occur this will be the first value in memory. If for any reason an incorrect error code is encountered the diagnostic reports an undefined error.

5.3.9 E9 Verb

E9 executes test Part 9. Part 9 tests the Trace Module/Emulator interface. The Trace Module must be connected to the Emulator via the Data and Control Cables during the execution of this test.

The test consists of loading and executing a single test

procedure from Prototype memory. The test procedure is designed such that it checks out both the data and all control lines between the Trace Module and Emulator. It was ,however, not possible to test all the control lines at the same time, therefore, the test was executed twice, each time setting up different trace configurations. The first time the procedure is executed it tests all data lines, control lines and qualifiers with the exception of D1-The IAQ data line qualifier, P4-48 of the Trace Module/Emulator data connector. The second execution of the procedure checks out D1. The data lines are tested by tracing a shifting pattern starting with >0001 and continuing through >8000 while the qualifiers and control lines are tested by causing the Trace Module to issue a HALT when it encounters certain conditions in the test procedure. The Emulator must then acknowledge the HALT and issue a HOLD which causes the Trace Module to stop. When the Trace Module is halted this should cause the Emulator to halt also.

5.3.10 TS Verb

TS executes test Part 10. Part 10 tests the Target System/Buffer interface using any of the available Buffers. The appropriate Target System must be connected in order to execute this test.

With the specially modified Target System the Diagnostic

is able to test the CRU data lines and all control lines. The CRU data lines are tested using the RC and WC (Read and Write CRU Field) Control Program commands and reading and writing patterns of 2, 4, >10, >100, and >8000. Next, the interrupt capability is tested by setting up all interrupt traps in Prototype memory and generating interrupts for levels 1, 2, 4, 8, and 15 (levels 1, 2, 3, and 4 for the TMS9980A/81). Each interrupt should return a code to workspace register 0 equal to its interrupt level. Thus, interrupt level 8 returns a code of 8 in register 0 (Workspace Pointer set to >100). Following this the Interrupt Request line (INTREQ) to the Emulator is tested by tracing a procedure executing out of Prototype memory which causes INTREQ to toggle. Next, RESET and LOAD are tested by setting up their interrupt vectors in Prototype memory and checking for a proper return code in Register 0 as in the previous interrupt test. Finally, Target System HOLD is tested with a test procedure executing out of Prototype memory. The test procedure sets and checks the HOLD line. If HOLD is operating properly, a code of >FEED is returned to Register 0. If an error occurs the procedure returns an error code of >0001 for HOLD stuck high or >0002 for HOLD stuck low. This concludes the Target System/Buffer interface test.

5.3.11 EM Verb

EM executes the Expansion Memory Test. The verb consists of three basic parts. The first part tests the 256x4 Bit Memory Mapping/Write Protect RAM by writing and reading the patterns >0000, >0100, >0200, >0400, >0800, >0500, >0A00, and >0F00. The patterns are written beginning at one of two different addresses depending on which Buffer Board is connected: addresses >1000->11FE for the TMS9980A/81 Buffer Board and addresses >2000->21FE for the SBP9900A and TMS9900-1 Buffer Boards. This part finishes leaving the RAM with F's written in all locations which disables the Write Protect feature for all of Expansion Memory and maps all addressing to Expansion Memory.

The next part of the verb tests all of the Expansion Memory by using the existing Address and Speed tests from the E2 verb. Any errors that occur during this portion of the test will be reported exactly as they would while executing the E2 verb. All the tests from the E2 verb are used in this portion of the EM verb with the exception of the Emulator Control Memory Pattern, Address and Speed tests.

The final part of the EM verb tests the Write Protect Violation Register and Flag. First, certain memory locations are cleared one at a time, each time making

sure that the Write Protect Violation Flag does not go active (it should not since memory is un-protected at this point). All memory is then write protected and mapped to the Expansion Memory Board. The Write Protect Violation Flag is checked to make sure it is not active (high) and a selected memory location (>8000, >4000, >2000, ..., >0000) is written with a >FFFF and the Write Protect Violation Flag is checked for active condition. Next, the Write Protect Violation Register is read at address >2200 in Control Memory (address >1200 for the TMS9980A/81) to see that the proper address was stored. The Write Protect Violation Flag is then checked to make sure that it was cleared after reading the Violation Register, and finally Expansion Memory is read at the selected location to make sure that the data remained unchanged.

5.3.12 EW Verb

EW executes the Write Protect Violation Register and Flag Test. The test clears all of the 256 word control RAM, and then writes a value of >FFFF to a single location in memory and reads all the rest of memory, making sure that the value was not written anywhere else. The test continues clearing memory and writing >FFFF for every location in memory. Next, a pattern test is run on the control RAM as described in Section 5.3.11 above. This

test is a more detailed check of the control RAM than under the EM verb and takes about 5 minutes to complete.

5.3.13 EA Verb

EA executes Parts 1, 2, 3, 4, 5, 6, 7 and 8 in that order.

5.3.14 LED Test

A test procedure can be used to check out operation of the Emulator green LED lamp. To test the LED the diagnostic must not be in the GC (Generate Command) mode. The user simply enters a IE command which should disable all tracing and breakpoints (note that the IE command will not modify breakpoint operation while in the GC mode), followed by a RE (Run Emulation) command which should cause the LED to light and remain lit. To turn off the LED the user then enters another IE command. This concludes the LED test.

5.4 HT Verb

The HT (HALT) Verb causes the Emulator to enter the Control Mode at the top of the control program from the Execute Mode, at the end of the currently executing instruction. This verb should always be able to bring the Emulator back to the control mode unless it is in a HOLD condition.

5.5 RS Verb

The RS (RESTART) Verb causes the Emulator to exit the

Execute Mode immediately and enter the Control Mode at the top of the control program. The difference between this verb and the HT verb is that the Emulator will not complete any currently executing instruction. This verb will override a HOLD condition.

5.6 SC Verb

The SC (SELECT CLOCK) Verb selects which clock will be used by the Emulator. The user may select either the target clock, the internal clock or may turn both clocks off. A sample of SC follows:

```
VERB? -SC
SELECT CLOCK  (TAR=0,INT=1,OFF=3)-  1
EMULATOR CLOCK ON
   NOTE:  "IE" COMMAND MUST FOLLOW

VERB? -
```

The special note always follows a SC command. Also, a RS command is always executed internally after any clock select and if the Emulator fails to RESTART an error message will also follow.

5.7 GC Verb

The GC (GENERATE COMMAND) Verb is a special function verb that allows the user to generate a list or table of verbs which can be executed together rather than one at a time. Normally when a verb is entered and after the last of the required parameters is supplied, the verb is executed immediately by the diagnostic. The GC verb, however, sets a flag which tells the diagnostic not to

execute the verb but to place the command opcode and all associated parameters into special command and data buffer areas in host memory. Each new command entered is placed after the previous one in the buffer space with an end of list pointer (>FFFF) inserted following the last parameter. Each verb placed in the buffer space like this takes up 4 words of memory. The first two words will always be the opcode and second argument for the particular command. The next two words will vary in meaning depending on the particular verb and will sometimes contain associated parameters or a pointer to the data buffer space where the remaining parameters can be found. After the user enters the GC verb he may then enter any of the available list of EMULATOR COMMAND VERBS with the exception of a few special verbs: HT, RS, CS and DS. Note: These verbs may be executed at this time but will not be included in the list being built since they only control CRU bits or set special flags. After entering a verb following the GC verb and providing the required information for that particular verb an asterisk (*) appears. This indicates that the user is presently in the GC mode and is building up the list of verbs. The user may continue to enter more verbs and thus build up a list of a maximum of 16 verbs, or the list can then be executed using the XT or the LT verbs described later.

Once the list of verbs has been executed the user may continue to execute that same list or he may execute new verbs separately. Once, however, a new verb is executed individually the list is destroyed and if the user wishes to re-execute the list he must re-enter it in its entirity. Once the user gives the command to execute the verb list, the diagnostic begins executing at the top of the list and continues until it finds an end of list pointer or an invalid command opcode. When the execution of the list is complete, the diagnostic will print out a header that tells how many verbs were executed. An example of the GC verb follows:

```
VERB?  -GC

VERB?  -IE     *

VERB?  -PC
SELECT MEMORY      (PROTO=0,CONTRL=1)    0     *

VERB?  -RM
STARTING MEMORY ADDRESS FOR READ    -246     *

VERB?  -WM
STARTING ADDRESS FOR WRITE    -10FD
DATA=1000
DATA=C080
DATA=6ABC
DATA=1000
DATA=1000
DATA=340
DATA=E
FINISHED?    (1=Y,0=N)    -1     *

VERB?  -PC
SELECT MEMORY      (PROTO=0,CONTRL=1)    1     *

VERB?  -RM
```

```
STARTING MEMORY ADDRESS FOR READ   -828    *

VERB?  -

VERB?  -
```

5.8 XT Verb

The XT (EXECUTE TABLE) Verb causes the diagnostic to execute the list of verbs previously built up after entering the GC Verb. The list of verbs is executed once only, beginning at the top of the list. After the execution a header message tells the user how many verbs were executed. Execution will stop only when the diagnostic encounters an end of list pointer or an undefined command.

5.9 LT Verb

The LT (LOOP ON TABLE) Verb loops on the XT Verb.

5.10 SL Verb

The SL (SHOW LIST) Verb is used in conjunction with the GC, XT, and LT verbs. The SL Verb shows all the verbs in the verb list including the name of the verb and its four associated data words from the buffer. The SL Verb continues displaying verbs from the list until it encounters an end of list pointer or an undefined verb. The data value preceding the two letter abbreviation is the actual location in host memory where the four data words begin. Depending on the verb, a further descriptor may follow the two letter abbreviation. An example of the SM Verb follows:

```
VERB? -SL
5FF8    "IE"                    0080    0000    0000    0000
6000    "RM"    (PROTO)         0480    0100    6AC0    0002
6008    "WM"    (PROTO)         0400    0400    6AC4    0004
6010    "RM"    (CNTRL)         0A80    0050    6ACC    0001
6018    "DB"                    0700    0000    6ACE    0003
END OF LIST

VERB? -
```

5.11 IE Verb

The IE (INITIALIZE EMULATOR) Verb selects emulator user memory, disables breakpoints (Breakpoint A and B set to >FFFF, Halt on Breakpoint disabled) and sets up the trace configuration to trace all MA (memory accesses) anywhere. It also returns the emulator buffer type (>0000=SBP9900A, >0001=TMS9900-1, >0002=TMS9980A/81). An example of the IE Verb follows:

```
VERB? -IE
EMU BUFFER TYPE=0000

VERB? -
```

5.12 SM Verb

The SM (SELECT MEMORY) Verb selects which of the two types of prototype memory (EMU 8KB or TARGET) will be used in succeeding read and write operations. It also reads the current selected memory configuration. This verb does not select between prototype and control memory, however, it will read the entire memory configuration. To select between PROTOTYPE and CONTROL MEMORY use the PC Verb as described in Section 5.25. An example of the SM Verb follows:

```
VERB? -SM
READ OR WRITE?  (R=1,W=0)  -0
SELECT EMU PROTOTYPE OR TARGET MEMORY  (E=1,T=0)  -1

VERB? -SM
READ OR WRITE?  (R=1,W=0)  -1
MEM. SELECTED=0 (1=CONTRL,0=PROTO), PROTO. MEM.=1 (0=TARG,1=EMU)

VERB? -
```

Note, that in the second example if the GC Verb had just been used prior to the SM Verb, the line of information following the READ OR WRITE? statement would not have been output.

5.13 CK Verb

The CK (READ CLOCK SELECTION) Verb tells which clock has been selected by the CS Verb. An example of the CK verb follows:

```
VERB? -CK
CLOCK= 0001      1=EMU  0=TARGET

VERB?-
```

5.14 RR Verb

The RR (READ PROTOTYPE CPU REGISTERS) Verb displays the Emulator's workspace pointer, program counter and status register in that order of the Emulator. An example of the RR verb follows:

```
VERB? -RR
WP=0000   PC=0200   ST=C0FF

VERB? -
```

5.15 WR Verb

The WR (WRITE PROTOTYPE REGISTERS) Verb loads the three

internal CPU registers with user specified values. The verb asks for a code indicating which register is to be written and then the value to be written. An example of this verb follows:

```
VERB? -WR
MODIFY REGISTER?  (WP=0,PC=2,ST=4)-  2
DATA=0600

VERB? -
```

5.16 RM Verb

The RM (READ MEMORY) Verb displays a block of 64 words from either control or prototype memory depending on which has been selected by the PC and SM verbs described in Sections 5.25 and 5.12 respectively. The verb asks for a starting address. The last 4 bits of the address supplied by the user are truncated. An example of the RM verb follows:

```
VERB? -RM
STARTING MEMORY ADDRESS FOR READ  -246
ADDR     0     2     4     6     8     A     C     E
0240    10FF  10FF  10FF  10FF  10FF  10FF  10FF  10FF
0250    10FF  10FF  10FF  10FF  10FF  10FF  10FF  10FF
0260    10FF  10FF  10FF  10FF  10FF  10FF  10FF  10FF
0270    10FF  10FF  10FF  10FF  10FF  10FF  10FF  10FF
0280    10FF  10FF  10FF  10FF  10FF  10FF  10FF  10FF
0290    10FF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF
02A0    FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF
02B0    FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF  FFFF

VERB? -
```

5.17 WM Verb

The WM (WRITE MEMORY) Verb writes blocks of up to 64 data words to either control or prototype memory depending on

which has been previously selected by the PC and SM Verbs described in Sections 5.25 and 5.12 respectively. The verb asks for a starting address where the data will be output to and the diagnostic truncates the address to an even byte address. The verb also asks for data and the user supplies the data in hex form. The user may continue to input data up to a maximum of 64 words. When the maximum number of input words is reached a message "DATA TABLE FULL" is printed and the verb is automatically exited and the 64 word block of data is written to memory. The user may then re-enter the verb and continue to write more data to memory. If at any time the user wishes to exit the verb and write the data to memory, he must enter the value "E" after the prompt "DATA=". The diagnostic then asks the user if he is finished and if so, all the data except the "E" is output to memory; if not, the "E" is entered as data. Note that not until the user exits the verb with the "E" response will the data be written to memory. If the user wishes to exit at any time during the execution of this verb by entering the command key or an "@" sign the operation will be aborted and memory will not be altered. An example of the WM verb follows:

```
VERB? -WM
STARTING ADDRESS FOR WRITE  -10FD
DATA=1000
```

```
DATA=C080
DATA=6ABC
DATA=1000
DATA=1000
DATA=340
DATA=E
FINISHED?  (1=Y,0=N)  -1

VERB? -
```

5.18 RC Verb

The RC (READ TARGET SYSTEM CRU FIELD) Verb reads the 16-Bit Target System CRU field. The user must specify the starting base address of the field and the field width. The field width values may be 0-15, 0 indicating a width of 16. An example of the RC verb follows:

```
VERB? -RC
CRU BASE ADDRESS=140
CRU FIELD WIDTH=F
TARGET SYSTEM CRU FIELD=7FFF

VERB? -
```

5.19 WC Verb

The WC (WRITE TARGET SYSTEM CRU FIELD) writes as many as 16 bits to the target system CRU Field. The user must specify the starting base address of the field and the field width to be written. An example of the WC verb follows:

```
VERB? -WC
CRU BASE ADDRESS=140
CRU FIELD WIDTH=0
DATA=0000

VERB? -
```

5.20 DT Verb

The DT (DEFINE TRACE CONFIGURATION) Verb defines the

trace configuration for the emulator. The user may either read the trace configuration or may define the trace configuration. The user must supply the Trace Count, a Trace Mode which tells the Emulator what type of information to trace, and both trace address boundaries. If a count of 0 is specified for the Trace Count, the Emulator will load the trace counter with >FF and will not halt when the trace counter decrements through zero If a non-zero value is specified, the Emulator will load the trace counter with that value and when the trace counter decrements to zero it will halt. The Emulator will trace according to the mode specified, whatever address values are less than or equal to the Upper Bound, AND whatever address values are greater than or equal to the Lower Bound. If the Upper and Lower Bounds are equal, the Emulator will trace only that specific address within the limits of the Trace Mode. An example of the DT verb follows:

```
VERB? -DT
READ OR WRITE?  (R=1,W=0)  -0
TRACE COUNT=FF
ENTER TRACE MODE (>F0=TRACE ALL,>B0=EXECUTE MODE,>B4=PC ONLY)  F0
LOW TRACE BOUND ADDRESS=0
UPPER TRACE BOUND ADDRESS=FFFE

VERB? -DT
READ OR WRITE?  (R=1,W=0)  -1
COUNT/MODE=FFF0     LOW ADDR=0000     HIGH ADDR=FFFE

VERB? -
```

5.21 DB Verb

The DB (DEFINE BREAKPOINT CONFIGURATION) Verb defines the breakpoint configuration for the Emulator. The breakpoint configuration may either be read or defined. To define the breakpoint configuration the user must specify the Breakpoint Mode, whether or not to halt once a breakpoint occurs, and both breakpoint values A and B. An example of the DB verb follows:

```
VERB? -DB
READ OR WRITE?  (R=1,W=0)  -0
BREAKPOINT ON:  (>B4=IAQ,>D0=WRITE,>D2=READ,>F0=ANY MA)  F0
ENABLE BREAKPOINTS?  (1=YES,0=NO)  -1
BRKPT A=100
BRKPT B=502

VERB? -DB
READ OR WRITE?  (R=1,W=0)  -1
BREAKPT/MODE= 2F0  BRKPT A= 0100  BRKPT B= 0502

VERB? -
```

5.22 RT Verb

The RT (READ TRACE SAMPLES) reads a specified number of trace samples from the Emulator's 512 byte trace RAM and displays the samples beginning with the latest sample traced. The user may look at all or part of the samples traced, by specifying a Display Count. The diagnostic will print out either the number of samples traced or the number of samples requested by the user, whichever value is smallest. The diagnostic will never display more samples than have been traced. In other words, if >20 samples have been traced by the Emulator and the user

places a value of >FF in the Display Count, the diagnostic will display only >20 samples. An example of the RT Verb follows:

```
VERB? -RT
DISPLAY COUNT=6
MEMR   CTRL  ADDR  CTRL  ADDR  CTRL  ADDR  CTRL  ADDR
FF     0600  0108  0600  0106  0600  0104  0600  0102
FB     0600  0100  0600  00FE
00FF SAMPLES TRACED    0006 SAMPLES DISPLAYED

VERB? -
```

The DATA is the actual trace of the address bus while the CTRL represents the three added bits of control information (DBIN, IAQ, and INTREQ).

5.23 RE Verb

The RE (RUN EMULATION) Verb executes the program resident in Prototype Memory (either Emulator 8KB Memory or Target Memory) starting at the address contained in the Program Counter register, with Workspace and Status as set up by the WR verb as described in Section 5.15. An example of the RE verb follows:

```
VERB? -RE

VERB? -
```

5.24 CP Verb

The CP (RUN PROGRAM IN CONTROL RAM) Verb executes the program in Control RAM beginning at address >828. The Workspace Pointer is the same as the Control Program's. The Test code executed in Control Ram must return to Control Program control by executing a RT (B *11)

instruction. An example of the CP verb follows:

```
VERB? -CP

VERB? -
```

5.25 PC Verb

The PC (SELECT PROTOTYPE OR CONTROL MEMORY) Verb sets a special flag in the diagnostic which tell the Emulator to select between Prototype or Control Ram memory. This verb does not, however, select between Emulator 8KB and Target Memory. After this verb is executed, all memory operations such as the use of RM and WM verbs will be directed toward the memory selected. If the user selects Prototype memory he must also decide which of the two types of Prototype memory will be used by the use of the SM verb as described in Section 5.12. An example of the PC verb follows:

```
VERB? -PC
SELECT MEMORY   (PROTO=0,CONTRL=1)  -1

VERB? -
```

5.26 PE Verb

The PE (PRINT ERROR COUNT) Verb prints the error counts of both the last EXECUTE verb test and the total error count of all previously executed verbs. An example of the PE verb follows:

```
VERB? -PE
PREVIOUS ERROR COUNT=0008,  TOTAL ERROR COUNT=00CE

VERB? -
```

5.27 DS Verb

The DS (DUMP EMULATOR STATUS) Verb dumps the status of the Emulator's 24 bit Host CRU field, including the 16 bit data field. An example of the DS verb follows:

```
VERB? -DS
DATA   HA- ER  CC  NU  NU  ID- HD- IP
4869    1   1   0   0   0   1   1   1

VERB? -
```

The DATA is the 16-Bit data CRU field. The remaining 8 bits displayed have the following meaning:

HA- Halt Acknowledge-

ER Emulator Ready (EMREDY)

CC Command complete (COMCMP)

NU Not Used

NU Not Used

ID- IDLE-

HD- HOLD-

IP Interrupt Pending (INTPEN)

5.28 Debug Verbs

In addition to the Test Verbs, the following Debug Verbs are included in the 9900/9980 Emulator Diagnostic.

5.28.1 LB Verb

The LB (LOAD HOST CRU BASE) Verb sets up the Host CRU base for all the debug verbs. An example of the LB verb follows:

```
VERB? -LB
```

```
ENTER CRU BASE ADDRESS  DEFAULT=XXXX-XXXX

VERB? -
```

5.28.2 LD Verb

The LD (LOAD HOST CRU) Verb allows the operator to output a 16-bit pattern to the Host CRU. An example of the LD verb follows:

```
VERB? -LD
ENTER BIT PATTERN  DEFAULT=XXXX-XXXX

VERB? -
```

5.28.3 ST Verb

The ST (STORE HOST CRU) Verb stores and displays 16 bits of Host CRU data. An example of the ST verb follows:

```
VERB? -ST
BIT PATTERN=4869

VERB?
```

5.28.4 SO Verb

The SO (SET CRU BIT TO ONE) Verb allows the user to set a Host CRU bit to logical one. The bit number is entered in decimal (0-99). An example of the SO verb follows:

```
VERB? -SO
ENTER BIT NUMBER DEFAULT=XX-XX

VERB?-
```

5.28.5 SZ Verb

The SZ (SET HOST CRU BIT TO ZERO) Verb allows the user to set a Host CRU bit to logical zero. The bit number is entered in decimal (0-99). An example of the SZ verb follows:

```
VERB? -SZ
ENTER BIT NUMBER DEFAULT=XX-XX

VERB? -
```

5.28.6 BT Verb

The BT (TEST HOST CRU BIT) Verb reads and displays one Host CRU bit. The bit number is entered in decimal (0-99). An example of the BT verb follows:

```
VERB? -BT
ENTER BIT NUMBER DEFAULT=XX-XX
BIT=X

VERB? -
```

5.28.7 BS Verb

The BS (BIT SAMPLE) Verb samples and displays a Host CRU bit once every second. The bit number is entered in decimal (0-99). An example of the bs verb follows:

```
VERB? -BS
ENTER BIT NUMBER DEFAULT=XX-XX
BIT=XXXX

VERB? -
```

5.28.8 HD Verb

The HD (CONVERT HEX TO DECIMAL) Verb converts a 4-digit hex number to decimal. An example of the HD verb follows:

```
VERB? -HD
ENTER HEX NUMBER XXXX     DECIMAL =XXXXXX

VERB?-
```

5.28.9 DH Verb

The DH (CONVERT DECIMAL TO HEX) Verb converts a 5-digit

decimal number to hex (Max=65535). An example of the DH verb follows:

```
VERB? -DH
ENTER DECIMAL NUMBER-XX-XX-XX=XXXX

VERB? -
```

5.28.10 BP Verb

The BP (SET BREAKPOINT) Verb allows the user to save a memory location and replace it with a BLWP instruction. This causes the program under test to return to DOCS control and dump the Workspace Pointer, Program Counter, and Status Register. The verb also dumps all current workspace register values. An example of the BP verb follows:

```
VERB? -BP
ENTER BREAK PT. ADDR.   DEFAULT=XXXX-XXXX

VERB? -
```

When the program encounters the breakpoint address the following information will be output:

```
BREAK ADDR=>XXXX,        INSTR=XXXX, WP=XXXX, PC=XXXX, ST=XXXX
R00=XXXX R01=XXXX R02=XXXX R03=XXXX R04=XXXX R05=XXXX R06=XXXX ...
R08=XXXX R09=XXXX R10=XXXX R11=XXXX R12=XXXX R13=XXXX R14=XXXX ...
```

5.28.11 CB VERB

THE CB (CLEAR BREAKPOINT) VERB RESTORES THE MEMORY LOCATION THAT WAS ALTERED BY THE BP VERB. AN EXAMPLE OF THE CB VERB FOLLOWS:

```
VERB? -CB

VERB? -
```

5.28.12 BL VERB

THE BL (LOOP ON SET BIT) VERB ALTERNATELY EXECUTES SET BIT TO ONE AND SET BIT TO ZERO INSTRUCTIONS. THIS GENERATES A SQUARE WAVE WITH A 50% DUTY CYCLE. AN EXAMPLE OF THE BL VERB FOLLOWS:

```
VERB? -BL
ENTER BIT NUMBER DEFAULT=XX-XX

VERB? -
```

5.28.13 LO VERB

THE LO (LOOP ON ONE) VERB REPEATIVELY SETS A CRU BIT TO A LOGICAL ONE. THE BIT NUMBER IS ENTERED IN DECIMAL (0-99). AN EXAMPLE OF THE LO VERB FOLLOWS:

```
VERB -LO
ENTER BIT NUMBER DEFAULT=XX-XX

VERB? -
```

5.28.14 LZ VERB

THE LZ (LOOP ON ZERO) VERB REPEATIVELY SETS A CRU BIT TO A LOGICAL ZERO. THE BIT NUMBER IS ENTERED IN DECIMAL (0-99). AN EXAMPLE OF THE LZ VERB FOLLOWS:

```
VERB? -LZ
ENTER BIT NUMBER DEFAULT=XX-XX

VERB? -
```

6.0 ERROR MESSAGES

WHEN AN ERROR OCCURS THE ERROR MESSAGE NUMBER FROM THE FOLLOWING LIST IS DISPLAYED ON THE FRONT PANEL AND THE FAULT LIGHT IS LIT.

#01 UNEXPECTED EMULATOR INTERRUPT

#02 HLTACK LINE FAILED TO RESPOND AFTER HALT WAS ISSUED

#03 EMREDY LINE FAILED TO RESPOND AFTER HALT WAS ISSUED

#04 COMMUNICA. ERR: HLTACK FAILED TO GO HIGH FOR "RE"

#05 COMMUNICA. ERR: EMREDY LINE LOW, EXPECTED HIGH

#06 COMMUNICA. ERR: EMREDY LINE FAILED TO GO LOW

#07 COMMUNICA. ERR: EMREDY NOT HIGH AFTER HOST READY SET LOW

#08 INVALID COMMAND

#09 INDEX OUT OF BOUNDS

#0A SEMANTIC ERROR

#0B WARNING: COMMND BIT HIGH, ERROR MESSAGE MUST FOLLOW

#0C BAD OPCODE AT COMMAND #XXXX

#0D EMU FAILED TO RESTART

#0E BAD OPCODE=XXXX

#0F FIRMWARE FAILED TO START AT TOP AFTER HALT

#10 CRU INTERFACE ERR: COMCMP BIT STUCK HIGH

#11 CRU INTERFACE ERR: COMCMP BIT STUCK LOW

#12 CRU INTERFACE ERR: IDLE BIT STUCK LOW

#13 CRU INTERFACE ERR: EXPTD CRU DATA=XXXX, ACTL DATA=XXXX

#14 INTPEN LINE STUCK HIGH, INTERRUPT TEST ABORTED

#15 INTERRUPT TEST ERR: EMREDY NEVER WENT LOW TO CAUSE
INTERRUPT

#16 INTPEN FAILED TO SHOW INTERRUPT W/EMREDY;
CRU INTRPT MASKED

#17 INTPEN FAILED TO CAUSE CRU INTERRUPT W/EMREDY

#18 HLTACK WAS NOT LOW IN CONTROL MODE

#19 HLTACK DOES NOT INDICATE EMULATION RUNNING

#1A EMULATOR FAILED TO RE-ENTER CONTROL MODE AFTER
EMULATION

#1B HLTACK FAILED TO CREATE INTERRUPT

#1C PATRN ERR: @ADDR=XXXX, EXPTD DATA=XXXX, ACTL
DATA=XXXX

#1D ADDRESS TEST ERR: ADDR & DATA NOT EQUAL, ADDR=
XXXX, DATA=XXXX

#1E @SPEED EMULATION FAILED

#1F @SPEED MEM ERR: @ADDR=XXXX, EXPTD DATA=XXXX,
ACTL DATA=XXXX

#20 @SPEED CONTROL TEST FAILED TO COMPLETE

#21 COMMND BIT FAILED, TEST ABORTED

#22 BITS REG. ERR: IDLE BIT STUCK LOW

#23 BITS REG. ERR: IDLE BIT STUCK HIGH

#24 BITS REG. FAILED TO CLEAR, ACTL DATA=XXXX

#25 BITS REG. ERR: BIT #XX FAILED "DB" COMMAND

#26 BITS REG. ERR: MAP BIT FAILED

#27 BITS REG. ERR: EMU 4K SELCT BIT FAILED

#28 BITS REG. ERR: BIT #XX FAILED "DT" COMMAND, EXPTD
DATA=XXXX, ACTL DATA=XXXX

#29 PATRN ERR: @ADDR=XXXX, EXPTD PATRN=XXXX, ACTL
PATRN=XXXX

#2A RIPPLE ERR: @ADDR=XXXX, TEST DATA=XXXX, ACTL DATA=
XXXX

#2B LOWER BOUND REG. FAILED TO CLEAR, ACTL DATA=XXXX

#2C UPPER BOUND REG. FAILED TO CLEAR, ACTL DATA=XXXX

#2D UP BND REG FAILED "DT" COMMAND, EXPTD DATA=XXXX
ACTL DATA=XXXX

#2E LO BND REG FAILED "DT" COMMAND, EXPTD DATA=XXXX
ACTL DATA=XXXX

#2F BREAKPT A REG FAILED TO CLEAR, ACTL DATA=XXXX

#30 BREAKPT B REG FAILED TO CLEAR, ACTL DATA=XXXX

#31 BP A FAILED "DB" COMMAND, EXPTD DATA=XXXX, ACTL
DATA=XXXX

#32 BP B FAILED "DB" COMMAND, EXPTD DATA=XXXX, ACTL
DATA=XXXX

#33 EMULATOR FAILED TO HALT ON TRACE MEMORY FULL

#34 EMULATOR FAILED TO INDICATE IDLE EXECUTION

#35 TRACE COUNTER ERR: EXPTD COUNT=XX, ACTL COUNT=XX

#36 UB COMPAR FAILED ON XXXX, EXPTD TRC CNT=XX, ACTL CNT=XX,
UB=XXXX, LB=XXXX

```
#37 LB COMPAR FAILED ON XXXX, EXPTD TRC CNT=XX, ACTL CNT=XX,
        UB=XXXX, LB=XXXX
#38 COMPAR ADDR ERR: FAILED COMPARING XXXX, EXPTD TRC CNT=XX,
        ACTL CNT=XX
#39 TRC MEM ADDR ERR: @ADDR=>XX, EXPTD DATA=>XXXX, ACTL
        DATA=>XXXX
#3A TRC MEM PATRN ERR: @ADDR=XX, EXP PATRN=>XXXX, ACTL
        PATRN=>XXXX
#3B DIAG RAM ERR: @ADDR=XX, EXPTD DATA=>X, ACTL DATA=>X
#3C PC MODE ERR: WRONG ADDR TRACED, EXPTD ADDR=XXXX,
        ACTL ADDR=XXXX
#3D EA MODE ERR: WRONG ADDR TRACED, EXPTD ADDR=XXXX,
        ACTL ADDR=XXXX
#3E PC MODE ERR: BAD TRACE CNT, EXPTD CNT=>XX, ACTL CNT=>XX
#3F EA MODE ERR: BAD TRACE CNT, EXPTD CNT=>XX, ACTL CNT=>XX
#40 IAQ/DBIN TRACE SPEED ERR: @ADDR=XX, EXPTD PATRN=X, ACTL
        PATRN=X
#41 EMULATOR HALTED, SHOULD NOT HAVE
#42 EMULATION FAILED COMPLETION
#43 COMPAR REG FAILED COMPARING ADDR >XXXX
#44 EMULATOR FAILED TO HALT ON BREAKPT IN READ MODE
#45 EMULATOR FAILED TO HALT ON BREAKPT IN WRITE MODE
#46 EMULATOR FAILED TO HALT ON BREAKPT IN IAQ MODE
#47 EMULATOR HALTED TOO SOON
#48 EMULATOR HALTED AT WRONG PLACE IN IAQ MODE
```

#49 EMULATOR HALTED AT WRONG PLACE IN WRITE MODE

#4A EMULATOR HALTED AT WRONG PLACE IN READ MODE

#4B CPU TEST FAILED TO COMPLETE

#4C FOLLOWING CPU TEST ERROR NOT DEFINED

#4D FAILURE EXECUTING XXXX

#4E EMULATOR & TRACE MODULE FAILED TO HALT ON BREAKPT

#4F EMULATOR FAILED TO HALT ON BREAKPT;TRACE MODULE HALTED

#50 EMULATOR HALTED;TRACE MODULE FAILED TO HALT ON BREAKPT

#51 BAD TM DATA: EXPTD D0-D3/DATA=XX/ XXXX, ACTL D0-D3/DATA=
XX/ XXXX

#52 BAD EMU ADDR TRACED: EXPTD ADDR=XXXX, ACTL ADDR=XXXX

#53 TARGET CRU ERR: EXPTD DATA=XXXX, ACTL DATA=XXXX

#54 INTERRUPT # XX FAILED, ACTL INTERRUPT=XX

#55 9980 GENERATED INTERRUPT FOR NOP, EXPTD PC=0202, ACTL
PC=XXXX

#56 TARGET RESET FAILED

#57 INTREQ FAILED TO TOGGLE: EXPTD TRC=X, ACTL TRC=X (TRC
BITS=INTREQ/IAQ/DBIN/X)

#58 TEST PROGRAM FAILED TO EXECUTE

#59 TARGET HOLD STUCK HIGH

#5A TARGET HOLD STUCK LOW

#5B DATA WRITTEN TO ADDR >XXXX, APPEARED AT ADDR >XXXX,
EXPTD DATA=X, ACTL DATA=X

#5C PATRN ERR: ADDR=XXXX, EXPTD PAT=X, ACTL PAT=X

#5D WRITE PROTECT VIOLATION FLAG STUCK HIGH, SHOULD BE LOW

#5E WRITE PROTECT VIOLATION FLAG FAILED TO CLEAR @ADDR=XXXX

#5F WRITE PROTECT VIOLATION FLAG LOW, SHOULD BE HIGH

#60 WRITE PROTECT VIOLATION REG. ERR: EXPTD ADDR=XXXX

ACTL=XXXX

#61 DATA WRITTEN TO PROTECTED AREA: @ADDR=XXXX, EXPTD

DATA=0000, ACTL=XXXX

#62 WP VIOLATION FLAG ACTIVE WRITING TO UNPROTECTED MEM.

@ADDR=XXXX

7.0 PART NUMBERS

emu900, program description	-9001	(PD)
	-2001	(SRC,PD)
Fiche Kit	2250229-0009	(SP)
EMU900, Standard Link	2250229-1006	(FLO)
	-9006	(LML)
	-7006	(LC)
UNIEMU, Test Module 1	2250229-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
EMUMSG, Message Module 1	2250265-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
UNIACT, Test Module 2	2250212-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
ACTMSG, Message Module 2	2250266-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
UNIINT, Test Module 3	2250213-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
INTMSG, Message Module 3	2250267-1003	(OBJ)
	-9003	(LIST)

	-2003	(SRC)
UNIMEM, Test Module 4	2250214-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
MEMMSG, Message Module 4	2250268-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
UNISPD, Test Module 5	2250215-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
SPDMSG, Message Module 5	2250269-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
UNIBIT, Test Module 6	2250216-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
BITMSG, Message Module 6	2250270-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
UNIREG, Test Module 7	2250217-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
REGMSG, Message Module 7	2250271-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)

UNILUB, Test Module 8	2250218-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
LUBMSG, Message Module 8	2250272-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
UNIBPR, Test Module 9	2250219-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
BPRMSG, Message Module 9	2250273-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
UNITRC, Test Module 10	2250220-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
TRCMSG, Message Module 10	2250274-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
UNIBPC, Test Module 11	2250221-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
BPCMSG, Message Module 11	2250275-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
UNICPU, Test Module 12	2250222-1003	(OBJ)

	-9003	(LIST)
	-2003	(SRC)
CPUMSG, Message Module 12	2250276-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
DMSTRT, Test Module 13	2250223-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
AU04A, Test Module 14	2250224-3001	(OBJ/DATA)
	-9001	(LIST)
DMEND, Test Module 15	2250225-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
UNIETI, Test Module 16	2250226-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
ETIMSG, Message Module 16	2250277-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
UNITAR, Test Module 17	2250227-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
TARMSG, Message Module 18	2250278-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)

UNIEXP, Test Module 18	2250236-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
EXPMSG, Message Module 18	2250279-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
DBGVRB, Test Module 19	2250230-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
DBGMSG, Message Module 19	2250280-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
UNIEND, Test Module 20	2250231-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)

A

APPLICATION	
NEXT ASSY	USED ON
	7506

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

REV
SHEET

REV STATUS OF SHEETS
REV
SHEET

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± 010
2 PLACE DECIMAL ± 02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL STD 100

DWN DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group *Dallas, Texas*

PD, EMU940, 9940 EMULATOR TEST-990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	2250228-9901

SCALE REV. C SHEET 1 of 35

TI-0023-C

TABLE OF CONTENTS

5.2.10	TS VERB
5.2.11	IT VERB
5.2.12	IE VERB
5.2.13	MS VERB
5.2.14	CK VERB
5.2.15	RR VERB
5.2.16	WR VERB
5.2.17	RM VERB
5.2.18	WM VERB
5.2.19	RC VERB
5.2.20	WC VERB
5.2.21	DT VERB
5.2.22	DB VERB
5.2.23	RT VERB
5.2.24	RE VERB
5.2.25	CM VERB
5.2.26	GC VERB
5.2.27	XT VERB
5.2.28	LT VERB
5.2.29	SL VERB
5.2.30	HT VERB
5.2.31	SS VERB
5.2.32	LP VERB
5.2.33	DS VERB
5.2.34	SO VERB

1.0 SCOPE

The TMS 9940 Emulator Diagnostic tests a 9940 Emulator and Buffer combination. It will also test the interface between the Emulator and a Trace module. The Emulator and the Trace module may occupy any CRU slot. The Emulator may be connected to any unshared interrupt level. This diagnostic will operate in a 990/10 or 990/4 with 32KB of memory. The Burn-In version will operate in 16KB of memory.

2.0 REFERENCES

Listed below are documents useful for checkout of the TMS 9940 Emulator and Buffer when using this diagnostic.

PART NUMBER	TITLE
945400-9701	DIAGNOSTIC HANDBOOK
941567-9901	TMS 9940 EMULATOR SPEC
949913-9901	TRACE MODULE SPEC

3.0 HARDWARE CONFIGURATION

In addition to a 990/10 or 990/4 with 32KB of memory and an I/O device, the following equipment is needed:

REQUIRED

TMS9940 Emulator Board	941570-0001
TMS9940 Buffer Module Assy.	949971-0001

OPTIONAL

Trace Module	949910-0001

4.0 SOFTWARE DESCRIPTION

4.1 Linking Information

EMU940, Standard Link: 2250228-1006 (FLO)

Linkable Parts:
- EM9940
- EM942
- EM943
- EM944
- EM945
- EM946
- EM947
- EM948
- EM949
- EM9410
- MSG994

EMU940BI, Burn-in Link : 2250228-1009 (FLO)

Linkable Parts: EM9940

EM942

EM943

EM944

EM945

MSG994

4.2 Loading and Initialization

4.2.1 Loading

Loading instructions for available load media may be found in the 990 Diagnostic Handbook.

4.2.2 Initialization

When the Diagnostic has loaded, follow the instructions in the Diagnostic Handbook. The following is a list of the questions asked by the test during initialization:

ENTER TRACE MODULE CRU BASE ADDRESS. DEFAULT = 0100-

ENTER THE EMULATOR CRU BASE ADDRESS. DEFAULT = 0140-

ENTER THE TARGET SYSTEM CRU BASE ADDRESS. DEFAULT = 0020-

********** WARNING **********

EMULATOR AND IO DEVICE INTERRUPT LEVELS MUST NOT BE THE SAME

ENTER THE EMULATOR INT. LEVEL. DEFAULT = 0006--

The warning message is to remind the user that the Emulator cannot share the same interrupt level as a selected I/O device. The interrupt levels of the Emulator and the I/O device must be made different in order for the diagnostic to function properly.

5.0 VERBS AND VERB DESCRIPTIONS

5.1 Verb List

E1	EXECUTE PART 1 - ECHO AND INTERRUPT TESTS
E2	EXECUTE PART 2 - IE AND SM COMMAND TESTS
E3	EXECUTE PART 3 - MEMORY TESTS
E4	EXECUTE PART 4 - CPU REGISTER TESTS
E5	EXECUTE PART 5 - CRU TESTS
E6	EXECUTE PART 6 - TRACE AND BREAKPOINT TESTS
E7	EXECUTE PART 7 - DECREMENTER AND INTERRUPT TESTS
E8	EXECUTE PART 8 - 9940 CPU TESTS
E9	EXECUTE PART 9 - TRACE MODULE INTERFACE TESTS
EA	EXECUTE PARTS 1-7
L1	LOOP ON PART 1
L2	LOOP ON PART 2
L3	LOOP ON PART 3

L4	LOOP ON PART 4
L5	LOOP ON PART 5
L6	LOOP ON PART 6
L7	LOOP ON PART 7
L8	LOOP ON PART 8
L9	LOOP ON PART 9
LA	LOOP ON PARTS 1-7
TS	EXECUTE TARGET SYSTEM TEST
IT	INITIALIZE TESTS
GC	GENERATE COMMAND TABLE
XT	EXECUTE COMMAND TABLE
LT	LOOP ON COMMAND TABLE
SL	SHOW COMMAND TABLE
HT	HALT EMULATOR
SS	SINGLE STEP CLOCK
LP	LOOP CONTROL
DS	DISPLAY STATUS
SO	SET CRU BIT TO ONE
SZ	SET CRU BIT TO ZERO
SC	SELECT INTERNAL OR EXTERNAL CLOCK
SB	SET BREAKPOINT
RB	REMOVE BREAKPOINT
PE	PRINT ERROR COUNT
IE	INITIALIZE EMULATOR
SM	SELECT MEMORY

CK	READ CLOCK STATUS
RR	READ CPU REGISTER
WR	WRITE TO CPU REGISTER
RM	READ EMULATOR MEMORY
WM	WRITE TO EMULATOR MEMORY
RC	READ EMULATOR CRU
WC	WRITE TO EMULATOR CRU
DT	DEFINE TRACE
DB	DEFINE BREAKPOINT
RT	READ EMULATOR TRACE DATA
RE	RUN EMULATION
CM	CONTROL MEMORY TEST COMMAND

5.2 Verb Descriptions

5.2.1 E1 Verb

E1 executes test Part 1. Part 1 tests the basic Host to Emulator CRU interface and the Emulator interrupt.

To test the Host to Emulator interface, the Host first halts the Emulator. Then Host Ready is set high. When Host Ready is high, the Emulator will echo all data sent by the Host on the CRU data lines (bits 0-15). The Emulator first sends FFFF and checks the returned data. Then a Zero bit is shifted thru the Ones and checked.

The interrupt is checked when Host Ready is set high. An interrupt should have occurred when Host Ready was set high.

5.2.2 E2 Verb

E2 executes test Part 2. Part 2 tests the IE and SM Emulator commands.

The IE command initializes the Emulator by setting the memory bounds for 2KB of main memory and 128 bytes of ram. It also clears all Breakpoints and sets up to trace all of memory. The SM command can read or modify the memory configuration.

To test these commands an IE is executed first, then the SM command is executed to read the memory selection setup by IE. If the returned data is incorrect, one of the commands failed, but each must be checked by the user to determine which failed.

The modify function of SM and the trace and breakpoint functions of IE are checked in later tests.

5.2.3 E3 Verb

E3 executes test Part 3. Part 3 tests the Emulator's memory. Since the Emulator can contain as much as 8KB of memory, Part 3 determines the amount of memory on the board and prints a message telling the user how much memory is being tested. No control memory is tested. Part 3 consists of 5 tests.

The first test checks the memory address lines. The first test writes the address of each location to itself in the main memory and the workspace memory. In the main memory and workspace memory flag words, an increment by 3 data pattern is written. These data patterns are then read and checked. Next, address data is shifted in main memory to toggle the MSB'S. This checks bad address lines to a single chip. The amount of shift is determined by the size of memory.

The second test writes and reads 0000 to all memory locations.

The third test writes and reads 5555 to all memory locations.

The fourth test writes and reads AAAA to all memory locations.

The fifth test writes and reads FFFF to all memory locations.

5.2.4 E4 Verb

E4 executes Part 4. Part 4 tests the Emulator CPU Registers and the Read Register and Write Register commands. The test writes the registers select value to each register(WP=0 PC=2 ST=4). It then reads each register and checks the data.

5.2.5 E5 Verb

E5 executes test Part 5. Part 5 tests the Emulators 9940 CRU functions. It also tests the Write and Read CRU instructions. E5 consists of 9 parts.

The first part tests the CRU Flag Bits by shifting a 0 and a 1 thru the Flag Bits.

The second part tests Direction Bits 0-15. The test sets one direction bit at a time to point out (causing the data to loop back), writes 0000 to P0-P15, and checks the data. Then, only one Direction Bit at a time is set to point in (no loop back).

The third part tests Direction Bits 16-31 in the same manner part 2 test bits 0-15.

The fourth part tests P0-P15 by shifting a 0 and a 1 thru the bits. All Direction Bits point out.

The fifth part tests P16-P31 by shifting a 0 and a 1 thru the bits.

The sixth part tests the Decrementer and P/C- Bit by loading and reading shifting 0 and 1 data patterns. Dynamic testing of the decrementer is done in a later test.

The seventh part tests the MPSI Bits and CB1. The test bit and CB1 are set to cause loopback of MPSI data. Then 0 and 1 are shifted thru the bits.

The eighth part tests CB0, INT1, INT2, CB1, CB2, and CB3. The Test Bit and CB0-CB3 are set high. P0-P10 are then read and the data checked. These lines are now an external CRU interface due to CB0 being set high. 1E7 is the expected read data. Next, INT1 and INT2 are checked. The Test Bit allows P30 and P31 to modify INT1 and INT2 so they can be checked. P11-P16 are read checked next. 3F should be the read data due to CB1-CB3 being set high.

The nineth part tests the I/O bits to see if they were set to a logic high by the setting of the direction bits to the output mode. The test execises the lower 16 I/O bits and then the upper 16 bits.

5.2.6 E6 Verb

E6 executes test Part 6. Part 6 tests the Run, Trace and Breakpoint functions of the Emulator. E6 consists of 14 parts.

The first part tests the Run Emulation command and the Emulator Idle detection logic. The Emulator is setup to start execution on an IDLE instruction. The Run Emulation command is then executed. If IDLE is not detected, either IDLE or the Run Emulation command failed.

The second part tests MA Trace with two trace ranges. The Emulator traces address's 0100-0102 and 0108-010A only. A breakpoint occurs after 255 address's have been traced(trace buffer full).

The third part tests EX Trace and IAQ Breakpoint. Only the instruction at address 0102 is in the trace range, but all addresses accessed by the instruction should be traced. The BP should occur on the instruction at address 0104.

The fourth part tests MA Breakpoint and checks IAQ BP to make sure a BP does not occur on an address if it is not an IAQ.

The fifth part checks BP on MA Write by first addressing the

BP address during a read and then during a write. No BP should have occurred on the read access.

The sixth part checks BP on MA Read by addressing the BP address on a write cycle first.

The seventh part checks for an Illegal Address BP at a non existing memory location.

The eighth part checks for an Illegal Address BP at a legal address that is not selected by the MS command.

The ninth test checks to see that MA BP stops even if the MA is an IAQ.

The tenth test allows the Emulator to trace more than 255 samples. It should not breakpoint after 255 trace samples.

The eleventh test does a Trace Memory test. This test traces all 0's by tracing writes to address 0000.

The twelfth part tests Trace Memory by tracing FE00FF. The Test Bit is set to put address lines 8-15 into the Flag Bits of the Trace Memory and tracing address 00FF only. Bit 7 is set to zero by the control program, causing FE to be read from the flag bits.

The thirteenth test traces high address's only to put as many 1's as possible in the Trace Memory Bits that trace

Address Bits 0-7.

The fourteenth test insures that the emulator will breakpoint on an illegal op code. A program with an illegal op code is executed and the resulting program counter and trace data are checked.

5.2.7 E7 Verb

E7 executes Part 7. Part 7 tests the 9940 Interrupts and the Decrementer. E7 consists of 4 tests.

The first test causes a Level 3 Interrupt and checks to see that the right Interrupt Vector was taken.

The second test checks the Level 1 Interrupt in the same manner as test 1.

The third test puts the Decrementer in the Timer Mode and loads a count into the Decrementer. The test then determines if a decrementer interupt was caused. This test also checks to make sure that when in Idle and CB3=1, that the Decrementer Interrupt does not occurr.

The fourth test puts the Decrementer in the Count Mode. By outputing positive transitions on P17 the Decrementer is made to count down. Enough transistions are issued to count the Decrementer down to zero. Then, checks are made to

insure that a Decrementer Interrupt occured and that the Decrementer reloaded with the proper value.

5.2.8 E8 Verb

E8 executes Part 8. Part 8 of the diagnostic controls loading and executing of the standard 990/4 AU Diagnostic to test the Emulator's CPU functions. The AU Diagnostic is divided into two modules so that it will fit into 2KB of memory. After the two AU Modules are executed, a third module which tests the additional instructions found in the 9940 is executed. This third module executes a copy of the ROM Self Test also. The third module executes for about 90 seconds. If an error is detected during the CPU tests an error message number will be printed. The following list describes the error associated with each number.

CPU ERROR MESSAGES

MSG. #	INSTRUCTION THAT FAILED
0000	C
0001	CB
0002	MOV
0003	MOVB
0004	A
0005	AB

0006	S
0007	SB
0008	SOC
0009	SOCB
000A	SZC
000B	SZCB
000C	JMP
000D	JLT
000E	JLE
000F	JEQ
0010	JHE
0011	JGT
0012	JNE
0013	JL
0014	JH
0015	JOP
0016	JNC
0017	JOC
0018	JNO
0019	COC
001A	CZC
001B	XOR
1000	SELF TEST FAILED
1200	DCA
1400	DCS

2001	SRA
2002	SLA
2003	SRL
2004	SRC
2005	LI
2006	AI
2007	ANDI
2008	ORI
2009	CI
200A	STWP
200B	LWPI
200C	CLR
200D	NEG
200E	INV
200F	INC
2010	INCT
2011	DEC
2012	DECT
2013	SETO
2014	SWPB
2015	ABS
2016	B
2017	BL
2018	BLWP
2019	X

201A	RT
201B	DIV
201C	MPY
201D	LIIM

5.2.9 E9 Verb

E9 executes test Part 9. Part 9 tests the Emulator-Trace Module interface. The Emulator and Trace Module interface cables must be connected before starting E9. The control cable must be disconnected to run all other tests. E9 consists of three parts.

Part 1 the Emulator and Trace Module are setup to trace a program to be executed in the Emulator. The Emulators Trace Memory is controlled by TSE from the Trace Module. A BP Address is set in the Emulator, but BP is not enabled. The Trace Module is set to count one external event. When the BP Address is detected, Event is sent to the Trace Module. The Trace Module then sends Halt Request to the Emulator. The Emulator halts and sets Control Mode- to the Trace Module low, halting the Trace Module. The trace data in the Emulator and Trace Module is then checked to verify proper operation of the interface.

Part 2 traces the same program only the Trace Module mask register allows all qualifiers to condition tracing. The

qualifiers allow only reads to be traced.

Part 3 is similar except only odd bytes are traced.

5.2.10 TS Verb

The TS verb performs tests on a special Target System. These tests check the Emulator to Target System interface. The test consists of six parts.

Part 1 tests P0-P31. A 32 bit shift register in the Target System is loaded through the P0-P31 I/O lines. Bit 0 is a zero, all other bits are ones. The zero is then shifted through all bits and read in each bit posistion through the P0-P31 I/O lines. If an error is detected the bit sequence from left to right of the displayed data is, P15-P0 then P31-P16.

Part 2 tests the CRU Expansion function. All 256 CRU addresses are sent to the Target System. Each address is latched up in the target system and checked. A flip-flop also latches the CRU write bit so it can be checked.

Part 3 tests the MPSI interface. The Target System connects the CRUOUT line to the TD line and the CRUCLOCK to TC. To check the MPSI read function 0000, 5555, AAAA and FFFF are sent over the CRU lines to be read by the MPSI interface. The MPSI write function is tested by doing a

write to the MPSI CRU address. The data is latched in the Target System and checked by reading P16-P31.

Part 4 tests the clock output. The Target System has a clock counting latch. This counter is turned on and a 16 bit STCR instruction is executed. The counter is then turned off. The counter should have counted >38 clocks.

Part 5 tests the Hold, Hold Acknowledged and Idle pins. The Emulator is placed in Hold and then in Idle. These states are then read over the Host-Target System interface and checked.

Part 6 tests the interrupts in a manner similar to E7, only the Host-Target System interface controls the interrupts and the EC line.

5.2.11 IT Verb

The IT verb asks the test initialization questions as described in Section 4.2.2.

5.2.12 IE Verb The IE verb is an Emulator Command Verb. An Emulator Command Verb causes a command to be sent to the Emulator using the Emulators Communication Protocol. IE is the "Initialize Emulator" command described in the Emulator Spec. The IE verb will print the buffer type that is

returned by the Emulator. The 9940 Buffer is Type FFFF.

5.2.13 MS Verb

The MS verb causes execution of the "Read/Write Emulator User Memory Selection" command described in the Emulator Spec. The verb asks the user to select read or write. If the user selects write the verb asks for the write data. For a read, the memory selection data is printed.

5.2.14 CK Verb

The CK verb is the "Read Clock Select" command described in the Emulator Spec. This verb prints a message telling which clock is selected.

5.2.15 RR Verb

The RR verb executes the "Read Prototype CPU Registers" command described in the Emulator Spec. The verb asks which register the user wants to read. The user enters 0 for the WP, 2 for the PC or 4 for the ST register. The contents of the selected register will then be printed.

5.2.16 WR Verb

The WR verb is the "Write Prototype CPU Registers' command

described in the Emulator Spec. The verb asks for the data to be written into the register, then asks, which register the user wants to write into. The user enters 0,2 or 4 as in the RR verb. When this verb is used under control of the GC verb, it may be entered as many as 16 times.

5.2.17 RM Verb

The RM verb is the "Read Prototype Memory" or "Read Control Memory" command described in the Emulator Spec. The verb will ask for memory address. The least significant 4 bits will be masked off to start displaying memory on an eight word boundry. 64 words of data will then be displayed. A Control Memory Command is generated for all address's above >2000. This allows reading the memory flag words. When reading the flag words, only the four most significant bits of each word displayed are valid.

5.2.18 WM Verb

The WM verb is the "Write Prototype Memory" or "Write Control Memory" command described in the Emulator Spec. As many as 64 consecutive words of memory data may be enterd with this verb. The first word of write data is asked for, then the number of words remaining to the user is printed and the user is asked if more write data is desired. If no,

the verb asks for the starting address for the data is to be written to. When WM is entered under control of the GC verb, a total of 64 words may be entered for all entries of WM. A Control Memory Command is generated for all address's above >2000.

5.2.19 RC Verb

The RC verb is the "Read Prototype CRU Field" or the "Read Control Program CRU Field" command described in the Emulator Spec. The verb asks for the number of bits to be read. Next, it asks for the CRU base address and prints the read data. The data is right justified. If the base address is greater than >3FF the Control Program CRU Command is generated.

5.2.20 WC Verb

The WC verb is the "Write Prototype CRU Field" or the "Write Control Program CRU Field" command described in the Emulator Spec. The verb asks for the number of bits to be written, the write data and the CRU base address. The write data is right justified. If the base address is greater than >3FF the Control CRU Command is generated. In the GC mode the WC verb may be used as many as 16 times.

5.2.21 DT Verb

The DT verb is the "Define Trace Configuration" command described in the Emulator Spec. The DT verb allows the user to specify one or two trace ranges. The verb asks the user for the trace count. Enter 0 for continuous trace. The verb then asks for the qualifier, lower address bound and upper address bound. It then asks if another trace range is wanted.

5.2.22 DB verb

The DB verb is the "Define Breakpoint Configuration" command described in the Emulator Spec. The DB verb will allow the user to define up to four breakpoint ranges. The test first asks if the user wants to enable the breakpoint. If no, only an event will be generated when a breakpoint would have occured. The verb then asks if breakpoint on an illegal address is wanted. The qualifier, and bounds are asked for next. If more breakpoints are desired the qualifier and bounds for each range will be asked for.

5.2.23 RT Verb

The RT verb is the "Examine Trace Samples" command described in the Emulator Spec. The RT verb reads the Trace Memory and displays the contents. The Trace Count is displayed

following the trace data. If the Trace Count is zero, only the trace count is printed.

5.2.24 RE Verb

The RE verb is the "Run Emulation" command described in the Emulator Spec. RE causes the Emulator to begin executing at the current WP and PC. These may have been entered using the WR verb.

5.2.25 CM Verb

The CM verb is the Control Memory Test command described in the Emulator Spec. CM causes the Emulator to run a checksum on Control Rom and a simple memory test on Control Ram. Only the address is given if Control Ram fails.

5.2.26 GC Verb

The GC verb allows the user to generate a table of Emulator commands which may be executed using the XT and LT commands. By using the GC command, Emulator commands may be issued in any desired sequence at speed. This allows scoping loops to be built which need not contain any unneeded Emulator commands.

During normal operation the Emulator Command Verbs are

executed as soon as they are entered. To build a table of commands the user enters the GC verb. The diagnostic will then ask for the next verb. Each Emulator Command Verb that is entered will then be added to the Command Table, but will not be executed at the time it is entered. When in the GC mode, "*" will be printed after an Emulator Command is entered. This alerts the user that the command was not executed, but was added to the Command Table. The format of the Command Table is described in the SL verb description.

To execute the Command Table the user enters the XT or LT verb. Once either of these verbs is entered all Emulator Commands entered will be executed upon entry and the COMMAND Table will contain only the last command entered. As long as no Emulator Commands are entered after XT or LT is entered, the Command Table will contain all the commands entered while in the GC mode.

5.2.27 XT Verb

The XT verb causes the Command Table to execute one time at speed. When executed this verb takes the diagnostic out of the GC mode.

5.2.28 LT Verb

The LT verb loops on the Command Table at speed. The table

should not contain an RE command unless the user is only interested in the actual transfer of the RE command to the Emulator. When executed this verb takes the Emulator out of the GC mode.

5.2.29 SL Verb

The SL verb prints the contents of the Command Table. This allows the user to verify the commands to be executed and modify them if necessary. A sample SL printout is shown below.

```
5C6A "IE"          0080  0000  1E32  0001
5C72 "RM"(PROTO)   0480  0100  2346  0040
5C7A "WM"(CNTRL)   0A00  8300  23C6  0004
```

As can be seen, each command is a four word control block. The first number in the printout is the address of the first word of the command's control block. Next, a command description is printed. Following the description are the four words of data that make up each command. If FFFF is in the first word of a command block it signals the end of the table.

The first word of the command block contains the Command Byte, the R/W Flag and Arg 1, as described in the Emulator

Spec.

The second word contains Arg 2.

The third word contains the starting address in the host, that data will be read from or stored to during data transfer commands.

The fourth word tells how many words of data will be transfered to or from the Emulator.

5.2.30 HT Verb

The HT verb halts the Emulator. HT will loop on a halt sequence if a non-zero value was enter the last time LP was called.

5.2.31 SS Verb

The SS verb is useful in checking out the Emulator's microcode. The SS verb allows the user to issue single or multiple clocks to the Emulator in the clock step mode. The SS verb tells the user to "ENTER THE NUMBER OF CLOCKS DESIRED". At this time the Emulator clock is stopped. If 1-FFFE is entered, that number of clocks will be issued. After each clock is generated, the clock number and microcode state at the end of the clock are printed. After the entered number of clocks has been generated the

diagnostic repeats it's clock number question. If the value FFFF is entered the diagnostic will ask which microcode state to stop on. Clocks will then be generated until the entered microcode state is detected. If the value 0 is entered, the diagnostic will turn the clock on and ask for a new verb.

5.2.32 LP Verb

The LP verb allows looping on seperate parts of the Execute Verbs. The LP verb asks if ;looping is desired, and if so, which part to loop on. If zero is entered the tests will not loop. If for example, the number four were entered, any Execute Verb that had a Part 4,would execute thru Parts 1,2, and 3 and begin looping on Part 4. The header messages for each part of an Execute Verb are numbered so the user may easily determine which number to loop on. The LP verb must be called again and zero entered to prevent looping. The verbs, SS,SO,SZ, and HT will loop if any non-zero value was last entered during LP.

5.2.33 DS Verb

The DS verb displays the Emulator Status. The status is the Emulators CRU interface to the host. The following is an example of a DS printout.

DATA	HA-	ER	CC	TE	EI-	HD-	IP	MC
AAAA	0	1	1	0	1	1	1	02

The data word is CRU bits 0-15. The data is the last data sent to the Emulator.

HA- is Hold Acknowledged(CRU bit 16).

ER is Emulator Ready(CRU bit 17).

CC is Command Complete(CRU bit 18).

TE is the test bit(CRU bit 20).

EI- is Emulator Idle(CRU bit 21).

HD- is Hold(CRU bit 22).

IP is Interrupt Pending(CRU bit 23).

MC is the Microcode State. This value is constantly changing if the clock is running.

5.2.34 SO Verb

The SO verb lets the user set a bit on the Host to Emulator or Host to Target System interface to a one. The desired bit number must be entered in hex. If a value greater than >FF is entered the Target System CRU base is used. If a

non-zero value was entered during the last call of the LP verb, SO will loop.

5.2.35 SZ Verb

The SZ verb allows the user to set a desired bit to a zero in the same manner that the SO verb allows setting a bit to a one.

5.2.36 PE Verb

The PE verb prints the accumulated error total since the last Execute or Loop Verb was issued. The error count may be larger than actual if a Trace count error occurs. If a trace count error is encountered, all expected and actual data is printed to the end of the expected data table. This allows the user to more easily see what the Emulator should have been doing.

5.2.37 SC Verb

The SP verb asks the usr if the internal or Target System clock is desired and enables the appropriate clock.

5.2.38 SB And RB Verbs

The SB and RB verbs allow setting and removing of

breakpoints in the diagnostic. They are for Code checking and are not intended to be used for hardware testing.

6.0 9940 DIAGNOSTIC ERROR MESSAGES

1 HALT ACKNOWLEDGED FAILED TO RETURN AFTER HALT WAS SENT TO THE EMULATOR

2 EMULATOR READY FAILED TO RETURN AFTER HALT WAS SENT TO THE EMULATOR

3 UNEXPECTED EMULATOR INTERRUPT

4 EMULATOR RETURNED AN ILLEGAL ERROR CODE
ILLEGAL CODE = XXXX

5 EMULATOR READY FAILED TO RESET AFTER HOST READY WAS SET

6 EMULATOR READY FAILED TO SET AFTER HOST READY WAS RESET

7 EMULATOR RETURNED AN INDEX OUT OF BOUNDS ERROR MESSAGE

8 EMULATOR RETURNED AN ILLEGAL COMMAND ERROR MESSAGE

9 EMULATOR RETURNED A SEMANTICS ERROR MESSAGE

10 ECHO COMPARE ERROR. EXPECTED =XXXX ACTUAL =XXXX

11 IE OR SM COMMAND FAILED. EXPECTED SM DATA = XXXX
ACTUAL SM DATA = XXXX

12 MEMORY DATA ERROR ADDRESS = XXXX DATA = XXXX
EXPECTED = XXXX

13 REGISTER DATA ERROR DATA = XXXX EXPECTED = XXXX
REG= XXXX (0=WP 2=PC 4=ST)

14 RUN EMULATION OR EMULATOR IDLE FAILED.

15 BREAKPOINT FAILED TO OCCUR.

16 TRACE COUNT ERROR. EXPECTED = XX ACTUAL = XX

17 TRACE DATA ERROR. EXPECTED = XXXX ACTUAL = XXXX
WORD # XX

18 PC ERROR. EXPECTED VALUE = XXXX ACTUAL
VALUE = XXXX

19 CRU DATA ERROR. EXPECTED = XXXX ACTUAL = XXXX
BASE ADDRESS = XXXX

20 EXPECTED BUFFER TYPE = FFFF. ACTUAL BUFFER
TYPE = XXXX

21 DECREMENTER INTERRUPT DID NOT CLEAR

22 DECREMENTER REGISTER AND T/-C DID NOT CLEAR

23 TEST BIT FAILED TO CLEAR

24 STATUS REGISTER ERROR. EXPECTED = XXXX ACTUAL = XXXX

25 DECREMENTER FAILED TO INTERRUPT AFTER >2000 COUNTS

26 BREAKPOINT OCCURED. IT SHOULD NOT HAVE.

27 DECREMENTER CAUSED AN INTERRUPT ON THE WRONG COUNT.
COUNTS REMAINING AT TIME OF INTERRUPT =>XXXX

28 DECREMENTER FAILED TO RELOAD WITH PROPER VALUE.
EXPECTED = 2000 ACTUAL = XXXX

29 CPU TEST ERROR. MESSAGE # XXXX

30 CPU TEST TIMED OUT. PASS OR FAIL NOT REPORTED.

31 INTERRUPT PENDING DID NOT CLEAR

32 INTERRUPT DID NOT OCCUR WHEN HOST READY WAS TOGGLED

33 INTERRUPT PENDING DID NOT SET WHEN HOST READY WAS TOGGLED

34 DATA ERROR. EXPECTED = XXXXXXXX ACTUAL = XXXXXXXX

35 EXPANSION ADDRESS ERROR. EXPECTED ADDRESS = XX ACTUAL ADDRESS = XX

36 CRU BIT WAS A ZERO. IT SHOULD HAVE BEEN A ONE.

37 CRU BIT WAS A ONE. IT SHOULD HAVE BEEN A ZERO.

38 MPSI READ ERROR. EXPECTED = XXXX ACTUAL = XXXX

39 T.M. CABLE STATUS(BIT 21) INDICATES DATA CABLE NOT PRESENT.

40 T.M. DATA SOURCE STATUS(BIT 23) FAILS. INDICATES EMULATOR IS NOT DATA SOURCE.

41 T.M. HALT PENDING STATUS(BIT 31) FAILED. IT SHOULD HAVE BEEN A ONE.

42 T.M. TRACING STATUS(BIT 20) FAILED. IT SHOULD HAVE BEEN A ZERO.

43 TRACE MODULE DATA ERROR ADDRESS = XX EXPECTED =XXXXX ACTUAL = XXXXX

44 MPSI WRITE ERROR. EXPECTED = XXXX ACTUAL = XXXX

45 CLOCK COUNT ERROR. EXPECTED = XXXX ACTUAL = XXXX

46 TARGET SYSTEM HOLD ACKNOWLEDGED FAILED.

47 TARGET SYSTEM IDLE FAILED.

48 TARGET SYSTEM RESET FAILED.

49 TARGET SYSTEM INT1 FAILED.

50 TARGET SYSTEM INT2 FAILED.

51 TARGET SYSTEM EC INPUT FAILED.

52 CONTROL ROM CHECKSUM ERROR.

53 CONTROL RAM DATA ERROR. ADDRESS = XXXX

7.0 PART NUMBERS

FICHE KIT	2250228-0009 (SP)
	-9001 (PD)
	-2001 (SRC,PD)
EMU940, STANDARD LINK	2250228-1006 (FLO)
	-9006 (LML)
	-7006 (LC)
EMU940BI, BURN-IN LINK	-1009 (FLO)
	-9009 (LML)
	-7009 (LC)
EM9940, TEST MODULE 1	-1003 (OBJ)
	-9003 (LIST)
	-2003 (SRC)
EM942, TEST MODULE 2	2250201-1003 (OBJ)
	-9003 (LIST)
	-2003 (SRC)
EM943, TEST MODULE 3	2250202-1003 (OBJ)
	-9003 (LIST)

	-2003 (SRC)
EM944, TEST MODULE 4	2250203-1003 (OBJ)
	-9003 (LIST)
	-2003 (SRC)
EM945, TEST MODULE 5	2250204-1003 (OBJ)
	-9003 (LIST)
	-2003 (SRC)
EM946, TEST MODULE 6	2250205-1003 (OBJ)
	-9003 (LIST)
	-2003 (SRC)
EM947, TEST MODULE 7	2250206-1003 (OBJ)
	-9003 (LIST)
	-2003 (SRC)
EM948, TEST MODULE 8	2250207-1003 (OBJ)
	-9003 (LIST)
	-2003 (SRC)
EM949, TEST MODULE 9	2250208-1003 (OBJ)
	-9003 (LIST)
	-2003 (SRC)
EM9410, TEST MODULE 10	2250209-1003 (OBJ)
	-9003 (LIST)
	-2003 (SRC)
MSG994, EMU994 MESSAGE MODULE	2250550-1003 (OBJ)
	-9003 (LIST)
	-2003 (SRC)

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED

REV																																
SHEET																																
REV STATUS OF SHEETS	REV																															
	SHEET																															

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± 010
2 PLACE DECIMAL ± 02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL STD 100

DWN DATE

CHK

ENGR

QA

APVD

CONTR NO

DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

PD, HSTSLV, 990/5 HOST/SLAVE TEST-990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	2267305-9901

SCALE | REV. | SHEET 1 of 17

TI-0023.C

TABLE OF CONTENTS

1.0 SCOPE

This document describes the 990/5 HSTSLV DIAGNOSTIC. The diagnostic is designed to verify the proper functioning of the 990/5 computer when the 990/5 is a slave processor.

2.0 REFERENCES.

For information beyond the scope of this document, the reader should refer to the following:

PART NUMBER	DOCUMENT TITLE
945400-9701	Model 990 Computer Diagnostic Handbook.
946294-9701	Model 990/5 Computer Hardware User's Manual.
943441-9701	990/5 Assembly Language Programmers Guide.
	TMS9902 Asynchronous Communications Controller Data Manual.
	TMS9903 Synchronous Communication Controller Data Manual.
945550-0001	EIA Loopback Connector Assembly.

3.0 EQUIPMENT AND SOFTWARE REQUIREMENTS

The following are the minimum requirements for the diagnostic to execute properly:

EQUIPMENT REQUIREMENTS

990/10 OR 990/5 HOST COMPUTER

990/5 SLAVE COMPUTER

APPROPRIATE I/O DEVICE

APPROPRIATE LOADING DEVICE

APPROPRIATE LOAD MEDIA

SOFTWARE REQUIREMENTS

LINKED PROGRAM	HSTSLV	(PN 2267305-1006)
LINKABLE PARTS	HSTSLV	(PN 2267305-1003)
	HTSVMG	(PN 2267306-1003)
	AU05A	(PN 2267314-1003)
	AU05B	(PN 2267308-1003)
	AU05C	(PN 2267309-1003)
	STANDIO	(PN 2267311-1003)
	AU05MSG	(PN 2267312-1003)

Note: If a 990/5 computer is used for the HOST computer it cannot have 64K bytes of onboard memory. To make sure that it will only have 32K bytes of onboard memory set the switch 7 at G09 to the ON position.

Note: The 990/5 computer used as the SLAVE must have switch 6 at G09 in the ON position.

4.0 LOADING

The loading procedures for all loading devices and load media is covered in the DIAGNOSTIC HANDBOOK (PN 943400-9701).

5.0 TESTING

There are two sections of the testing procedure. They are discussed in the following paragraphs:

5.1 HOST - SLAVE TESTING

This test verifies the proper functioning of the INTER-PROCESSOR INTERFACE of the 990/5 computer. This is done in the following manner:

a) host starts slave by an I/O reset

b) slave interrupts host

c) host sends interrupt acknowledge

d) host interrupts slave

e) slave sends interrupt acknowledge

If an error occurs in this test one of the following could be the probable cause for failure:

1) wrong switch setting

2) INTER-PROCESSOR failure on the 990/5

3) TMS 9900 did not recognize or clear the interrupt

The recommended action would be the following:

1) verify that the switches are correct
   (see paragraph 3.0)

2) check the INTER-PROCESSOR interface to
   see if that caused the failure

3) replace the TMS 9900 chip

5.2 SLAVE TESTING

After the completion of the HOST - SLAVE test, AU05 (PN 2267307-1006) is downloaded to the slave 990/5 computer. Then the following is done:

a) an I/O reset is done to start the slave

b) the slave is monitored for messages

The monitoring is accomplished by using a communications area at the beginning of AU05. The HOST processor checks certain locations to see if the slave has a message to output or if the test is done (see the DIAGNOSTIC HANDBOOK for information on the BURN-IN COMMUNICATION FLAGS).

For detailed information on the slave tests see the AU05 PD (section 5.0).

6.0 VERBS

By use of the verbs the operator can control what the diagnostic is doing to some extent. In this particular case that is somewhat difficult since the slave tests consist of a stand-alone diagnostic. In the AU05 PD (section 7.0) there are methods of controlling the slave test. They can be used in this case also.

6.1 IT OR INITALIZE TEST

This verb is used to insert the parameters needed for correct operation of the diagnostic. The operator must answer the following questions:

ENTER THE SLAVE'S CRU BASE, DEF = 1E80

The operator must now enter the target slave's CRU base. The range is >1E80 TO >1EC0.

ENTER INTERRUPT LEVEL OF /5, DEF = OD

The operator must now enter the interrupt level of the target slave.

WHICH MEMORY SWITCH QUESTION DO YOU WANT?
(O= SWITCHES, 1= MEMORY ADDRESS), DEF = 1

To enter the target slave's starting TILINE ADDRESS, the operator must either know the address, or read it from the switches. Then the operator must answer one of the following questions:

FOR MEMORY ADDRESSES

ENTER START OF SLAVE'S MEMORY (EXAMPLE 1FEOOO = MAX.) DEF= OOOOOO

The operator must enter the slave's starting memory address in hexadecimal format.

FOR SWITCH SETTINGS

ENTER THE SLAVE'S MEMORY ADDRESS SWITCH SETTINGS (I.E. OOFF = ALL SWITCHES ON) -

To answer this question all the operator must do is look on the target slave board and read the switches at location TO5.

ENTER THE POWER FREQ. (0= 60 HTZ.,1= 50 HTZ.) DEF= 0

This question is to take into consideration the differences between American and foreign electrical current cycles.

IDLE THE SLAVE ON ERRORS (0= NO, 1= YES) DEF= 0

This question allows the operator the choice of stopping the slave if the slave tests detect a failure.

6.2 EA OR LA, EXECUTE ALL OR LOOP ON ALL TESTS VERB

These verbs will cause the HOST to execute or loop on all of the tests. Upon completion of the EA verb, control is returned to DOCS at which time "VERB" is displayed allowing the operator to choose one of the DOCS verbs or another of the test verbs. The LA verb cause continuous execution of the tests until the operator interrupts the loop, at which time "VERB" is displayed. Again allowing the operator to choose a DOCS verb or another test verb.

6.3 MC, MOVE CODE VERB

This verb will move the slave tests to the target slave over the TILINE. It sets up the slave registers >80 and >82. >82 will contain >9900. >80 will contain the address that is the beginning location of the test. The target slave is now ready for the test to begin execution.

6.4 MS, MONITOR SLAVE

This verb will give the slave an I/O reset to begin execution of the downloaded diagnostic. It will then monitor and display all messages from the slave.

6.5 MM, MODIFY MEMORY

This verb is identical to the .MM verb in DOCS (see DIAGNOSTIC HANDBOOK) except that it works only for the target slave's memory.

7.0 MESSAGES

There are two type of messages associated with the HSTSLV diagnostic. They are discussed in the following paragraphs:

7.1 HOST MESSAGES

The host has two type of messages associated with it. They are described below:

7.1.1 HEADER MESSAGES

HOST SLAVE STAND ALONE TEST

This is the title for the test and is displayed every time the diagnostic is loaded.

START SUBTEST 0 HOST/SLAVE INTERPROCESSOR TEST

INTERPROCESSOR SUBTEST COMPLETE

These messages are the only subtest messages associated with the host.

TEST COMPLETE

This message is displayed when the test runs to completion

7.1.2 ERROR MESSAGES

All error messages associated with the HOST come from subtest 0. They are self-explanitory. For the corrective action the operator should reference paragraph 5.1.

7.2 SLAVE MESSAGES

For all messages from the slave, the operator should refrerence the AU05 PD (section 6.0). These messages are designated as slave messages by the following header:

(SLV MSG)

8.0 PART NUMBERS

TITLE	PART NUMBER	
PROGRAM DESCRIPTION	2267305-2001	ROFF SOURCE
	-9001	ROFF OUTPUT
	-9901	ROFF DOCUMENT
FICHE KIT	2267305-0009	SP
HSTSLV LINKED TEST	2267305-1006	FLO
	-7006	LC
	-9006	LML
TEST PARTS		
HSTSLV	2267305-1003	OBJ
	-2003	SRC
	-9003	LST
HTSVMG	2267306-1003	OBJ
	-2003	SRC
	-9003	LST
AU05A	2267314-1003	OBJ
	-2003	SRC
	-9003	LST

AU05B	2267308-1003	OBJ
	-2003	SRC
	-9003	LST
AU05C	2267309-1003	OBJ
	-2003	SRC
	-9003	LST
STANDIO	2267311-1003	OBJ
	-2003	SRC
	-9003	LST
AU05MSG	2267312-1003	OBJ
	-2003	SRC
	-9003	LST

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																														
SHEET																														
REV STATUS OF SHEETS	REV																													
	SHEET																													

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± 010
2 PLACE DECIMAL ± 02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL STD 100

	DATE
DWN	
CHK	
ENGR	
QA	
APVD	
CONTR NO	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

PD, LP810 MC-810 LINE PRINTER TEST-990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	2250106-9901

SCALE	REV. B	SHEET 1 of 31

Y1-0023-C

TABLE OF CONTENTS

1.0 SCOPE

The LP810 PRINTER DIAGNOSTIC (2250106) is used to test the TEXAS INSTRUMENTS MODEL 810 PRINTER when the unit is connected to a 990 computer using the TTY/EIA INTERFACE MODULE (P/N 945075), a TMS 9902 ASYNCHRONOUS COMMUNICATIONS CONTROLLER, or a TMS 9903 SYNCHRONOUS CONTROLLER in asynchronous mode.

2.0 REFERENCES

For information beyond the scope of this document refer to the following:

PART NUMBER	TITLE
938751-9901	Model 810 Printer Test Procedure
938151-9901	Specification, Model 810 Printer
994353-9701	Operating Instructions Model 810 Printer
945408-9701	TTY/EIA Interface Module Depot Maintenance Manual
945417-9701	Model 990/10 Computer System Hardware Reference Manual
945251-9701	Model 990/4 Computer System Hardware Reference Manual
2261934	Model 990/5 Computer CPU Specification Drawing
945400-9701	Diagnostic Handbook
945403-9701	Model 990/4 Computer System Depot Maintenance Manual

945404-9701	Model 990/10 Computer System Depot Maintenance Manual
	TMS 9902 Asynchronous Communications Controller Data Manual
	TMS 9903 Synchronous Communications Applications Manual

3.0 REQUIREMENTS

This section describes the hardware and software requirements for running the LP810 diagnostic.

3.1 HARDWARE REQUIRED

In addition to hardware described in the DIAGNOSTIC HANDBOOK (P/N 945400-9701) the following equipment is required to run the test:

Module 810 Printer Interface Kit (P/N 938120)

990 Computer System with at least 8K words of memory

3.2 SOFTWARE DESCRIPTION

The LP810 diagnostic can be run on the 990 family computers. The LP810 diagnostic is a test that runs under control of DOCS.

LP810, LINE PRINTER DIAGNOSTIC LINK 2250106-1006

Linkable parts: LP810

PRINTSUB

TSTPK1

TSTPK2

TSTPK3

LP810MSG

4.0 LOAD PROCEDURE

The procedure for loading the LP810 Diagnostic is given in the Diagnostic Handbook.

5.0 TEST EXECUTION AND DESCRIPTION

This section describes how to execute the LP810 diagnostic.

5.1 SOFTWARE INITIALIZATION

The LP810 Diagnostic will begin execution by printing the following message:

LP810 MODEL MC-810 PRINTER TEST XX/XX RR

Where XX/XX is the release date and RR is the revision level of the LP810 Diagnostic. Use this date to match the object deck and assembly listing.

A series of initialization questions will then be asked. If the default is correct, press the return key. If not, enter the correct value and press the return key.

Then the LP810 Diagnostic requires the following set of questions to be answered:

```
ENTER PRINTER CRU BASE                                  DEFAULT = 0060 -
USE THE TTY/EIA BOARD (0) OR TMS 9902/9903 (1)          DEFAULT = 00 -
ENTER CLOCK FREQUENCY (0=3MHZ,1=4MHZ)                   DEFAULT = 00 -
ENTER PRINTER INTERRUPT LEVEL                           DEFAULT = 0E -
DO YOU WANT TO RUN TEST WITH INTERRUPTS ?               DEFAULT = 01 -
DOES PRINTER HAVE FULL ASCII CHARACTER SET ?            DEFAULT = 01 -
DOES PRINTER HAVE VCO OPTION ?                          DEFAULT = 01 -
IS THE COMPUTER LINE CURRENT 60 HZ ?                    DEFAULT = 01 -
DO YOU WANT TO RUN THE MANUAL INTERVENTION TEST         DEFAULT = 01 -
ENTER COLUMN WIDTH (0=132,1=80)                         DEFAULT = 01 -
EXECUTE EA VERB? (DEF=1) -
```

If the entire test is to be run, enter a '1' to execute all. If not enter a '0'. If the above parameters need to be changed, use the IT verb.

DOCS will prompt the operator by printing VERB ? -. A two letter verb from the following list can then be entered.

ET -- Execute Test

LT -- Loop On Test

EA -- Execute All tests in order

LA -- Loop On All tests in order

IT -- Initialize Test

The TMS 9902 and TMS 9903 sections in the diagnostic assume certain CRU bases as established in the 990/5 specification. These sections were written specifically for use of the MC-810 PRINTER off the communications ports on the 990/5.

CRU BASE 1700 (9902 PORT)

CRU BASE 1740 (9902 PORT)

CRU BASE 1780 (9903 PORT)

5.2 TESTS PERFORMED

This section describes the tests available in the LP810 Diagnostic.

5.2.1 TEST 01

Test 01 does not print on the MC-810 printer. It tests the following functions of the printer interface (TTY/EIA card, TMS 9902, or TMS 9903) attached to the MC-810 printer:

A. The Data Set Ready bit is tested in the following manner. With the printer deselected, to inhibit printing, the Data Set Ready bit is set to a 1 to enable the transmitting of a character to the line printer. At this time the Data Set Ready bit is tested to ensure that it has been set and transmitting is enabled.

B. At the same time that the Data Set Ready bit is being tested the Transmit bit on TTY/EIA or Transmit Buffer Register Empty on TMS 9902/9903 are also tested. With a character being transmitted to the line printer the Transmit bit is tested to verify it is set to a logical 1.

C. The Interrupt capability is tested during the execution of the character transmiting routine by having all possible interrupts enabled.

D. The Baud rate is calculated and tested to verify it is within tolerances for 4800. This is done by determining the number of characters transmitted within 75 ms and comparing the result to tolerances set forth for the appropriate line current, 50 or 60Hz, selected at the beginning of the diagnostic.

Shown below are the pencil switch settings for the possible baud rates for the LP810 Line Printer.

BAUD RATE	SWITCH		
	1	2	3
110	OFF	OFF	OFF
150	ON	OFF	OFF
300	OFF	ON	OFF
1200	ON	ON	OFF
2400	OFF	OFF	ON
4800	ON	OFF	ON
9600	OFF	ON	ON
PARALLEL	ON	ON	ON

5.2.2 TEST 02

This is the first test that prints on the MC-810 printer. It uses the line feed character to force printing. This test prints each character in the character set 20 times on one line. The full ASCII or the Standard ASCII character set is used according to the option chosen by the operator at initialization time. All characters should be checked to verify that every character belonging to the selected set printed correctly. Detected error conditions will cause an error message to be printed on the interactive device.

```
***** BEGIN TEST 02 CHARACTER SET TEST *****
!!!!!!!!!!!!!!!!!!!!
"""""""""""""""""""""
####################
$$$$$$$$$$$$$$$$$$$$
          .
          .
          .
```

5.2.3 TEST 03

This test prints a ripple dump. The ripple dump contains 80 or 132 characters per line depending on the option selected by the operator and consists of the full selected character set. Line feeds are used to force printing. Each character appears once in each column. The full character set ripple dump contains 95 lines and the standard character set ripple dump contains 63 lines. Check to verify that all charcters did print in every column. The omission a of character from a column or the printing of a partial line will cause an error message to be printed on the interactive device.

```
***** BEGIN TEST 03 RIPPLE DUMP *****
!##!& ()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZ...
##!& ()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZa...
#!& ()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZab...
!& ()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUVWXYZabc...
                              .
                              .
                              .
```

5.2.4 TEST 04

This is the Print buffer test. This tests the Printers ability to handle different sized Print buffers by Printing a diamond Pattern of characters in the form of a ripple dump. There will be either 80 or 132 characters Per line depending on the option selected. Check to verify that all columns within the diamond contain a character and that no Partial lines have been Printed. Detected error conditions will cause an error message to be Printed on the interactive device.

```
***** BEGIN TEST 04 PRINTER BUFFER TEST *****
                                                !"
                                               !"##$
                                              !"##$!&
                                             !"##$!&'(
                                            !"##$!&'( )*
                                           !"##$!&'( )*+,
                                          !"##$!&'( )*+,-.
                                                .
                                                .
                                                .
                                          !"##$!&'( )*+,-.
                                           !"##$!&'( )*+
                                            !"##$!&'( )
                                             !"##$!&'(
                                              !"##$!&
                                               !"##$
                                                !"
```

5.2.5 TEST 05

This is the carriage return test. The carriage return feature is tested by printing an english sentence with every other character omitted, sending a carriage return, and printing the the characters that were omitted before. If the carriage return operates correctly, the following message is printed on one line in a readable form:

THIS LINE IS BEING PRINTED IN TWO PARTS VERIFY THAT ONLY ONE LINE IS PRINTED

Otherwise the line will not be in a readable form.

5.2.6 TEST 06

This is the programmable forms length test. First the forms length is set to 33 lines. Then a form feed command is given and the following message is printed on the MC-810 under test:

THIS IS THE CENTER OF FORM PAGE ONE OF TEST 06

The forms length is the set to 66 and the following message is printed on the top of the next page:

FORM FEED THIS IS TOP OF FORM PAGE TWO OF TEST 06

5.2.7 TEST 07

This tests the tab to line command. This test prints the following message ten times on the MC-810 under test:

THIS IS LINE XX AS NUMBERED ON THIS FORM YY LINES FROM TOP OF FORM

Where XX takes the values 1,2,4,8,16,32,40,44,46 and 47; and YY takes the values 4,5,7,11,19,35,43,47,49 and 50. On TI form 12461 the XX value corresponds to line number printed in the left margin of the form. The YY value is the count of lines from the top of form. The operator should verify that this information is correct.

5.2.8 TEST 08

This tests the set vertical tab command and the vertical tab command. This test prints the following message ten times on the MC-810 under test:

THIS IS LINE XX AS NUMBERED ON THIS FORM YY LINES FROM TOP OF FORM

Where XX takes the values 1,2,4,8,16,32,40,44,46 and 47; and YY takes the values 4,5,7,11,19,35,43,47,49 and 50. Vertical tabs are set on the lines corresponding to the YY values. On TI form 12461 the XX value corresponds to line number printed in the left margin of the form. The YY value is the count of lines from the top of form. The operator should verify that this information is correct.

5.2.9 TEST 09

This tests horizontal tab set command and horizontal tab command. After numbering the columns, up-arrows are printed in the following columns: 10, 15, 20, 40, 80, 90, 115, 117, 119 and 128. The operator should verify that this information is correct:

```
***** BEGIN TEST 09 SET AND TEST HORIZONTAL TABS *****
         1         2         3         4         5         6
12345678901234567890123456789012345678901234567890123456789...
         !    !    !                   !                   ...
```

Printing is done only through column 80 if the option is selected in IT.

5.2.10 TEST 0A

This tests tab to column address command. After numbering the columns, up-arrows are printed in the following columns: 10, 15, 20, 40, 80, 90, 115, 117, 119 and 128. The operator should verify that this information is correct:

```
***** BEGIN TEST 0A TAB TO ADDRESS *****
         1         2         3         4         5         6
12345678901234567890123456789012345678901234567890123456789...
         !    !    !                   !                   ...
```

Printing is done only through column 80 if the option is selected in IT.

5.2.11 TEST OB

This tests line width set command. The columns are numbered with the line width set to 132, then the line width set to 80. With this line width, 240 'A' characters are sent to the printer. These characters are printed on three lines with 80 characters per line. the line width is then reset to 132 characters per line if column width of 132 was selected by the operator, and 132 'B' are printed. The operator should verify that this information is correct:

```
***** BEGIN TEST OB SET LINE WIDTH *****
         1         2         3         4         5         6
12345678901234567890123456789012345678901234567890123456789012345678901234567890...

THE NEXT THREE LINES ARE ONLY 80 CHARACTERS LONG

AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA...
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA...
AAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAAA...

THE NEXT LINE IS 132 CHARACTERS LONG

BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB...
```

5.2.12 TEST 0C

This tests the printer lines per inch option. The printer is set to eight lines per inch, then eight lines of ten characters each are printed. The printer is then reset to six lines per inch, and six lines of ten characters each are printed. The operator should verify that this information was printed:

***** BEGIN TEST 0C PRINT 8 LINES PER INCH *****
THE NEXT 8 LINES ARE BEING PRINTED AT 8 LINES PER INCH

8888888888
8888888888

.
.
.

THE NEXT 6 LINES ARE BEING PRINTED AT 6 LINES PER INCH

6666666666
6666666666

.
.
.

5.2.13 TEST OD

This tests the printer characters per inch option (VCO and FCO printers only). The printer is set to 16.5 characters per inch, and six lines of seventeen characters each are printed. The printer is then reset to ten characters per inch, and six lines of ten characters each are printed. The operator should verify that this information was printed:

***** BEGIN TEST OD PRINT 16.5 CHARACTERS PER INCH
THE NEXT 6 LINES ARE BEING PRINTED AT 16.5 CHARACTERS PER INCH

AAAAAAAAAAAAAAAA

AAAAAAAAAAAAAAAA

.

.

.

THE NEXT 6 LINES ARE BEING PRINTED AT 10 CHARACTERS PER INCH

BBBBBBBBBB

BBBBBBBBBB

.

.

.

5.2.14 TEST 0E

This tests the printer vertical format control option (VCO printers only). Store and recall vertical format information commands are tested, VFC channel 7 is programmed and stored, and VCF channel 8 is programmed and stored. Then VFC channel 7 is recalled and six lines of the followins message are printed on the MC-810 under test:

THIS IS LINE XX AS NUMBERED ON THIS FORM YY LINES FROM TOP OF FORM

Where XX takes the values 1, 2, 4, 8, 16 and 30; and YY takes the values 4, 5, 7, 11, 19 and 33. VFC channel 7 vertical tabs are set on the lines corresponding to the YY values except 33 which is top of form. On TI form 12461, the XX value corresponds to the line number printed in the left margin of the form. The YY value is the count of lines from the top of form. Channel 8 is recalled and the above message is repeated on the second page of the test. The XX takes the values 3, 5, 17, 37 and 57; and YY takes the values 6, 8, 20, 40 and 60. The operator should verify that this information is correct:

***** BEGIN TEST OE STORE AND RECALL VFC CHANNEL *****

THIS IS LINE 1 AS NUMBERED ON THIS FORM 4 LINES FROM TOP OF FORM

THIS IS LINE 2 AS NUMBERED ON THIS FORM 5 LINES FROM TOP OF FORM

THIS IS LINE 4 AS NUMBERED ON THIS FORM 7 LINES FROM TOP OF FORM

.

.

.

5.2.15 TEST OF

Test OF, the manual intervention test, is executed according to the options chosen by the operator. When executed, the operator is instructed on the selected DOCS I/O device:

BEGIN MANUAL INTERVENTION TESTING POWER DOWN THE PRINTER
DEPRESS RETURN KEY

The power ON/OFF switch on the rear of the printed should be turned off and the return key on the I/O device depressed. At this time the Data Set Ready bit is tested to verify that it has been reset to a 0 with the printer powered off. After doing this the following message will

be printed on the I/O device:

POWER UP THE PRINTER BUT DO NOT PUT THE UNIT ON LINE
DEPRESS RETURN KEY

Turn the power ON/OFF switch to ON, do not put the printer on line, and depress the return key on the I/O device. The Data Set Ready bit is now being tested to verify that it has been reset to a 0 after the printer was powered on and left off line. The printer is then selected, put on line, and the following message should be printed on it:

VERIFY THE BELL AND THAT THE PRINTER ON LINE LED IS OFF

After printing the message the printer is deselected causing the on line led to go off. Verify that the message is correct, that the bell sounds, and that the on line led is off. Once this is verified respond to the message now displayed on the I/O device:

VERIFY MESSAGE ON PRINTER
DEPRESS RETURN KEY

After depressing the return key on the I/O device the printer is selected, put on line, and the parity error detection feature is tested with the following message being printed on the line printer:

VERIFY THAT THE NEXT LINE PRINTED IS 20 PARITY ERROR SYMBOLS
(twenty parity error, ,symbols)

If the parity characters are not printed the pencil switch settings on the inside, front of the printer should be checked against the chart shown below.

PARITY	SWITCH	
	4	5
IGNORE	OFF	OFF
ODD	ON	ON
EVEN	ON	OFF

If in error the switches should be reset to the correct value and the test restarted.

If the parity characters are printed the message displayed on the I/O device should be responded to by depressing the return key:

VERIFY MESSAGE ON PRINTER

With the following message, verifying the end of the test, being printed on the line printer:

END OF MANUAL INTERVENTION TEST

5.2.16 TEST 10

Test 10, the elongated character test, checks to make sure that a LP810 line printer with the elongated character option installed is able to print characters at the rate of 5 characters per inch(VCO and FCO printers only). A printer not having the elongated character option will print characters at the rate of 10 per inch when this test is run.

When this test is run a ripple dump containing the elongated characters is printed for 20 lines. Only upper case letters are included in this character set.

The following messages will appear on the printer during this test:

*****BEGIN TEST 10 PRINT 5 CHARACTERS PER INCH *****

THE NEXT 20 LINES ARE BEING PRINTED AT 5 CHARACTERS PER INCH

5.3 VERB DESCRIPTIONS

This section describes the verbs available in the LP810 Diagnostic.

5.3.1 ET - EXECUTE TEST

FORMAT: VERB ? -ET.....TEST -XX

ACTION: Execute the test specified by XX, a hexidecimal test number.

5.3.2 LT - LOOP ON TEST

FORMAT: VERB ? -LT.....TEST -XX

ACTION: Loop on test number XX displaying the loop count on the front panel. The test loop may be terminated by entering a '.

5.3.3 EA - EXECUTE ALL TESTS

FORMAT: VERB ? -EA

ACTION: All tests (>01 through >10) are executed.

5.3.4 LA - LOOP ON ALL TESTS

FORMAT: VERB ? -LA

ACTION: Loop on execution of all tests in order, displaying the loop count on the front panel each time all tests are completed.

5.3.5 IT - INITIALIZE TEST

FORMAT: VERB ? - IT

ACTION: The test initialization questions described in section 4.2 are asked. This verb allows the operation to change the test parameters.

6.0 ERROR MESSAGE NUMBERS FOR LP810 DIAGNOSTIC

The following error messages may be printed and are considered to be self-explanatory in all cases, assuming some familiarity with the MC-810 specifications.

This diagnostic also contains an alternate error message mode which can be selected by setting the MNTFLG variable in the PRINTSUB module to a nonzero value. This can be done using the front panel or by using the .MM verb available in DOCS. This error message mode prints the following message in addition to the message that would normally be printed.

*** ERROR CODE = XXXX PC = XXXX ***

The PC referred to is the program counter value when the error was detected.

HEX NO MESSAGE AND CONDITION

*** ERROR CODE = XXXX PC = XXXX ***

1 ***** ERROR TEST 01 MUST RUN WITHOUT ERROR BEFORE RUNNING ANY OTHER TEST

This message is printed along with any error message occurring in Test 01 and is meant as an informational message.

2 ERROR BAUD RATE LESS THAN 4800

The Interface Module was unable to transmit at least 30 characters at 60Hz or at least 36 characters at 50Hz, indicating a Baud transfer rate less than 4800. The Transmit bit, CRU Input bit 8, is tested for a set/reset condition to calculate the Baud rate. Check the pencil switch seetings in the inside, front of the printer against the chart located in Test 01 section.

3 ERROR BAUD RATE GREATER THAN 4800

The Interface Module transmitted more than 34 characters at 60Hz or more than 40 characters at 50Hz, indicating a Baud transfer rate greater than 4800. The Transmit bit, CRU Input bit 8, is tested for a set/reset condition to calculate the Baud rate. Check the pencil switch settings in the inside, front of the printer against the chart located in Test 01 section.

4 ERROR TRANSMIT STILL IN PROGRESS AFTER TIMED OUT

The Data Terminal Ready bit, CRU Output bit 9, and the Rquest To Send bit, CRU Output bit A, are set to a 1 forcing the Transmit bit, CRU Input bit 8, to set to a 1. The error occurs when the Transmit bit fails to reset to a 0 after 25ms.

5 ERROR PRINTER IS OFF LINE
DATA SET READY IS LOW AFTER TIME OUT

With the Write Request bit, CRU Input bit B, set to a 1 and data available for transmittal to the printer the Data Set Ready bit, CRU Input bit E, fails to set to a 1 after 6 seconds.

6 ERROR EXPECTED NEW STATUS FLAG INTERRUPT
DID NOT OCCUR (DSR HIGH-TO-LOW)

This error is reported only if running with interrupts. The New Status Flag failed to set to a state of 1 after the Data Set Ready bit, CRU Input bit E, changed states, from a 1 to a 0.

7 ERROR EXPECTED NEW STATUS FLAG INTERRUPT
DID NOT OCCUR (DSR LOW-TO-HIGH)

This error is reported only if running with interrupts. The New Status Flag failed to set to a state of 1 after the Data Set Ready bit, CRU Input bit E, changed states, from a 0 to a 1.

8 ERROR EXPECTED WRITE REQUEST INTERRUPT DID NOT OCCUR

This error is reported only if running with interrupts. The Write Request bit, CRU Input bit B, failed to set to a 1 indicating that the Transmit Shift Register on the TTY/EIA has not yet finished sending a character to the printer, even though the Transmit bit, CRU Input bit 8, has been reset to a 0 state indicating a character is not being transmitted to it.

9 ERROR UNEXPECTED INTERRUPT AT PRINTER INTERRUPT LEVEL

This error in reported only if running with interrupts. An unexpected interrupt, not originating from the line printer, was detected.

A ERROR DSR TRUE WHILE PRINTER IS POWERED DOWN

This error is reported only during the Manual Intervention Test, Test OF. The Data Set Ready bit, CRU Input bit E, was in a state of 1 after the line printer was powered off.

B ERROR DSR TRUE WHILE PRINTER IS OFF LINE

This error is reported only during the Manual Intervention Test, Test OF. The Data Set Ready bit, CRU Input bit E, was in a state of 1 after the line printer was powered on and left in an off line condition.

C ERROR DSR LOW-TO-HIGH NEVER SET, LINE BUFFER NEVER RECEIVED CHARACTERS

The printer failed to print something that was transmitted to it as measured by the Data Set Ready bit, CRU Input bit E, not going from a state of 0 to a state of 1.

7.0 PART NUMBERS

TITLE	NUMBER	
PROGRAM DESCRIPTION	2250101-2001	ROFF Source
	-9001	ROFF Output
FICHE KIT	2250106-0009	SP
LP810 - Linked Test	2250106-1006	FLO
	-7006	LC
	-9006	LML
LP810 - Main Module	2250106-1003	OBJ
	-2003	SRC
	-9003	LIST
PRINTSUB - Print Subroutines	937946-1003	OBJ
	-2003	SRC
	-9003	LIST
TSTPK1 - Test Package 1	937947-1003	OBJ
	-2003	SRC
	-9003	LIST
TSTPK2 - Test Package 2	937948-1003	OBJ
	-2003	SRC
	-9003	LIST
TSTPK3 - Test Package 3	937949-1003	OBJ

	-2003	SRC
	-9003	LIST
LP810MSG - Test Messages	2250246-1003	OBJ
	-2003	SRC
	-9003	LIST

A

APPLICATION	
NEXT ASSY	USED ON
	7506

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

REV
SHEET

REV STATUS OF SHEETS — REV / SHEET

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ANGLES ±1° 3 PLACE DECIMAL ±.010 2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

DWN | DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS INCORPORATED
Equipment Group Dallas, Texas

PD, LPTEST, LINE PRINTER DIAGNOSTIC-990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	2250123-9901

SCALE | REV. A | SHEET 1 of 19

TI-0023-C

TABLE OF CONTENTS

1.0 SCOPE

The line printer diagnostic is used to test any of the following Centronics line printers when they are connected to a TI-990 computer through a TTY/EIA card PN=945075-0001

Centronics	301	80 char line	165 char sec
Centronics	306	80 char line	100 char sec
Centronics	500	132 char line	100 char sec
Centronics	501	132 char line	165 char sec
Centronics	588	132 char line	88 char sec
Centronics	101AL	132 char line	165 char sec
Centronics	102AL	132 char line	330 char sec

2.0 REFERENCES

TITLE	PART NUMBER
Specification 990 Serial Line Printer Kit	974991
Model 990 Reference Manual	943442-9701
Diagnostic Handbook	945400-9701
Manual, Model 306 Line Printer	974993-9701
Manual, Model 500 Line Printer	974998-9701

3.0 EQUIPMENT AND SOFTWARE REQUIREMENTS

This section describes the minimum equipment requirements and necessary software for the LPTEST diagnostic.

3.1 EQUIPMENT REQUIREMENTS

In addition to hardware required by DOCS and specified in the Diagnostic Handbook, one of the following devices is required.

990 Line Printers

-Model 306 Impact Line Printer Kit	945113-0001
-Model 588 Line Printer Kit 5x7	945112-0001
-Model 588 Line Printer Kit 9x7	945112-0004

3.2 SOFTWARE REQUIREMENTS

The line printer diagnostic runs on a 990 computer. The line printer diagnostic is a test that runs under control of DOCS. The operator I/O can be through an appropriate interactive device or the programmers front panel depending on which version of DOCS is used. The test module is: LPTEST, Linked Object: 2250123-1006(FLO)
Linkable Parts: LPTEST

LPMSG

4.0 LOADING

Loading procedures for all available media are found in the Diagnostics Handbook.

5.0 TEST EXECUTION AND DESCRIPTION

The following set of questions need to be answered to initialize LPTEST. These are the same questions asked by the IT verb.

AC LINE FREQ 50 OR 60 HZ - DEFAULT=60HZ -

PRINTER MODEL (101,102,301,306,500,501, OR 588) 306-

LP UNDER TEST CRU BASE ADDRESS - DEFAULT=0060 -

LP UNDER TEST INT LEVEL -DEFAULT=000E -

After these questions are answered, the test can be executed or any of the two letter verbs from the following list can be entered.

5.1 AVAILABLE VERBS

These are the verbs contained within the LPTEST Diagnostic Module. Also available are all the verbs supported by DOCS.

E1	Execute Part 1
E2	Execute Part 2
E3	Execute Part 3
EA	Execute Part 1-3
IT	Initialize Test

5.2 PARTS

The following is a list of all the functions that are tested by each of the parts of the LPTEST.

5.2.1 PART 1 - E1 VERB

1.) Check Read Request

2.) Check Data Set Ready and Data Terminal Ready

3.) Check Interrupt ID Line

4.) Check Transmitting, Write Request and Interrupt Lines

5.) Check Data Set Ready Interrupts

6.) Check Baud Rate

7.) Check LINE/MIN Rate

5.2.2 PART 2 - E2 VERB

1.) Check Power OFF/ON

2.) Check SELECT/DESELECT Switch

3.) Check Elongated Character Feature

4.) Check Form Feed

5.) Check Vertical Tab

6.) Check Character Parity

7.) Check Printer Bell

8.) Check Buffering for Different Line Sizes

5.2.3 PART 3 - E3 VERB

1.) Check Ripple Pattern

5.3 VERB DESCRIPTION

A brief description of each verb and the action taken by each of them is given in the following sections. Error messages that may be printed by any of the verbs

are given in a separate section.

5.3.1 E1 VERB - EXECUTE TEST 1

This test does the following to verify the line printer interface is working.

A.) Check Read Request (it should be 0).

B.) Check Data Set Ready and Data Terminal Ready.

C.) Check the Interrupt ID Line.

D.) Send a reset and top of form character to check that the Transmitting, Write Request, and Interrupt Lines work and that an interrupt occurs.

E.) Send a line to check Data Set Ready interrupts.

F.) Check the Baud Rate and print it in decimal.

The message printed is:

LP BAUD RATE= XXXX (DEC), WHERE XXXX=THE DECIMAL BAUD RATE FOR THE INTERFACE.

(Refer to the line printer specification for the correct value).

G.) Check the LINE/MIN Rate and print it in decimal.

The message printed is:

LP LINE/MIN=XXXX (DEC), WHERE XXXX=THE DECIMAL LINE/MIN

(Refer to the line printer specification for the

correct value).

5.3.2 E2 VERB - EXECUTE TEST 2

This test checks certain basic and special functions for the line printer. The functions checked and reported results are given below.

A.) POWER OFF/ON; SELECT/DESELECT SWITCH

After the following message, the line printer power should be turned off and the user should respond to continue:

TURN OFF LINE PRINTER -
5 SEC DELAY

The second line is printed to notify the user of the delay and to wait for transients to fade out. At this time Data Set Ready is tested for a 0.

The next message should be:

TURN ON AND SELECT LP-

After turning on and selecting the line printer the user should respond to continue. The following message is printed:

3 SEC DELAY

After a three second delay, Data Set Ready is tested for a 1.

The next check is for the deselect switch. The following message should be printed:

DESELECT LP-

The line printer should be deselected and the operator should continue. A three second delay will follow and will be signified by a message. At this time, Data Set Ready is tested for a 0.

The line printer should be selected and the operator should respond to continue testing.

B.) ELONGATED CHARACTER CHECK

This verifies that the elongated character feature of the line printer is functional. Note that two lines will be printed on an 80 column printer and one line on a 132 column printer. The following message is printed preceding the print on the line printer:

VERIFY ELONGATED CHARS =
"#%'()+./0123456789:;<=>?ABCDEFGHIJKLMNOPQRSTUVWXYZ(/)-

The characters printed on the line printer should be checked against the line above to verify the correct characters were printed. The characters printed are in the ASCII range of >20 to >5F.

Note: The character set for the line printer may vary from the comparison standard printed above.

Following the elongated character print on the line printer, the following message will be printed on the interactive device:

RETRY ?-

If the user wants to print the line again, he should enter a 1.

C.) FORM FEED TEST

This test is signaled by :

TOP OF FORM TEST

Being printed on the interactive device. The following message should appear on the line printer:

SKIP TO NEXT PAGE

Which should be followed by a form feed. Next, this message will be printed on the printer:

SKIP TWO PAGES

Which should be followed by two form feeds.

D.) VERTICAL TAB TEST

This test checks the vertical tabbing function of the line

printer. The following message is printed on the interactive device to signify the beginning of the test:

VERTICAL TAB TEST

After this message, beginning at the top of form, lines of the following form should be printed one inch (6 lines) apart on the page until eleven lines are printed:

TAB LINE NO. XX , WHERE 01<XX<11.

E.) CHARACTER PARITY TEST

This test verifies that the even parity checking of the printer circuitry is functioning correctly by sending the incorrect parity bit for each of the sixty-four printable characters. The following message precedes the test and is printed on the interactive device:

LP CHARACTER PARITY CHECK CHARACTERS SHOULD BE @'S

If the printer circuitry is working correctly, a line of 64 @ signs should be printed on the printer.

F.) PRINTER BELL TEST

To test the printer bell, the following message is printed on the interactive device followed by a two second audible tone from the printer bell.

LP BELL TEST

G.) BUFFERING FOR DIFFERENT LINE SIZES

This test is designed to test the line printer capabilities for buffering different size lines. The test starts by printing a full line and continues by sending lines of one character less until one character is sent on a line. When this occurs, the cycle reverses until a full line is printed again. (Any error messages that may occur will be printed by the generalized print routines.)

```
LINE PRINTER BUFFER TE
LINE PRINTER BUFFER T
LINE PRINTER BUFFER
LINE PRINTER BUFFER
LINE PRINTER BUFFE
LlNE PRINTER BUFF
LINE PRINTER BUF
LINE PRINTER BU
LINE PRINTER B
LINE PRINTER
LINE PRINTER
LINE PRINTE
LINE PRINT
LINE PRIN
LINE PRI
LINE PR
LINE P
LINE
LINE
LIN
LI
L
```

5.3.3 E3 - EXECUTE TEST 3

This test prints a ripple pattern on the line printer.

The following message will appear on the interactive device.

RIPPLE DUMP TEST-HIT ANY KEY TO TERMINATE

As per the instruction, enter any key on the interactive Device to terminate the ripple dump and return control to The verb decoder.

5.3.4 EA - EXECUTE ALL TESTS

Entering this verb will cause all three tests to be executed.

5.3.5 IT VERB - INITIALIZE TESTS

Use the "IT" verb for reinitialization of the LP Test. (I.e., a line printer in a different CRU location, at A different interrupt level, or a different model number). This feature also circumvents reloading the software to Change the hardware configuration.

Below are the questions asked by the IT Verb.

AC LINE FREQ 50 OF 60 HZ - DEFAULT=60HZ-

PRINTER MODEL (101,102,301,306,500,501, OR 588) 306-

LP UNDER TEST CRU BASE ADDRESS - DEFAULT=0060 -

LP UNDER TEST INT LEVEL - DEFAULT=000E -

5.4 STANDARD DOCS INITIALIZATION

Reference Diagnostics Handbook

5.5 PROGRAMMERS PANEL DOCS INITIALIZATION

If the front panel version of DOCS is being used, the test operates the same way but all I/O is through the front panel. When DOCS is brought up, it will ask the question,

'IDLE ON ERRORS'. Once this is answered, the questions listed in the previous section will be asked. After answering these the test will start execution of the EA verb. See the Diagnostic Handbook on how to interface through the front panel.

6.0 ERROR MESSAGES

Given below are the error numbers,messages and conditions given for line printer diagnostics. These messages are printed when an error is detected in the line printer general output routine.

NO.	ERROR MESSAGES AND CONDITIONS
1	Message: ERROR 1 RRQ=1 Condition: Read Request, bit C, is set to a 1. It should be set to a 0.
2	Message: ERROR 2 DSR=0 Condition: Data Set Ready, CRU input bit E, is set to a 0. It should be a 1.
3	Message: ERROR 3 INT=1 Condition: Interrupt ID, CRU input bit F, is set to a 1. It should be set to a 0.
4	Message: NO WRQ INT AFTER CHARACTER Condition: A Data Set interrupt did not occur after

sending a character that causes a line to be printed.

5 Message: DSR INT WITH DTR RESET

Condition: A Data Set interrupt has occurred, making the line printer ready to accept characters from the Interface while the Data Terminal Nterface is in a reset state. This disenables the line printer from accepting characters from the Interface board.

6 Message: INT ERROR-BAUD RATE CHECK ABORTED

Condition: A Write Request Interrupt did not occur. This caused an error interrupt to occur while measuring the baud rate, aborting the rate.

7 Message: INT ERROR-LINE/MIN RATE ABORTED

Condition: Either a Write Request Interrupt did not occur or a Write Request Interrupt did occur and a Data Set Ready Interrupt did not occur causing an error interrupt to occur while trying to measure the LINE/MIN rate of the printer. the count is aborted.

8 Message: DSR=1 AFTER DESELECT

Condition: The Data Set Ready (printer ready), CRU input bit E, was not reset to 0 after deselection of the line printer.

9 Message: DSR=0 AFTER SELECT-PRESS SELECT ON LP

Condition: The Data Set Ready (printer ready), CRU input bit E, was not set to a 1 after selection of the line printer.

A Message: ERROR-DSR=0 AFTER SELECT

Condition: The Data Set Ready (printer ready), CRU input bit E, was not set to a 1 after select on line printer was depressed.

B Message: ERROR-DSR=1 AFTER POWER LOSS

Condition: Data Set Ready (printer ready), CRU input bit E, was not reset to 0 after a power loss.

C Message: ERROR-TRANSMITTING DID NOT SET AFTER LDCR

Condition: Transmitting, CRU input bit 8, did not set to 1 during transmission of a character to the line printer.

D Message: ERROR-WRQ DID NOT RESET

Condition: Write Request, CRU input bit B, is set to a 1. This indicates that the Transmit Shift Register on the Interface Module has finished sending a character to the printer and is ready to receive another. Write Request bit should be set to a 0 at this point indicating that it has not finished sending a character to the printer.

E Message: ERROR-XMITING DID NOT RESET

Condition: Transmitting, CRU input bit 8, is set to a 1 indicating that a character is being transmitted to the printer. At this point CRU input bit 8 should be a 0 since a character is not being transmitted.

F Message: ERROR-NO DSR INT AFTER CHARACTER

Condition: The Data Set Ready (printer ready), CRU input bit E, is set to a 0 indicating the printer is busy (i.e., de-delected,out-of-forms,printing,etc.) It should be set to a 1 indicating the printer is selected and ready to accept a character from the Interface.

10 Message: ERROR-DSR NOT=0 AFTER SENDING CHARACTER

Condition: The Data Set Ready (printer ready), CRU input bit E, is set to a 1 indicating the printer is selected and ready to accept characters from the Interface. At this point it should be set to a 0 which indicates that the printer is in a busy state and not able to accept characters (i.e., deselected,out-of-forms, printing,etc.)

11 Message: ERROR-DID NOT GET DSR INT AFTER PRINTING

Condition: A second Data Set Ready Interrupt did not occur after a line was printed.

12 Message: ERROR-DSR DID NOT SET AFTER PRINTING

Condition: Data Set Ready did not return to a 1 after printing a line.

13 Message: CHARACTER SENT=XX , Where XX is the hexadecimal value for the character just sent.

Condition: This message serves as a preface to succeeding error messages to designate the character transmitted when the error was detected.

7.0 PART NUMBERS

PROGRAM DESCRIPTION	2250123-9901 (PD)
	-2001 (ROFF SRC)
FICHE KIT	-0009 (SP)
LPTEST, LINKED TEST	-1006 (FLO)
	-7006 (LC)
	-9006 (LML)
LPTEST, TEST MODULE	-1003 (OBJ)
	-2003 (SRC)
	-9003 (LIST)
LPMSG, MESSAGE MODULE	2250263-1003 (OBJ)
	-2003 (SRC)
	-9003 (LIST)

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7047				

REV
SHEET

REV STATUS OF SHEETS | REV | SHEET

UNLESS OTHERWISE SPECIFIED
DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± .010
2 PLACE DECIMAL ± .02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL STD 100

DWN | DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

PD, MAP12, MAPPING DIAGNOSTIC-990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	2268218-9901
SCALE	REV. A	SHEET 1 of 43

TI-0023 C

TABLE OF CONTENTS

1.0 SCOPE

This document describes the 990/12 map diagnostic program, MAP12. The map diagnostic is used to verify that the map logic in the 990/12 computer is functioning according to specifications. The test also provides isolation capabilities through the use of a hybrid combination of machine language code and microcode.

1.1 TESTING GOALS

MAP12 is designed to meet the following testing goals:

a) Field fault isolation to the board level. MAP12 would attempt to differentiate between functional failures caused by logic failures on the SMI12 board from functional failures caused by errors in external logic. This may not be applicable to the AU12 board since MAP12 assumes a functioning AU baord (i.e. The AU12 diagnostic has been run without error).

b) The code written for the MAP12 test module willl operate in the BLUE-M ovens, the GREY ovens, Hot Mock Up, and in the field.

c) MAP12 will output as much fault isolation information as is reasonably possible. This information will not only include functional failure data but also information related to the physical

location of failure occurence.

d) MAP12 will use a "small to big" testing philosophy. This approach starts by assuming as little as possible about the functional status of the board. The diagnostic uses this logic kernal to test additional logic sections and in turn uses these to continue testing in a building block fashion. The assumptions made by MAP12 are listed below.

1) Power-up microdiagnostic has run without error.

2) ROM ALC selftest has run without error.

3) AU12 diagnostic has run without error.

1.2 GENERAL SOFTWARE DESCRIPTION

MAP12 is a DOCS type test. It is also a hybrid diagnostic consisting primarily of machine language code with some use made of microcode for obtaining data not available at the machine language level.

2.0 REFERENCES

PART NUMBER	TITLE
2250077-9701	Model 990/12 Computer Assembly Language Programmer's Guide
0945400-9701	Model 990 Computer Diagnostic Handbook
	Diagnostic Standards
	Model 990/12 Specification
	Logic Diagrams AU12
	Logic Diagrams SMI12

3.0 EQUIPMENT AND SOFTWARE REQUIREMENTS

This section describes the minimum equipment requirements and the available linked object software for the diagnositc.

3.1 EQUIPMENT REQUIREMENTS

Refer to the Diagnostic Handbook for a discussion of the equipment requirements for DOCS. In addition to the equipment specified in the Diagnostic Handbook, the following equipment is required: IN10;990/12 Computer with 16K words of memory

3.2 SOFTWARE REQUIREMENTS

This section lists the linked object modules available in the MAP12 release. Refer to 7.0 for a list of all part numbers.

Loadable test module: MAP12, 2268218-1006 (FLO)

Linkable parts: MAP12M

MAP121

MAP122

MAP123

MAP124

MAP125

MAP126

MAP127

MAP128

MAP129

MAP12B

MAP12C

MAPSB

MAPSM

MAPDT

MAP12S

MAP12E

MAP12H

MICROD

4.0 LOADING

Refer to the Diagnostic Handbook for information on loading from all available media.

5.0 TEST EXECUTION AND DESCRIPTION

This is a DOCS type test and reference should be made to the Diagnostic Handbook for information on DOCS start-up and initialization.

5.1 INITIALIZATION

After the test module is loaded into memory, the test will output the following messge on the interactive device:

MAP12 990/12 MAP LOGIC TEST VERSION MM/YY *X

Where MM/YY *X is the date and revision level that the test was last released. Following this the test will ask the following initialization question:

SHOULD ERROR TRACE MEMORY BE DUMPED ON AN ERROR CONDITION? DEF=0000

The user should respond to this with a 0 for no or a 1 for yes.

5.2 EXECUTION

After the initialization questions have been answered and the operator has directed the test to start executing ("EXECUTE?-1" or "VERB?-EA") The messages shown below will be output if MAP12 runs without error.

```
E1-LDD LDS AS PRIV INST'S-1 SEC.
COMPLETE E1

E2-INTERRUPT AFTER LDS LDD-1 SEC.
COMPLETE E2

E3-XOP AND MAP STATUS BIT-1 SEC.
COMPLETE E3

E4-MAP INTERRUPT TEST-1 SEC.
COMPLETE E4

E5-SLSP INSTRUCTION TEST-1 SEC.
COMPLETE E5

E6-EXECUTE ENABLE TEST-1 SEC.
COMPLETE E6

E7-WRITE ENABLE TEST-1 SEC.
COMPLETE E7

E8-TILINE AND LONG DISTANCE MAP FILE-2 SEC.
COMPLETE E8

E9-ADDRESS DEVELOPMENT TEST-30 SEC.
COMPLETE E9

EB-LOGIC LIMIT TEST-30 SEC.
COMPLETE EB

EC-INSTRUCTION TEST-1 SEC.
COMPLETE EC
```

When MAP12 is executed in a loop ("VERB?-LA" or front panel version), the loop count will be output to the front panel.

5.3 TEST DESCRIPTION

MAP12 is designed to test all of the map logic on the SMI12 board and some mapping related circuitry on the AU12 board. MAP12 consists of the subtests listed

below. These should normally be executed in the order shown.

SUBTEST	DESCRIPTION
E1	LDD and LDS as privileged instructions
E2	Interrupt after LDD, LDS string
E3	XOP and map status bits
E4	Map interrupt test
E5	SLSP instruction test
E6	Execute enable test
E7	Write enable test
E8	TILINE and long distance map file test
E9	Address development test
EB	Logic limit test
EC	Instruction test

The correct operation of the AU12 board, the Writable Control Store, the Error Trace Memory Logic, and the TILINE memory are required in order to get meaningful results from the MAP12 test.

5.3.1 E1/L1 VERB -- LDD and LDS As Priviledged Instructions

This test verifies that the LDD and LDS instructions cause a privileged operation interrupt when they are executed with the privileged bit set. The Trace Memory status bit is also checked for correct operation.

5.3.2 E2/L2 VERB -- Interrupt After LDS and LDD

This part verifies that an interrupt will not occur until after a string of long distance instructions is complete. This is done by executing an LDS with an operand of >FBFE (this causes a TILINE timeout) followed by three LDD instructions. A check is then made to verify that the TILINE timeout interrupt did not occur until after the LDD instruction. The Trace Memory status bit is also checked for correct operation.

5.3.3 E3/L3 VERB -- XOP and Map Status Bit

A check is made to verify that when an XOP instruction is executed it clears the privileged and map bits in the status register. A software type XOP is used (ST11 cleared).

5.3.4 E4/L4 VERB -- Map Interrupt Test

This test is the first to actually enable mapping. It uses map file 2 (LDS) with all the limit registers equal to >FFF0. This should cause the map error. After the interrupt occurs, several checks are performed to verify that the interrupt occured correctly. This includes checking the Trace Memory status bit. Also, a check is performed to verify that a RSET, setting and resetting bit 4 at CRU Base >1FA0, and a SBZ of bit 11 at CRU Base >1FC0 will clear the map interrupt.

5.3.5 E5/L5 VERB -- SLSP Instruction Test

This test verifies the correct operation of the SLSP instruction. Since the SLSP instruction uses the same microcode as SLSL AU12 checks most of the operation. This test checks the mapping routine which is not checked in AU12. This test follows the same testing procedure as AU12.

5.3.6 E6/L6 VERB -- Execute Enable Test

This part checks the Execute Enable bit in Mapping Limit Register one. It also checks the corresponding enable flag in the status register as well as the interrupts, interrupt status bits, and the Trace Memory status bits associated with this enable. It uses map file 1 to cause the error.

5.3.7 E7/L7 VERB -- Write Enable Test

This test checks the write enable bit in mapping limit register one and also checks the protection enable bit in the status register, the level 2 interrupt status register, and the Error Trace Memory write violation bit. It uses map file 2 (LDD) to cause the error. An LDS instruction is tried first to show that the error does not occur during a read.

5.3.8 E8/L8 VERB -- TILINE And Long Distance Map File Test

The TILINE peripheral area and the ROM areas are defined to be at AU addresses >F800 through >FFFE when map file 0 is being used or the map is turned off.

If map files 1 or 2 are being used these addresses go to the last 2K of the first 32K of memory.

To verify that map file 0 can address the TILINE peripheral and ROM areas the data at addrss >FFFC is moved into a register. Then the map is turned on and another move is done using map file 0. The data read both times should be the same. Next, the same location is written to and read using map files 1 and 2. The data patterns used are 0 through >FFFF.

If a TILINE timeout occurs while doing the write, the test assumes the machine does not have memory at the location >FFFC, aborts this test, and goes to the next test.

An LDS instruction is executed with its address in the ROM area. The ROM address area is from >FC00 to >FFFE. The data that was loaded into map file 2 is then compared to the data in the ROM area and the data should be the same. This verifies that the map file 2 data can be loaded from the system ROM. The map is not turned on so this also shows that the long distance instructions will

load with the map off.

5.3.9 E9/L9 VERB -- Address Development Test

This test verifies that the TILINE address development logic operates correctly. This includes the base register map file, adder, and latch circuits. This test uses microcode in the Writeable Control Store to input address patterns to the adder circuits, to force a map cycle, and to select a specific base + address result into the Trace Memory. The test can then read the Trace Memory and compare with expected data. The test does not generate TILINE cycles for the mapped address and will maintain TILINE memory integrity. Error Trace Memory and the breakpoint system must function correctly for this test to pass.

5.3.10 EB/LB VERB -- Logic Limit Test

This test checks for the correct operation of the base selection logic. This includes the limit register map file, adder, and base selection circuits.

The limit hardware works by adding together bits 0 - 10 of the limit register and bits 0 - 10 of the AU address. If a carry is developed, then the corresponding base is not selected. If all three limits generate a carry, this is defined as a map error and the map error interrupt

line is set.

Again, microcode will be used to input address patterns to the adder circuits and to force a mapping cycle. The test will then verify the results by reading the Trace Memory and comparing with expected data. The test does not generate TILINE cycles for the mapped address and will maintain TILINE memory integrity.

5.3.11 EC/LC VERB -- Instruction Test

This test verifies that all instructions with a source or destination operand will work correctly when they are preceeded by a long distance instruction. This verifies the map logic on the SMI12 board and the microcode that controls the map logic.

The instructions should work as follows when preceded by a LDS or LDD instruction:

Branch Instructions -- a LDD or LDS will not affect the execution of a B, BL, or BLWP instruction.

Immediate Instructions -- A long distance instruction should have no effect on immediate instructions.

Instructions with TS or TD field = 0 -- A long distance instruction should have no effect.

Instructions with TS or TD field not = 0 -- If an

instruction with a source address with TS non zero is preceeded by a LDS the source operand should be fetched using map file 2. If the instruction is preceeded by a LDD, the LDD should not affect the source fetch. If an instruction with a destination address with TD non zero is preceeded by a LDD, the destination operand is stored using map file 2. It the instrucion is preceeded by an LDS, the LDS should not affect the destination store.

XOP -- The software is not affected by a long distance instruction.

5.3.12 OTHER MAP12 VERBS

MAP12 also contains the verbs SB, CB, SM, and ET. The verbs SB and CB are used to set and clear breakpoints respectively. They can be used to manually test the hardware breakpoint feature of the 990/12. All of these verbs have a RSET instruction at the beginning which clears breakpoints. For this reason, breakpoints cannot be used within a test verb. The SM verb is used to stimulate mapping. Using this verb allows the operator to enter any base and limit values he wishes and the computer can be made to loop on this data. This allows manual verification of mapping hardware with an oscilloscope.

The ET (Expand Trace) verb is used to help understand the Error Trace Memory buffer information. If executed after a test fails, it will format the last Error Trace Memory buffer in a more readable form. If the listings are not available then any value may be entered when asked 'ENTER MODULE LOAD BIAS'. The folllowing abbreviations are used in the printout:

TIL -- The 20 bit TILINE word address.

STAT -- The merged status bits as read from Error Trace Memory. These are broken out bit by bit in the right 12 columns.

MEM -- This is the 16 bit CPU byte address represented by the TILINE address.

LIST -- This is the listing address. The module load bias is subtracted from the MEM address to get this value.

WV -- Write violation (1=error, 0=no error).

EV -- Execute violation (1=error, 0=no error).

TO -- TILINE timeout.

PE -- Memory parity error.

ME -- Mapping error.

IL -- Illegal opcode.

PV -- Privileged violation.

TIL R/W -- 0=TILINE read, 1=TILINE write.

TIL -- 0=no TILINE access, 1=TILINE access.

CA R/W -- 0=Cache read, 1=Cache write

CA -- 0=no Cache access, 1=Cache access

EOI -- 0=no end of instruction, 1=end of instruction

6.0 MESSAGES

This section describes the messages produced by MAP12. There are two types of messages, Header messages and Error messages. Each type is discussed in its own section following. All messages are output through DOCS service routines on the selected header and error message device(s).

6.1 HEADER MESSAGES

Header messages are output to indicate the progress of the test and are for general information only. Most of the header messages are listed in section 5.2.

6.2 ERROR MESSAGES

Error messages which appear in MAP12 take the following form:

EXY-ERROR MESSAGE TEXT

Where X is the test number and Y is the error number within that test. The error messages with explanations follow.

E11-LDS DID NOT CAUSE PRIV INT

The LDS instruction was performed while in non-privileged mode and a level 2 privileged violation interrupt was expected but none occurred.

E12-LDD DID NOT CAUSE PRIV INT

Same as E11 except that the LDD instruction was

performed.

E13-LDS OR LDD CAUSED NON-PRIV INT

The LDD or LDS instruction was performed in non-privileged mode. A level 2 interrupt occurred but of the wrong type.

E14-TRACE MEM PRIV BIT NOT SET

The LDD and LDS instruction was performed in non-privileged mode and the proper interrupt occurred. However, the privileged violation bit in Error Trace Memory did not set to one. This is a problem with Error Trace Memory so the AU board is suspected.

E15-MULTIPLE INT'S CAUSED

The interrupt occurred correctly for the LDS instruction but was not cleared causing it to interrupt more than once.

E16-INCORRECT STATUS-SHOULD=>0182

After the privileged violation is caused in verb E1, the return status in the level 2 interrupt handler is tested. This indicates that the interrupt did not cause the return status to be saved or that the status was set incorrectly before the interrupt was caused. This error suggests a faulty AU board.

E21-NO INT AFTER LONG DIST. STRING

A TILINE timeout was caused at the end of a string of LDD and LDS instructions. The TILINE timeout was supposed to stay pending, but it never occurred.

E22-NON-TILINE TIMEOUT INT CAUSED

A level 2 interrupt occurred in test E2 but it was not a TILINE timeout.

E23-INT TRAPPED FROM WRONG PC

A TILINE timeout occurred in test E2 but the return PC indicates that it occurred at the wrong time.

E24-MULTIPLE INT'S CAUSED OR INT OCCURRED IN STRING

The TILINE timeout occurred at the right place in test E2 but after it was cleared it occurred again.

E31-XOP DID NOT CAUSE INT TRAP

A software XOP was attempted but it did not trap to the XOP handler routine.

E32-NEW STATUS BAD; SHOULD=>0202

The software XOP trapped to the right place. The status register was preloaded with >0182. The map file and privileged bits should have been cleared. The XOP in progress bit should have been set to one.

E33-OLD STATUS (R15) BAD; EXP=>0182

The XOP trapped to the right place and the new status was correct but the return status did not contain the preloaded value.

E34-R11 LINK BAD; SHOULD BE=WP PTR

The XOP instruction was supposed to put the address of the source operand into register R11 in the new context. The instruction tested was XOP R0,0. So R11 should contain the address of R0 which is the same as the address contained in the workspace pointer.

E35-BAD XOP CAUSED; XOP IN R10

An incorrect XOP level occurred. The level value is in register R10.

E41-NO MAP ERR CAUSED

Test E4 causes a mapping error intentionally using map file 2. This error indicates that the map error level 2 interrrupt did not occur.

E42-L2 MAP ERR BIT NOT SET

A level 2 interrupt occurred in test E4 but it was the wrong type of level 2 interrupt (not a map error).

E43-TRACE MEM MAP ERR BIT NOT SET

The map error interrupt occurred correctly but the map error bit in Error Trace Memory was not turned on.

E44-MULTIPLE INT'S CAUSED

The map error occurred correctly in test E4 but was not cleared. This caused a multiple interrupt to occur.

E45-MAP ERR DID NOT TERM INST EXEC

The map error was supposed to occur during a long distance source (LDS) on a MOV instruction. The map error was supposed to terminate execution of the MOV instruction but the MOV did occur.

E46-RSET DID NOT RESET MAP ERR BIT

A RSET instruction was used to clear the level 2 map error interrupt but it did not clear the level 2 error status register.

E47-SBZ DID NOT RESET MAP ERR BIT

A SBZ instruction to the map error bit in the error status register was used to clear the levle 2 map error interrupt but failed to clear it.

E48-MAP ERR NOT RESET BY SBZ 4

A SBZ to bit 4 at CRU base >1FA0 was used to clear the level 2 map error interrupt but failed to do so.

E49-MAP ERR NOT RESET BY SBO 4

Same as E48 only a SBO instruction was used.

E51-SLSP FINAL BLOCK ADDRESS INCORRECT

<ACTUAL ADDR IN R7, EXPECTED ADDR IN R8>

E52-LDS SLSP FINAL BLOCK ADDRESS INCORRECT

<ACTUAL ADDR IN R7, EXPECTED ADDR IN R8>

E53-LDD SLSP FINAL BLOCK ADDRESS INCORRECT

<ACTUAL ADDR IN R7, EXPECTED ADDR IN R8>

E54-SLSP NEXT-TO-LAST BLOCK ADDRESS INCORRECT

<ACTUAL ADDR IN R9, EXPECTED ADDR IN R10>

E55-LDS SLSP NEXT-TO-LAST BLOCK ADDRESS INCORRECT

<ACTUAL ADDR IN R9, EXPECTED ADDR IN R10>

E56-LDD SLSP NEXT-TO-LAST BLOCK ADDRESS INCORRECT

<ACTUAL ADDR IN R9, EXPECTED ADDR IN R10>

E57-BREAKPOINT DID NOT OCCUR DURING ATTEMPT TO REEXECUTE LDD SLSP FROM A BREAKPOINT INTERRUPT

E58-EXP RETURN PC ERROR WHEN LDD SLSP REEXECUTED FROM A BREAKPOINT

E61-NO EXEC VIOLATION OCCURRED

The memory protection bits for map file 1 were set up to cause an execution violation but no level 2 execution violation interrupt occurred.

E62-L2 EXEC VIOLATION MISCOMPARE

A level 2 interrupt occurred in test E6 but it was the wrong type interrupt (not an execution violation).

E63-TRACE M. EXEC VIOL MISCOMPARE

A level 2 execution violation interrupt occurred correctly in test E6 but the Error Trace Memory execution violation bit was not set.

E64-EXEC VIOL DID NOT INHIBIT WRITE

The level 2 execution violation interrupt in teset E6 occurred correctly, but it did not inhibit the write in the MOV instrucion where the violation was caused.

E65-MULTIPLE INT'S CAUSED

The execution violation interrupt occurred correctly but it was not cleared so it interrupted more than once.

E71-NO WRITE VIOLATION ERR CAUSED

A write violation error is force by using the protection bits in map file 2 and the LDD instruction. This error indicates that no level 2 interrupt occurred.

E72-L2 WRITE VIOLATION MISCOMPARE

A level 2 intrrupt occurred in test E7 but it was the wrong type (not a write violation).

E73-TRACE M. WRITE VIOL MISCOMPARE

A write violation level 2 interrupt occurred correctly in

test E7 but the write violation bit in Error Trace Memory did not turn on.

E74-NO WRITE INHIBIT ON WRITE VIOL

The write violatiin interrupt occurred correctly but the violation did not inhibit the write in the MOV instruction where the error was caused.

E75-MULTIPLE INT'S CAUSED

The write violation level 2 interrupt occurred normally but it was not cleared, causing multiple interrupts.

E76-INT OCCURRED AT WRONG PLACE

A level 2 interrupt occurred in test E7 but the return program counter was incorrect. This error may occur when the real problem is that the wrong type level 2 interrupt occurred (refer to error E72).

E81-UNMAPPED MF 0 ROM READ DIFFER **UNMAPPED IN R7--MF 0 IN R0**

Address >FFFC is read in unmapped mode and using map file zero. The same TILINE address should be accessed on both reads. Registers R7 and R0 contain the data that was read on the respective reads, showing that different TILINE addresses were accessed.

E82-MF 1 2 RAM READ DIFFER **MF 1 IN R7--MF 2 IN R0**

Addresses >0000 through >FFFF are read using map files 1 and 2. The same data should be accessed on each pair of reads since the same TILINE addresses should be generated from teh different map files. Registers R7 and R0 contain the data when a miscompare occurs.

E83-UNEXPECTED INT OCCURRED

TILINE timeouts are expected during the 'map file 1 and 2 RAM read' portion of E8. Before this section strts, any level 2 interrupt that occurs is considered an error. Any interrupt other than a TILINE timeout is an error at any time during test E8.

E84-EXPECTED INT NOT TILINE TIMEOUT

A level 2 interrupt occurred during the map file 1 and 2 RAM read portion of test E8 but it was not a TILINE timeout.

E91-BKPT DID NOT OCCUR ON INPUT ADD **ADDR IN R6**

The writeable control store routine used in test E9 only return to assembly language if a breakpoint is present. For this error to ever occur, the breakpoint had to exist at that time. This means that the breakpoint must have been spuriously reset.

E92-UNEXP L2 ERR'S ON MCODE PASS **L2 STAT IN R2**

This error appears when more than one level 2 interrupt

occurs in test E9. One of teh level 2 interrupts will be the breakpoint or else error E91 would have appeared instead. The error status register contains >0040 when a breakpoint occurs.

E93-EXP(R0-R1) ACT(R10-R11) ADDR'S DO NOT COMPARE
ADDR IN R6--BASE IN R7--BASE IN R3
The computed TILINE address in registers R0 and R1 is not the same as the TILINE address fetched from second entry in Error Trace Memory. The map logic did not compute the address correctly.

EB1-BKPT DID NOT OCCUR ON INPT ADD **ADDR IN R6**
Refer to the description for error E91.

EB2-UNEXP L2 ERR'S ON MCODE PASS **L2 STAT IN R2**
Refer to the description for error E92.

EB3-EXP'D INT NOT MAP OVFL **L2 STAT IN R2**
The test was supposed to generate a map error but no map error occurred.

EB4-NO EXP'D MAP OVFL ERR **L2 STAT IN R2**
The test was supposed to generate a map overflow error but did not.

EB5-EXP(R0-R1) ACT(R10-R11) ADDR'S DO NOT COMPARE
ADDR IN R6--BASE IN R12--LIMIT IN R7

The map logic computed an incorrect address. The actual address generated by the mapping hardware is in registers R10 and R11. The computed expected value is in registers R0 and R1. The base address at the time the error occurred is in register R12 and the limit value that should have been selected is in register R7. Register R6 contains the CPU address that was mapped.

EC1-UNEXPECTED RESULT IN R0

Test EC uses registers R0, R1, and R2 to test many different instructions preceeded by long distance instructions. These registers are preloaded and their contents are checked after execution. This error indicates that the long distance execution of that instruction failed.

EC2-UNEXPECTED RESULT IN R1

Refer to the description of error EC1.

EC3-UNEXPECTED RESULT IN R2

Refer to the description of error EC1.

EC4-UNEXPECTED RESULT AT SRC

The unmapped source address is checked to make sure it is correct after the long distance execution.

EC5-UNEXPECTED RESULT AT DEST

The unmapped destination address is checked to make sure it contains the correct data after long distance execution of an instruction. Note that if the LDD instruction was used the unmapped destination address should never have been generated. If the target instruction was preceeded by an LDS instruction, then the unmapped destination address should be used. This error indicates that the unmapped destination address contained incorrect data.

EC6-UNEXPECTED RESULT AT MSRC

The mapped source address is checked even if the target instruction was preceeded aby an LDD instruction (which would cause the unmapped source and the mapped destination addresses to be used). This error indicates that the mapped source address contains the wrong data.

EC7-UNEXPECTED RESULT AT MDEST

The mapped destination address contains incorrect data after long distance execution of a target instruction.

7.0 PART NUMBERS

TITLE	NUMBER	
PROGRAM DESCRIPTION	2268218-2001	ROFF Source
	-9001	ROFF Output
	-9901	PD Document
FICHE KIT	-0009	SP
MAP12 - Linked Test	-1006	FLO Fully Linked Object
	-7006	LC Link Control
	-9006	LML Link Map List
MAP12M - Main Driver	-1003	OBJ
	-2003	SRC
	-9003	LIST
MAP121 - Test 1	2268219-1003	OBJ
	-2003	SRC
	-9003	LIST
MAP122 - Test 2	2268220-1003	OBJ
	-2003	SRC
	-9003	LIST
MAP123 - Test 3	2268221-1003	OBJ
	-2003	SRC
	-9003	LIST

MAP124 - Test 4	2268222-1003	OBJ
	-2003	SRC
	-9003	LIST
MAP125 - Test 5	2268223-1003	OBJ
	-2003	SRC
	-9003	LIST
MAP126 - Test 6	2268224-1003	OBJ
	-2003	SRC
	-9003	LIST
MAP127 - Test 7	2268225-1003	OBJ
	-2003	SRC
	-9003	LIST
MAP128 - Test 8	2268226-1003	OBJ
	-2003	SRC
	-9003	LIST
MAP129 - Test 9	2268227-1003	OBJ
	-2003	SRC
	-9003	LIST
MAP12B - Test B	2268228-1003	OBJ
	-2003	SRC
	-9003	LIST

MAP12C - Test C	2268229-1003	OBJ
	-2003	SRC
	-9003	LIST
MAPSB - Set and Clear	2268230-1003	OBJ
	-2003	SRC
	-9003	LIST
MAPSM - Stimulate Mapper	2268231-1003	OBJ
	-2003	SRC
	-9003	LIST
MAPDT - Dump Trace	2268232-1003	OBJ
	-2003	SRC
	-9003	LIST
MAP12S - I/O Service Routines	2268233-1003	OBJ
	-2003	SRC
	-9003	LIST
MAP12E - Error Messages	2268234-1003	OBJ
	-2003	SRC
	-9003	LIST
MAP12H - Header Messages	2268235-1003	OBJ
	-2003	SRC
	-9003	LIST

MICROD - Microcode		2268236-1003	OBJ
	Overlay	-2003	SRC
		-9003	LIST
MAP12MAC - Macro File		2268485-2003	SRC

8.0 ALGORITHMS

This section contains all algorithms for MAP12 in Metacode. It is hoped that this will be helpful in future debugging and updates.

```
************
Begin
Perform MAP12
End
************
Procedure MAP12                    ;990/12 MAP DIAGNOSTIC
        Begin
            Get VERB response
            If response EQ IT then Perform IT    **IT
            If response EQ EA then Do
                Perform TEST1
                Perform TEST2
                Perform TEST3
                Perform TEST4
                Perform TEST5
                Perform TEST6
                Perform TEST7
                Perform TEST8
                Perform TEST9
                Perform TESTB
                Perform TESTC
            End
            If response EQ LA then
              Until interrupted (by operator) Do
               Perform TEST1
               Perform TEST2
               Perform TEST3
               Perform TEST4
               Perform TEST5
               Perform TEST6
               Perform TEST7
               Perform TEST8
               Perform TEST9
               Perform TESTB
               Perform TESTC
               Increment loop count
               Output loop count to front panel
            End
            If response EQ E1 then perform TEST1     **EX1
            If response EQ E2 then perform TEST2     **EX2
            If response EQ E3 then perform TEST3     **EX3
```

```
            If response EQ E4 then perform TEST4     **EX4
            If response EQ E5 then perform TEST5     **EX5
            If response EQ E6 then perform TEST6     **EX6
            If response EQ E7 then perform TEST7     **EX7
            If response EQ E8 then perform TEST8     **EX8
            If response EQ E9 then perform TEST9     **EX9
            If response EQ EB then perform TESTB     **EXB
            If response EQ EC then perform TESTC     **EXC
            If response EQ SB then perform SETBP     **SBX
            If response EQ CB then perform CLRBP     **CBX
            If response EQ SM then perform STMAP     **SMX
            If response EQ ET then perform EXTRC     **ETX
            If response EQ L1 then
              Until interrupted by operator Perform TEST1
            If response EQ L2 then
              Until interrupted by operator Perform TEST2
            If response EQ L3 then
              Until interrupted by operator Perform TEST3
            If response EQ L4 then
              Until interrupted by operator Perform TEST4
            If response EQ L5 then
              Until interrupted by operator Perform TEST5
            If response EQ L6 then
              Until interrupted by operator Perform TEST6
            If response EQ L7 then
              Until interrupted by operator Perform TEST7
            If response EQ L8 then
              Until interrupted by operator Perform TEST8
            If response EQ L9 then
              Until interrupted by operator Perform TEST9
            If response EQ LB then
              Until interrupted by operator Perform TESTB
          If response EQ LC then
            Until interrupted by operator Perform TESTC
        End
*************
```

```
************
Procedure TEST1 **E1X     ;LDD and LDS Privileged Mode
        Begin
            Turn off map
            Restrict Interrupts
            Set PRIV bit in status (non-privileged mode)
            Set level 2 trap
            Test LDS    **E1XINT
              should cause L2 interrupt
            Test LDD    **E1XINT
              should cause L2 interrupt
        End
************
Procedure TEST2 **E2X     ;Interrupt after LDD and LDS
        Begin
            Turn map off
            Restrict interrupts
            Set level 2 trap
            Execute LDD-LDS string
              Cause interrupt on first one
            Verify that interrupt was not serviced until
              entire string was executed
        End
************
Procedure TEST3  **E3X    ;XOP and map status bit check
        Begin
            Turn off map
            Restrict interrupts
            Set up XOP traps
            Execute XOP 0
            Check for correct operation
        End
************
Procedure TEST4  **E4X    ;Map interrrupt test
        Begin
            Set map files 0 and 1 to zeros
            Set level 2 interrupt trap
            Execute MOV using MF 2 which causes map error
            Check for correct operation
            Check other reset method.
        End
************
```

```
************
Procedure TEST5  **E5X   ;SLSP instruction test
        Begin
            Enable mapping
            Set up trap
            Execute SLSP and check results
            Execute SLSP preceeded by LDD
            Execute SLSP preceeded by LDS
            Reexecute from a breakpoin
        End
************
Procedure TEST6  **E6    ;Execute enable memory management test
        Begin
            Set up map files 0 and 1
            Set level 2 trap
            Enable memory management
            Execute MOV using MF 2 causing EXEC VOILATION
            Check L2 status
            Check Trace memory status
        End
************
Procedure TEST7  **E7     ;Write enable test
        Begin
            Set up map files 0 and 1
            Set level 2 trap
            Enable memory management
            Execute MOV using map file 2
              cause write violation interrupt
            Check L2 and memory status
        End
************
```

```
************
Procedure TEST8  **E8X   ;TILINE and LDD/LDS test
        Begin
            Read from >FFFC unmapped
            Read from >FFFC with map file 0
            If data is different then ERROR
            Else clear value at >FFFC with map file 1
            While value at >FFFC is not zero Do
                Read from >FFFC with map 1
                Read from >FFFC with map 2
                If data is different then ERROR
                Else increment value at >FFFC by 2
            End
        End
************
```

```
************
Procedure  TEST9 **E9X   ;Address development test
        For Map file = 0 and 1 Do
            Load MICROCODE
            Until BASE.REG. EQ 4 Do
                Set limit register
                Until ADDRESS.BASE.PAIRS EQ END Do
                     Get address/base pair
                     Put base address in map 0
                     Load map 0
                     Put address in MICROCODE
                     Load into WCS
                     Set breakpoint for address
                     XOP into WCS
                     Read Trace Memory
                     Calculate expected TILINE address
                     If EXP NE ACT then ERROR
                     Calculate next base address
                     Calculate next AU address
                 End
                 Increment BASE.REG.
            End
        End
************
```

```
************
Procedure TESTB  **EBX   ;Limit logic test
        For Map file = 0 and 1 Do
            Load MICROCODE
            Until LIMIT.REG. EQ 4 Do
                Set limit register
                Until ADDRESS.BASE.PAIRS EQ END Do
                      Get address/base pair
                      Put limit value in map 0
                      Load map 0
                      Put address in MICROCODE
                      Load into WCS
                      Set breakpoint for address
                      XOP into WCS
                      Read Trace Memory
                      Calculate expected TILINE address
                      If EXP NE ACT then ERROR
                      Calculate next base address
                      Calculate next AU address
                 End
                 Increment BASE.REG.
            End
        End
************
Procedure TESTC   **ECX  ;Long distance instruction test
        For LDD and LDS Do
            While INSTRUCTION.TABLE.POINTER NE END Do
                Initialize target instruction area with NOP
                Set up target instruction area
                Set up registers
                Set up memory locations
                Execute target instruction
                If ACT NE EXP then ERROR
                Increment INSTRUCTION.TABLE.POINTER
            End
        End
************
```

```
*************
Procedure SETBP  **SBX   ;Set breakpoint utility
        Begin
            If BREAKPOINT.ACTIVE.FLAG set then Perform CLRBP
            Get breakpoint address
            Get breakpoint type
            Disable interrupts
            Set BREAKPOINT.ACTIVE.FLAG
            Set up L2 trap
            Set breakpoint type
            Set breakpoint register address
            Enable breakpoint
            Enable interrupts
        End
*************
Procedure CLRBP  **CBX   ;Clear breakpoint utility
        Begin
            If BREAKPOINT.ACTIVE.FLAG reset then END
            Disarm breakpoint
            Restore L2 trap
            Reset BREAKPOINT.ACTIVE.FLAG
        End
*************
```

```
************
Procedure STMAP  **SMX   ;Stimulate mapper utility
        Begin
            Get map file value
            Get limit and base values
            Get AU address
            Get LOOP.OPTION
            Get L2INTS.OPTION
            Load MICROCODE
            Load map file
            Put AU address in MICROCODE
            Load into WCS
            Disable interrupts
            If LOOP.OPTION EQ NO then set breakpoint
            XOP into WCS
            Apply AU address to ABUS
            Force map cycle
            Transfer mapped address to Trace memory
            If pending interrupts then EOI
            Else loop
            If error condition then END
            If no breakpoint interrupt then  enable interrupts
            Else Begin
                 Output mapped address
                 Clear breakpoint
            End
        End
************
Procedure EXTRC    **ETX ;Expand trace utility
        Begin
            Read and output L2 status
            Format and output Trace Buffer
        End
```

A

APPLICATION	
NEXT ASSY	USED ON
	7506

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

REV
SHEET

REV STATUS OF SHEETS — REV / SHEET

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL STD 100

DWN — DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group — *Dallas, Texas*

PD, MAPTST, MAP LOGIC TEST-990/10

SIZE	CODE IDENT NO	DRAWING NO
A	96214	2250561-9901

SCALE — REV. — SHEET 1 of 29

TI-0023-C

TABLE OF CONTENTS

1.0 SCOPE

This document describes the 990/10 Map Test. The Map Test is used to verify that the Map Logic in the TI 990/10 Minicomputer is working correctly.

NOTE: There are two versions of this Test.

If the 990 being tested has a 'Printed Circuit' AU Board, use the following part:

MAPTST	02250561-1006 (FLO)

For the office systems configuration, use the following part:

MAPTSTOS	02250561-1010 (FLO)

2.0 REFERENCES

For information beyond the scope of this document reference the followings:

PART NUMBER	TITLE
943441-9701	Model 990 Computer TMS 990 Microprocessor Assembly Language Programmer's Guide
945417-9701	Model 990/10 Computer Systems Hardware Reference Manual
945400-9701	Diagnostic Handbook
944932	Logic Diagram AU 1, 990/10
944952	Logic Diagram AU 2 with Map, 990/10

3.0 EQUIPMENT AND SOFTWARE REQUIREMENTS

3.1 EQUIPMENT

The following equipment is required to run this test. A 990/10 System with mapping option and appropriate loading device, interactive device.

3.2 SOFTWARE REQUIREMENTS

The following software modules should be used for the test.

MAPTST, STANDARD LINK		2250561-1006	(FLO)
LINKED PARTS:	MAPTSTPC	2250561-1003	(OBJ)
	MAPMSG	2250560-1003	(OBJ)
MAPTSTOS, OFFICE SYSTEM LINK		2250561-1010	(FLO)
LINKED PARTS:	MAPTSTO	2250561-1004	(OBJ)
	MAPMSG	2250560-1003	(OBJ)

4.0 LOADING

This test must be loaded into low memory (address >100 or less). The reason for this is that Subtest number 5 of the Test uses memory at >3000 to test the LMF instruction. Subtest 6 also uses memory at location >3000.

Refer to the Diagnostic Handbook (PB 945400-9701) for procedures on how to run 990 stand-alone diagnostics.

5.0 TEST EXECUTION AND DESCRIPTION

This test is a 'stand-alone' test.

Refer to the Diagnostic Handbook for a description of a stand-alone test and what options are available and how to select them.

5.1 TEST DESCRIPTION

This test is designed to test all of the map logic on the AU1 and AU2 boards. It also checks the micro code ROM bits that control the map logic.

This test is composed of 6 subtests. There is a main part of the test that calls each of the 6 subtests in order.

When each subtest starts executing, it outputs a starting message, and when it completes, it outputs a completion message.

When running the test, be sure to verify that each subtest beginning and completion message is output. If a subtest outputs a beginning message but no completion message, there is a problem in the computer.

The following is a description of each subtest and how it verifies the map logic.

SUBTEST	DESCRIPTION
1	Map File Bit Test
2	Map Interrupt Test
3	Address Development Test
4	Limit Logic Test

5	Latched Map Increment Test
6	Instruction Test

5.1.1 MAP FILE BIT TEST

Subtest number one verifies all of the map file registers. It does this by writing various patterns to the map files, then reading the data back and checking it. The patterns used are:

a) zero's

b) -1's

c) Walking ones

It writes the data to map files 0 and 1 by doing load map file instructions. It loads map file 2 by doing a LDS instruction. It reads the data out through the map CRU.

In order to verify that the addressing is working correctly when loading map files 0 and 1, this test first loads map file 0, then 1 and checks the data. It then goes back and does the test over, but loads map file 1 first.

Subtest 1 does not modify the interrupt vectors or the interrupt mask. The map is not turned on in subtest 1.

5.1.2 MAP INTERRUPT TEST

Subtest 2 checks several special conditions associated with the map status bit and the map interrupt.

The first part verifies that the LDD and LDS instructions cause a privilged operation interrupt when they are executed with the privileged bit set.

The second part verifies that an interrupt will not occur until after a string of long distance instructions is complete. The way this is done is an LDS with an operand of '>FBFE' is executed (this causes a TILINE timeout) followed by three LDD instructions. A check is then made to verify that the TILINE timeout interrupt did not occur until after the instruction after the last LDD instruction.

The third part verifies that when an XOP instruction is executed, it clears the privileged and map bits in the status register.

The fourth part verifies that a level 2 map interrupt will occur by turning the map on then doing an LDS followed by a MOV from a high address. The map file 2 is set up with all of the limit registers equal to FFFF, so this should cause a map error. Both map files 0 and 1 are loaded with all zeros. After the interrupt occurs several checks are performed to verify the interrupt occured correctly. Also a check is performed to verify that a RSET instruction and setting and resetting bit 4 of the map CRU will clear the map interrupt.

The fifth part verifies that map file 0 can address the TILINE peripheral and ROM areas defined to be at AU addresses F800 through FFFE. If map files 1 or 2 are being used these addresses go to the last 2K words of the first 32K words of memory. To verify that map file 0 can address the Tiline peripheral and ROM areas the data at address FFFC is moved into a register. Then the map is turned on and another move is done using map file 0. The data read both times should be the same. Next the same location is written to and read using map files 1 and 2. The data patterns used are 0 through FFFF. If a TILINE timeout occurs while doing the write the test assumes the machine does not have memory at the location FFFC and aborts this test and goes to the next test. An LDS instruction is executed with its address in the ROM area. The ROM address area is from FC00 to FFFE. The data that was loaded into map file 2 is then compared to the data in the ROM area and the data should be the same. This verifies that the map file 2 data can be loaded from the system ROM. The map is not turned on so this also shows that the long distance instructions will loaded with the map off.

5.1.3 ADDRESS DEVELOPMENT TEST

Subtest 3 verifies that the map adder circuitry is working

correctly. It is not necessary that memory exists for all addresses.

The map adder circuitry works as follows:

1. The base value is added to the AU memory address.
2. The correct sum (from B1 or B2 or B3) is gated into the 20 bit Latched Map Address Register (LMPA).
3. The same 20 bit address is sent to the TILINE as a memory or peripheral address.

This test works by setting up the base being tested in map file 2. Then an LDS instruction is executed followed by an MOV *R0,R6 where register 0 has the AU address being used.

The map latch bits are set up so the address developed is latched up in LMPA. The LMPA is then read and the address checked. If there is no memory at the address that was developed, a TILINE timeout will be enabled but subtest 3 does not enable the interrupts (the mask is = 1) so the interrupt does not occur.

The adder chips that are used to add the base and memory addresses are 4 bit chips. Therefore it is not necessary to add all combinations of AU addresses and base address. Instead only 4 bits are tested at a time. This reduces the number of adds from 1,048,567 (2**20) to 2048 (2**10) adds.

The 3 bases are tested one at a time. Base 1 first then

base 2 then base 3.

PATTERNS USED

TEST BITS 12 - 15

BASE VALUE	AU ADDRESS	LMPA
0	20	20
1	20	40
.	¦	.
.	¦	.
F	20	200
0	40	40
.	¦	.
.	¦	.
F	40	220
	.	
	.	
0	1E0	1E0
.	.	.
.	.	.
F	1E0	3C0

TEST BITS 8 - 11

0	200	200
10	200	400
.	¦	.
.	¦	.
F0	200	2000
0	400	400
.	¦	.
.	¦	.
F0	400	2200

DO TEST WITH CARRY INTO CHIP SET

BASE VALUE	AU ADDRESS	LMPA
F	20	200
1F	20	400
.	¦	.
.	¦	.
FF	20	2000
F	220	400

FF	220	2200
F	1E20	2000
FF	1E20	3E00

TEST BITS 4 - 7

0	2000	2000
100		4000
F00	2000	20000
0	4000	4000
F00	4000	22000
0	E000	E000
F00	E000	2C000

NOW TEST WITH CARRY INTO CHIP SET

BASE VALUE	AU ADDRESS	LMPA
FF	20	2000
1FF		4000
FFF	20	20000
FF	2020	4000
FFF	2020	22000
FF	E020	100000
FFF	E020	2E0000

TEST BITS 0 - 3

0	20	000020
1000	¦	020020
.	¦	.
.	¦	.
F000	20	1E0020

NOW TEST WITH CARRY INTO CHIP SET

FFF	20	20000
1FFF	¦	40000
.	¦	.
.	¦	.
FFFF	20	200000

5.1.4 LIMIT LOGIC TEST

Subtest 4 verifies that the limit adder circuits and the base selection circuitry is working correctly. The limit hardware works as follows.

Bits 0 - 10 of the limit register and bits 0 - 10 of the AU address are added together.

If a carry is developed then the corresponding base is not selected. If all 3 limits generate a carry this is defined as a map error and the map error interrupt line is set. The base register for the limit register being tested is set up to point to the test buffer. The other two base registers are set up to point to their respective buffers. Base 1 uses buffer 1, 2 uses 2 and 3 uses 3.

Then the test does a

```
LDS
SETO *R7
```

Where register 7 has the test address.

Therefore if the correct base is used the test buffer will have a -1 stored into it. If the wrong base is chosen then one of the other buffers will get the -1.

If limit register 3 is being tested and a carry is developed then a map error interrupt will occur. The level 2 interrupt routine is set up to expect this and will not report an error but if a map interrupt occurs when limit register 3 should not generate a carry, an error will be reported.

The following patterns are used to test the limit registers:

LIMIT REG. VALUE	MEMORY ADDRESS	BITS BEING TESTED
0	0	0 - 3
1000	0	
.	¦	(Note: if testing L2 or L3,
.	¦	start with 20)
F000	0	
0	1000	
.	¦	
.	¦	
F000	1000	
.		
.		
0	F000	
.	¦	
.	¦	
F000	F000	

000	800	0-3 with carry into chip
1800	¦	
.	¦	
.	¦	
F800	800	
800	1800	
.	¦	
.	¦	
F800	1800	
.		
.		
800	F800	
.	¦	
.	¦	
F800	F800	

LIMIT REG. VALUE	MEMORY ADDRESS	BITS BEING TESTED
F000	0 (or 20)	4 - 7
F100	⋮	
.	⋮	
.	⋮	
FF00	0	
F000	100 (or 120)	
.	⋮	
.	⋮	
FF00	100	
.		
.		
F000	F00 (or F20)	
.	⋮	
.	⋮	
FF00	F00	

F080	0080	4 - 7 with carry
F180	⋮	
.	⋮	
.	⋮	
FF80	0080	
.		
.		
F080	0F80	
.	⋮	
.	⋮	
FF80	0F80	

FF00	0 (or 20)	8 - 10
FF20	⋮	
.	⋮	
.	⋮	
FFE0	0	
FF00	20 (or 40)	
.	⋮	
.	⋮	
FFE0	20	
.		
.		
FF00	E0	
.	⋮	
.	⋮	
FFE0	E0	

5.1.5 LATCHED MAP INCREMENT TEST

There is some special hardware that is used to increment the LMPA register when doing LMF instruction. This subtest verifies that the increment logic works properly. This subtest uses all of the available memory past the end of the test and only does a thorough test of the increment logic if 1 million words of memory is available.

This test will output a message telling the operator how much memory is available on the machine. The operator should verify that the correct memory configuration is printed out.

The first part of subtest 5 builds a list of all of the available memory in the 990/10.

The second part of subtest 5 uses this list and does the following:

1. Starting at address 3000 it writes BAD0 to every available memory location to the end of memory.
2. The test is done on a 4K area of memory at a time except for the first block which is the area from address 3000 to 4000 which is a 2K block of memory.
3. The 4K area under test is initialized with the data 0 thru 1FFE. The first block of memory is set to 1000 thru 1FFE.
4. The actual test is a load map file 1 from each 6 word

area of the test area. After each load the map file is read and the data is compared with the data in memory.

5. The test area is then initialized to BAD0 and the next 4K area is initialized to 0 thru 1FFE.

6. The test is repeated until the load map file has been done from all available memory.

5.1.6 INSTRUCTION TEST

Subtest 6 verifies that all instructions with a source or destination operand will work correctly when they are preceded by a long distance instruction. This verifies the map logic on the AU1 board and the micro code that controls the map logic.

The instructions should work as follows when preceded by a LDS or LDD instruction.

Branch instructions -- a LDD or LDS will not affect the execution of a B, BL or BLWP instruction.

Immediate Instructions -- a long distance instruction should have no affect on immediate instructions.

Instructions with TS or TD field = 0 -- a long distance instruction should have no affect.

Instructions with TS or TD field not 0 -- If an instruction with a source address with TS non zero is proceded by a LDS the source operand should be fetched using map file 2. If the instruction is preceded by a LDD, the LDD should not

affect the source fetch.

If an instruction with a destination address with TD non zero is preceded by a LDD, the destination operand is stored using map file 2.

If the instruction is preceded by an LDS, the LDS should not affect the destination store.

XOP -- The software is not affected by a long distance instruction.

The following instructions are executed, preceded by an LDS and LDD.

Throughout subtest 6, map file 2 is set up as follows:

L1	B1	L2	B2	L3	B3
0	0	0	0	0	0

This will cause any instruction that uses map file 2 to add >1000 to the address.

The following list of instructions is executed, preceded by an LDS instruction, then by an LDD instruction. The operation of the instruction should not be affected.

A	W0,W1	S	W0,W1
AB	W0,W1	SB	W0,W1
ABS	W0	SETO	W0
CLR	W0	SOC	W0,W1
DEC	W0	SOCB	W0,W1
DECT	W0	SWPB	W0
INC	W0	SZC	W0,W1

```
INCT  WO                SZCB  WO,W1
INV   WO                XOR   WO,W1
MOV   WO,W1             SLA   WO,0
MOVB  WO,W1             SRA   WO,0
NEG   WO                SRC   WO,4
                        SRL   WO,0
```

The following list of instructions are executed, preceded by an LDS instruction. Then the Double Operand Instructions (A thru SZCB) are executed, preceded by a LDD instruction.

The source operand should be mapped by map file 2. The address should be S6L2+>1000 or S6L3+>1000. When the double operand instructions are preceded by a LDD, the destnation address should be S6L4+>1000.

```
ABS   @S6L2             A     @S6L3,@S6L4
CLR   @S6L2             AB    @S6L3,@S6L4
DEC   @S6L2             MOV   @S6L3,@S6L4
DECB  @S6L2             MOVB  @S6L3,@S6L4
INC   @S6L2             S     @S6L3,@S6L4
INCB  @S6L2             SB    @S6L3,@S6L4
INV   @S6L2             SOC   @S6L3,@S6L4
NEG   @S6L2             SOCB  @S6L3,@S6L4
SETO  @S6L2             SZC   @S6L3,@S6L4
SWPB  @S6L2             SZCB  @S6L3,@S6L4
```

Next, the following instructions are tested, preceded by an

LDS instruction.

B, BL, BLWP, JMP, MPY (with Tx = 0 or not 0);

DIV (with Tx = 0 and not 0);

MOV with Ts = 0, 1, 2 (with and without index) and 3.

Next, the following instructions are tested, preceded by an LDS then an LDD.

The Immediate instructions --

AI, ANDI, LI, ORI, CI and LIMI

The Compare instructions --

C, CB, COC, CZC (these are tested with TS + TD zero and nonzero)

The XOP instruction -

The MOV (with TD = 0, 1, 2 with and without indexing and 3) instruction is tested, preceded by an LDD instruction.

The CRU instructions --

TB, SBZ, SBO, LDCR, STCR (NOTE: LDCR and STCR are not tested with Tx=0)

Others --

STST, XOR, STWP and LMF

6.0 MESSAGES

When this tests starts executing with one or more output devices, it will output the following message:

MAPTST 990/10 MAP LOGIC TEST VERSION XX/YY *

Where XX/YY is the date the test was released. This number is used to correlate the listing and the object code.
If the test runs without any errors, the header and completion messages will be as follows:

START MAP-FILE BIT TEST

BIT TEST COMPLETE

START MAP INTERRUPT TEST

INT TEST COMPLETE

BASE REG ADDER VERIFICATION TEST

BASE REG TEST COMPLETE

LMF LMPA INCREMENT TEST

THE CURRENT MEMORY CONFIGURATION IS:

ADDRESS (BYTE)

FROM TO

000000 010000

LMF TEST COMPLETE

INSTRUCTION TEST

INSTRUCTION TEST COMPLETE

TEST COMPLETE LOOP COUNT = 0001

6.1 ERROR MESSAGES

Whenever the test finds an error it outputs the error message number to the programmer's front panel.
The following is a list of the error message numbers.
NOTE: If an error message is print in several parts, the subsequent parts of a message will all have a message number of zero.

NUMBER	MESSAGE
1	SUBTEST 1

ERROR MAP FILE DATA:

	EXPECTED						READ					
MAP	L1	B1	L2	B2	L3	B3	L1	B1	L2	B2	L3	B3
0	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
1	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000
2	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000	0000

In order to find the error compare the data that was read (READ) back from the map files with the data that was written to the map files (EXPECTED). The bits that are different are the bits that are in error.

SUBTEST 2 or 6

20

ERROR

ADDRESS (IN LISTING) OF ERROR XXXX

WORK POINTER PC STATUS AT TIME OF ERROR WP= 1E7C PC=1BE8 ST=C00F

WORKSPACE DATA

0001 0001 FFFE 0004 0008 0000 0000 0000

0000 0AE0 1EA 1CB8 1FA0 0140 0428 C20F

To find out what the failure was, go to the listing for the MAP test. Be sure to check the version number of the test and the listing to make sure the listing and the test match. Look in the listing at the address XXXX given in the message. The instruction at that address should be a 'BLWP *W8'. The next part of the message shows what the program counter, the work space pointer, the status and the contents of the 16 work space registers were at the time of the BLWP. This information should show why the error message was printed out and help isolate the problem.

SUBTEST 2

21

ERROR WHEN DOING LDS WITH MAP FILE IN ROM AREA

MAP FILE 2 DATA

A C T U A L	E X P E C T E D
XXXX XXXX XXXX XXXX	XXXX XXXX XXXX XXXX

The above message will be output when the 6 data words loaded into map file 2 is not the same as the data in memory at address FFE0 through FFEE. Where the ACTUAL is the data loaded into map file 2 and the EXPECTED is what was in memory at address FFE0.

SUBTEST 3

30

ERROR IN THE MAP FILE 2 BASE REGISTER OR THE MAP ADDER CIRCUITRY

MAP FILE 2

L1	B1	L2	B2	L3	B3	AU ADDRESS
0000	0000	0000	0000	0000	0000	0020

LATCHED MAP ADDRESS

EXPECTED	ACTUAL
000020	000020

If an address chip failed then by comparing the expected value with the actual LMPA value and by looking at which base was being used it should be clear which chip is bad.

SUBTEST 4

41

EXPECTED MAP INT DID NOT OCCUR

MAP FILE 2 DATA = FFFF FF99 FFFF FF9A E000 FF9C

MEMORY ADDRESS = 2020

If limit register 3 is being tested and a carry is expected the check routine checks the map interrupt occured flag. If a map error interrupt did not occur, the above message will be output.

42

ERROR * DID 'LDS'

'SETO'

DATA STORED USING BASE 1 INSTEAD OF BASE 2

MAP FILE 2 DATA = 1000 009D 0000 009B 0000 009C

MEMORY ADDRESS = 0000

The check routine in the test checks the 4 buffers to see if the -1 is in the correct buffer. If the limit was not exceeded the -1 should be in the test buffer. If the data is not in the correct buffer, an error message is output.

43

ERROR UNEX MAP INTERRUPT

If the map interrupt occured during the test but was not 'EXPECTED' this means that all 3 limit registers were exceeded when they should not have been. In that case the above message is output.

SUBTEST 5

5

LMF ERROR

MAP FILE ADDRESS = 003000

MAP FILE 1 DATA

A C T U A L	E X P E C T E D
1000 1002 1000 1006 1000 100A	1000 1002 1000 1006 1000 100A

If the data loaded into map file 1 is not the same as the data in memory the above message will be output. Bits 11 thru 15 of the limit register are masked out when checking the load.

7.0 PART NUMBERS

FICHE KIT	2250561-0009	(SP)
PROGRAM DESCRIPTION	-2001	(ROFF SOURCE)
MAPTST, Standard Linked Test	-1006	(FLO)
	-9006	(LML)
	-7006	(LC)
MAPTSTOS, Office System Linked Module	-1010	(FLO)
	-9010	(LML)
	-7010	(LC)
MAPTSTPC, Standard Test Module	2250561-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)
MAPTSTO, O.S. Test Module	2250561-1004	(OBJ)
	-9004	(LIST)
	-2004	(SRC)
MAPMSG, Message Module	2250560-1003	(OBJ)
	-9003	(LIST)
	-2003	(SRC)

A

APPLICATION	
NEXT ASSY	USED ON
	7506

REVISIONS

LTR	DESCRIPTION	DATE	APPROVED

REV																													
SHEET																													

REV STATUS OF SHEETS	REV																						
	SHEET																						

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES TOLERANCES ANGLES ±1° 3 PLACE DECIMAL ± 010 2 PLACE DECIMAL ± 02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL STD 100

DWN — DATE

CHK

ENGR

QA

APVD

CONTR NO

DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS INCORPORATED
Equipment Group Dallas, Texas

PD, PROMPG, PROM PROGRAMMER INTERFACE TEST-990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	2250118-9901

SCALE — REV. B — SHEET 1 of 34

TI-0083 C

TABLE OF CONTENTS

1.0 SCOPE

The Prom Programmer Interface/Personality Card Diagnostic Test is used to test hardware located on the Prom Programmer Interface Module, in addition to any of the programmer Personality Cards listed in Section 3.1. The diagnostic will operate with the Prom Programmer Interface Module in any CRU slot in a 990 Computer.

2.0 REFERENCES

TITLE	PART NUMBER
Specification, Prom Programmer	944929-9901
Specification, PROM II Personality	937356-9901
Specification, EPROM II Personality	937351-9901
Specification, FPLA Personality	937371-9901
Model 990/4 Computer System Hardware Reference Manual	945251-9701
Model 990/10 Computer System Hardware Reference Manual	945417-9701
Model 990 Computer Diagnostic Handbook	945400-9701

3.0 EQUIPMENT AND SOFTWARE REQUIREMENTS

This section describes the minimum equipment requirements and the available object software for the diagnostic.

3.1 EQUIPMENT REQUIREMENTS

In addition to the hardware required by DOCS, which is specified in the Diagnostic Handbook, the following equipment may be required:

PROM Programmer Console, 115 VAC	944996-0001
PROM Programmer Interface Module	944925-0001
PROM Personality Card	945135-0001
Special PROM Test Adapter	947853-0004
EPROM Personality Card	945165-0002
Special EPROM Test Adapter	947853-0001
PROM II Personality Card	937363-0001
Special PROM II Test Adapter No. 1	947853-0005
Special PROM II Test Adapter No. 2	947853-0006
EPROM II Personality Card	937362-0001
FPLA Personality Card	937361-0001
Special FPLA Test Adapter No. 1	947853-0007
Special FPLA Test Adapter No. 2	947853-0008
Special FPLA Test Adapter No. 2	947853-0009
Half-Size Extender Card	226851-0001
A Dual Channel Oscilloscope	

3.2 SOFTWARE REQUIREMENTS

The Fully Linked Object test module is PROMPG PN 2250118-1006 (FLO).

4.0 LOADING

Loading procedures from all available media are found in the Diagnostics Handbook.

5.0 TEST EXECUTION AND DESCRIPTION

The Prom Programmer Interface/Personality Card Diagnostic (PROMPG) tests all the hardware located on the Prom Programmer Interface Module in addition to the hardware located on the Personality Cards listed in Section 3.1.

5.1 OPERATOR INTERFACE

PROMPG runs under control of DOCS. Refer to the section on DOCS in the Diagnostic Handbook for a detailed description of DOCS.

5.1.1 STANDARD DOCS INITIALIZATION

Reference Diagnostics Handbook.

5.1.2 PROGRAM PANEL DOCS INITIALIZATION

When running PROMPG in Front Panel Mode, besides asking the question IDLE ON ERRORS ?, the IT verb questions are asked.

5.2 AVAILABLE VERBS

In addition to the verbs supplied by the DOCS package, PROMPG also provides the verbs listed in Table 1.

TABLE 1

VERB	FUNCTION
IT	Initialize Test
E1	Interface Module Register Test
E2	Interface Module Logic Test
E3	Interface Module/Personality Card Scope Loop Test (Factory Use Only)
E4	Personality Card Loopback Test (Factory Use Only)
E5	PROM II Personality Card Test
E6	EPROM II Personality Card Test
E7	FPLA Personality Card Test
EA	Execute Verbs E1 and E2
L1	Loop on E1 Verb
L2	Loop on E2 Verb
L4	Loop on E4 Verb
L5	Loop on E5 Verb
L6	Loop on E6 Verb
L7	Loop on E7 Verb
LA	Loop on Verbs E1 and E2
PE	Print Error Count
AL	Alternate Writing Patterns
PQ	Set/Reset PQREG12 Line

PR	Toggle PREAD- Line
RW	Read/Write Registers
A1	PROM II Personality Card Loopback Test
P1	Simulate Programming for EPROM II
P2	Simulate Read Data for EPROM II
P3	Initiate Programming Sequence for EPROM II
F1	Toggle Output Address/Data Lines for FPLA
F2	FPLA Personality Card Loopback Test
F3	Simulate Programming for FPLA

5.2.1 IT VERB (Initialize Test)

The IT verb is used to set up the initial test parameters for the diagnostic. When the test first starts executing these questions are part of the initial set of questions the operator must answer.

The "IT" verb consists of the following messages:

PROM PROGRAMMER CRU BASE ADDRESS DEFAULT =0020

PROM PROGRAMMER INTERRUPT LEVEL DEFAULT =0006

5.2.2 EA VERB (Execute All)

This verb executes verb E1 followed by E2.

5.2.3 LOOP VERBS (L1, L2 and L4 through L7)

These verbs are used to loop on verbs E1, E2, and E4 through E7. The "L" verbs will loop until the operator terminates the execution by the striking the RETURN key on the interactive device.

Note: Part 3 is a loop type test and therefore does not have an "L" verb.

5.2.4 LA VERB (Loop On All)

This verb is used to loop on verbs E1 and E2. The "LA" verb will loop until the operator terminates the execution by striking the RETURN key on the interactive device.

5.3 VERB DESCRIPTIONS

5.3.1 EXECUTE VERBS

The following describes verbs E1 through E7 of the diagnostic.

5.3.1.1 E1 Verb (Interface Module Register Test)

E1 tests the data, address and pulse width registers on the Prom Programmer Interface Module by loading and reading back data patterns beginning with >001 and ending with >800 (>80 for the data registers and >20 for the pulse width registers). The pattern is shifted to the left after each load and read. This test should not only point out any lines tied high or low but should also find lines shorted to each other. The remainder of the test makes sure that the proper data is loaded into the proper register by clearing all registers, writing a pattern of all F's into one of the registers at a time and then reading that register and all other registers. If the output data happens to have been loaded into a register other than the one desired, an error message will be output which indicates which register was written to and which other registers were affected.

When E1 starts executing, it outputs the header message:

PROM PROGRAMMER INTERFACE MODULE REGISTER TEST

When it finishes it outputs:

TEST COMPLETE

5.3.1.2 E2 Verb (Interface Module Logic Test)

E2 does the following to test the logic on the Interface Module.

1. Loads pulse width register.
2. Checks that busy is not set after a load.
3. Starts programming cycle and tests that busy does not set.
4. Waits for end of one programming cycle and checks that busy resets.
5. Clears the interrupt mask and enables interrupts; starts programming cycle and tests to see that an interrupt occured; and finally, checks that no timing error occurred.
6. Starts another programming cycle. Verifies that busy is set, then resets GO to cause a programming error.
7. Tests that programming error bit sets.
8. Loads all registers with 1's.
9. Executes reset instruction and verifies that all registers reset.

When E2 starts executing, it outputs the header message:

PROM PROGRAMMER INTERFACE MODULE LOGIC TEST

When it finishes it outputs:

TEST COMPLETE

5.3.1.3 E3 Verb (Interface Module/Personality Card Test)

E3 is a loop type test for use in testing the Prom Programmer Interface Module in addition to the PROM, EPROM, PROM II and FPLA Personality Cards. IT will loop until interrupted with the RETURN key from the interactive device. It is suggested that an oscilloscope be used while executing E3 to check the timing on the Interface Module. A train of pulses, 1 MS in width, should be present at U34, PIN 12 on the Interface Module upon execution of E3. E3 also tests the program pulse on each of the Personality Cards mentioned. When testing the EPROM Personality Card use the EPROM Special Test Adapter (947853-0001). When testing the PROM Personality Card use the PROM Special Test Adapter (947853-0004), and for the FPLA Personality Card use the FPLA Special Test Adapter No. 3 (947853-0009).

A half-size extender card (226851-0001) is required to test the Interface Module.

Note: No error messages are output during this test. When E3 starts executing, it outputs the following message:

INTERFACE MODULE/PERSONALITY CARD SCOPE LOOP TEST,
VERIFY PULSE WIDTH=1 MS
USE SPECIAL TEST ADAPTER FOR THIS TEST
PRESS RETURN KEY TO END TEST

5.3.1.4 E4 Verb (Personality Card Loopback Test)

E4 is a loopback type test for use with either the PROM, EPROM or PROM II Personality Cards and requires that the Interface Module be connected to the Prom Programmer Console and one of the Personality Cards. This test also requires that the Special EPROM Test Adapter (947853-0001), PROM Test Adapter (947853-0004) or PROM II Test Adapter (947853-0006) be inserted in the Personality Card. The function of the special test adapter is to loop back the address lines sent to the Personality Card.

This test performs a rippling 1's test for the first 8 bits in the address register which are then sent back through to the output lines.

When E4 starts executing it outputs the header message:

PERSONALITY CARD LOOPBACK TEST
USE SPECIAL TEST ADAPTER FOR THIS TEST

When it finishes it outputs:

TEST COMPLETE

5.3.1.5 E5 Verb (PROM II Personality Card Test)

E5 is a two part, non-interactive test for the PROM II Personality Card. The first part uses comparators on the Personality Card to moniter three voltages during a programming cycle. The output of the comparators are wire OR'ed with three of the PROM output lines. The test initiates a programming pulse and then reads the PROM outputs. The three lines connected to these comparators should be low. The table below shows the voltages which are monitered and the output line used.

LINE	MONITERING
PROM5	Prom Vcc Voltage
PROM6	Chip Enable
PROM7	Programming Pulse

The second part is a data path verification test which uses PQADDR0 to cause the DATA-IN lines to be looped back to the DATA-OUT lines. All possible 8-bit patterns are applied to the inputs and the results compared with expected data. When E5 begins execution it prints out the following header message:

PROM II PERSONALITY CARD TEST

TESTING PROG PULSE, VCC AND CHIP ENABLE

TESTING DATA LINES

When the test is complete it prints the message:

TEST COMPLETE

5.3.1.6 E6 Verb (EPROM II Personality Card Test)

E6 is a test for use with the EPROM II Personality Card only. E6 tests the card with a pattern and loopback test and with a timing and voltage test.

The pattern/loopback test checks out all eight input/output data lines (PQDATA0-7 and PROM0-7) and their associated buffers by shifting a "1" through all bit positions starting with the least significant bit (PQDATA7/PROM7) and reading back the looped-back data. It also tests 11 of the 12 address lines (PQADDR1-11) and buffers in the same manner as with the data lines with the exception that all address lines are looped back through PROM5 by a hardware multiplexer.

The next portion of the test checks the status of three diagnostic signals: PVCCTS- (returned by way of PROM7), PLSECHK- (returned by way of PROM6) and POWRTST- (returned by PROM4). PVCCTS- tests for completion of the entire programming pulse sequence. PLSECHK- tests for completion of only the chip enable signal, CHPENA. POWRTST- checks for proper operation of all voltage

supplies and timing signals applied to the EPROM socket. E6 does not include a check of the correct length of the programming pulse. When E6 begins execution it prints out the following header message:

EPROM II PERSONALITY CARD TEST

TESTING INPUT/OUTPUT ADDRESS AND DATA LINES

TESTING DIAGNOSTIC SIGNALS

When the test is complete it prints the message:

TEST COMPLETE

5.3.1.7 E7 Verb (FPLA Personality Card Test)

E7 checks out six of the input/output data lines (PQDATA2-7 and PROM2-7) and their associated buffers by shifting a "1" through all bit positions starting with the least significant bit (PQDATA7/PROM7) and reading back the looped-back data. E7 also checks the operation of the program pulse, VCC and the 10.5 volt supply. The status of these signals is returned on the data line PROM1. This test does not include a check of the proper length of the programming pulse. When E7 begins execution it prints out the following header message:

FPLA PERSONALITY CARD TEST

TESTING INPUT/OUTPUT DATA LINES

TESTING DIAGNOSTIC SIGNAL

When the test is complete it prints the message:

TEST COMPLETE

5.3.2 DEBUG VERBS

The following describes the Debug verbs available in this diagnostic which are useful for scoping purposes in testing out each of the Personality Cards. Each of these verbs may be terminated by striking the ENTER key on the interactive device.

5.3.2.1 PE Verb (Print Error Count)

This verb is used to print out the error count. The format is:

ERROR COUNT = XXXX

5.3.2.2 AL Verb (Alternate Writing Patterns)

This verb is useful for scoping purposes when checking out the address and data lines and associated buffers coming across the interface cable. The user types in "AL" and the following appears on the interactive device:

```
ALTERNATE WRITING PATTERNS
1ST ADDRESS PATTERN (3 CHAR HEX FOLLOWED BY CAR. RETRN) DEF=XXX
1ST DATA PATTERN (2 CHAR HEX) DEF=XX
2ND ADDRESS PATTERN  DEF=XXX
2ND DATA PATTERN  DEF=XX
```

LOOP? (N=0,Y=1) DEF=1

With this verb the user may write any sequence of two patterns to the address and data lines simultaneously. When used with the EPROM II board the addresses are latched into the on-board address storage register.

5.3.2.3 PQ Verb (Set/Reset PQREG12 Line)

This verb is useful for scoping purposes when checking the operation of the PQREG12 output control line from the interface card to the Personality Card. If used in the toggle mode the verb provides a fast square wave output on PQREG12. If used in either the set or reset mode the verb executes a SBO or SBZ instruction to bit 12 on the CRU. The format for this verb is as follows:

SET/RESET PQREG12 LINE

MODE? (0=TOGGLE,1=SET,2=RESET) DEF= 0

5.3.2.4 PR Verb (Toggle PREAD- Line)

This verb is useful for scoping purposes when checking the operation of the PREAD- output control line from the Interface Module to the Personality Card. This verb provides a fast square wave output on PREAD-. The format for PR is as follows:

TOGGLE PREAD- LINE

5.3.2.5 RW Verb (Read/Write Registers)

The RW verb allows the reading and writing of any of the registers of the PROM Programmmer Interface Module. The user is asked to select the register to be read or written. Then the verb action is requested (read or write). If the write action is chosen, the last prompt will be for write data (8 bits). On a read, the 8 bits of read data will be printed in hexadecimal. The format for the RW verb is as follows:

```
READ/WRITE INTERFACE MODULE REGISTERS
SELECT REGISTER (0=MSB ADDR,1=LSB ADDR,3=DATA,
  5=PULSE WIDTH,7=PROM OUTPUTS)  DEF=0
READ OR WRITE? (0=READ,1=WRITE)  DEF=1
```

If the user wishes to write data to the register, the following message is printed out:

```
ENTER WRITE DATA  DEF=XX
```

If the user specifies to read the register, the following message appears:

```
READ DATA=XX
```

5.3.2.6 A1 Verb (Loopback Test for PROM II)

The A1 verb requires a Special Test Adaptor (947853-0005) which loops some of the address lines (PQADDR1-3) back through the PROM output lines. The verb applies all possible 3 bit patterns on the address lines and compares that pattern with the appropriate

bits in the PROM output. This verb, combined with the E4 verb and the PROM II Test Adapter (947853-0006) will loopback-test all address lines used on the PROM II Personality Card. The format for the A1 verb is as follows:

PROM II PERSONALITY CARD LOOPBACK TEST

When it finishes it outputs:

TEST COMPLETE

5.3.2.7 P1 Verb (Simulate Programming for EPROM II)

This verb is useful for scoping purposes when checking the complete operation of the programming sequence, including address and data setup, timing and FPLA operation (refers to the FPLA located on the EPROM II Personality Card). This verb can actually be used to program an EPROM of the correct type (refer to the specification for the EPROM II Personality Card) one word at a time. When used in conjunction with the P2 (Simulate Read Data) verb, the complete operation of the EPROM II Personality Card can be verified. The format for the P1 verb is as follows:

SIMULATE PROGRAMMING FOR EPROM II

ENTER EPROM TYPE (0=NOT USED,1=2516,2=2532,3=NOT USED,
4=NOT USED) DEF= 1

ENTER ADDRESS, DEF=XXXX

ENTER DATA, DEF=XX

LOOP? (N=0,Y=1), DEF=1

The user must specify which type of EPROM is in use. This places a "TYPE CODE" in three of the four most significant address bits and determines which of the EPROM socket pins the program pulse signal will be routed to by the on-board FPLA. For more information about this refer to the hardware specification for the EPROM II Personality Card, P/N 937351. This verb also checks PVCCTS- for proper completion of the program sequence.

If used in the loop mode, the verb will continue to program the desired data at the desired address location.

5.3.2.8 P2 Verb (Simulate Read Data for EPROM II)

This verb is useful for scoping purposes when checking the complete operation of the read sequence, including address setup and data return, timing and FPLA operation (refers to the hardware FPLA located on the EPROM II Personality Card). This verb can actually be used to read an EPROM of the correct type (refer to the specification for the EPROM II Personality Card) one word at a time. When used in conjunction with the P1 (Simulate Programming) verb, the complete operation of

the EPROM II Personlity Card can be verified. The format for the P2 verb is as follows:

```
SIMULATE READ DATA FOR EPROM II
ENTER EPROM TYPE (0=NOT USED,1=2516,2=2532,3=NOT USED,
 4=NOT USED)  DEF= 1
ENTER ADDRESS, DEF=XXXX
DISPLAY RESULT?, (0=NO, 1=YES) DEF=1
LOOP? (N=0,Y=1), DEF=1
```

The user must specify which type of EPROM is in use. This places a "TYPE CODE" in three of the four most significant address bits and determines which of the EPROM socket pins the control signals will be routed to by the on-board FPLA. For more information about this refer to the hardware specification for the EPROM II Personality Card, P/N 937351.

If used in the loop mode the verb will continue to read data from the desired address location.

If the user specifies to display the returned data, the resulting format appears as follows:

```
RETURNED DATA=XX
```

5.3.2.9 P3 Verb (Initiate Programming Sequence for EPROM II)

This verb is useful for scoping purposes when checking the operation of the program pulse and associated timing circuitry for the EPROM II Personality Card. If

used in the loop mode the verb provides a continuous train of program pulse timing sequences. The format for the P3 verb is as follows:

```
INITIATE PROGRAMMING SEQUENCE FOR EPROM II
ENTER EPROM TYPE (0=NOT USED,1=2516,2=2532,3=NOT USED,
 4=NOT USED)  DEF= 1
LOOP? (N=0,Y=1) DEF=1
```

The user must specify which of the type of EPROM is in use. This places a "TYPE CODE" in three of the four most significant address bits and determines which of the EPROM socket pins the program pulse signal will be routed to by the on-board FPLA. For more information about this refer to the hardware specification for the EPROM II Personality Card, P/N 937351. This verb also checks PVCCTS- for proper completion of the program sequence.

5.3.2.10 F1 Verb (Toggle Output Address/Data Lines for FPLA)

This verb is useful for scoping purposes when simultaneously checking the operation of the 12 FPLA Personality Card address lines (A-L) and control circuitry associated with each; the 6 data lines and associated circuitry; and the programming signals, all of which may be monitored at the FPLA socket.

The sequence of pulses which appear at the FPLA socket

address pins begins with 0 volts, steps to 5 volts, then to 10.5 volts and back to 0 volts. Each of the data lines should be toggling at this same time as should the programming signals.

The format for this verb is as follows:

TOGGLE FPLA OUTPUT ADDRESS/DATA LINES

5.3.2.11 F2 Verb (Loopback Test for FPLA)

F2 is a loopback type test and requires the use of a special FPLA test adapter (947853-0007 or -0008). The function of each of the special test adapters is to loop back the 12 address lines through to the 6 return data lines at the FPLA socket. The test performs a rippling 1's test for the most significant 6 address lines (using test adapter, 947853-0007) or the least significant 6 address lines (using test adapter, 947853-0008) depending on which adapter part number the user specifies.

When F2 begins execution it outputs the following messages:

FPLA PERSONALITY CARD LOOPBACK TEST

WHICH TEST ADAPTER IN USE (0=-7,1=-8) DEF=0

LOOP? (0=NO,1=YES) DEF=1

5.3.2.12 F3 Verb (Simulate Programming for FPLA)

This verb is useful for scoping purposes when checking

the complete operation of the programming sequence. This verb can actually be used to program a single minterm in a FPLA or program the control functions of the FPLA. The format for the F3 verb is as follows:

SIMULATE PROGRAMMING FOR FPLA

ENTER MINTERM ADDRESS DEF=XXX

ENTER PRODUCT TERM ADDRESS DEF=XX

SEQUENCE TYPE (PRODUCT=0,OR TERM=1,ENABLE=2) DEF=0

LOOP? (0=NO,1=YES) DEF=1

For more information about the operation of the FPLA Personality Card and the sequence of signals generated by this verb, refer to the hardware specification for the FPLA Personality Card, P/N 937371.

If used in the loop mode, the verb will continue to program the minterm under the conditions specified.

6.0 ERROR MESSAGES

Number	Message And Condition
1	Message: *ERROR* ADDRESS REGISTER: EXPECTED DATA=XXX, ACTUAL DATA=XXX Condition: This error indicates faults in the Interface Module's address register. The message reports expected and actual address patterns.

2 Message: *ERROR* DATA REGISTER:
EXPECTED DATA=XX, ACTUAL DATA=XX

Condition: This error indicates faults in the Interface Module's data register. The message reports expected and actual data patterns.

3 Message: *ERROR* PULSE WIDTH REGISTER:
EXPECTED DATA=XX, ACTUAL DATA=XX

Condition: This error indicates faults in the Interface Module's pulse width register. The message reports expected and actual 6-bit return patterns.

4 Message: *ERROR* BUSY SET BEFORE GO IS GIVEN

Condition: CRU input bit F was set to 1 at a time when the module was not in the programming mode. Input bit F should be set to 0.

5 Message: *ERROR* BUSY DID NOT SET AFTER GO IS GIVEN

Condition: CRU input bit F is reset to 0 after module has entered the programming mode. Input bit F should be a 1 at this time.

6 Message: *ERROR* BUSY DID NOT RESET AT THE END OF PROGRAMMING CYCLE

Condition: CRU input bit F is equal to 1 at a time when the module is not in the programming mode. Input bit F should be a 0 at this time.

7 Message: *ERROR* EXPECTED INTERRUPT DID NOT OCCUR

Condition: A timeout interrupt did not occur at the end of a programming cycle with the interrupts enabled.

8 Message: *ERROR* TIMING ERROR BIT SET

Condition: Programming sequence did not complete in expected amount of time.

9 Message: *ERROR* NO TIMING ERROR WHEN GO RESETS DURING PROGRAMMING CYCLE

Condition: CRU output bit F was reset to 0 prior to completion of programming sequence and the timing error bit, CRU input bit D, was not set to 1.

A Message: *ERROR* RESET DID NOT CLEAR UPPER BYTE OF ADDRESS REGISTER

Condition: Reset failed to clear address register.

B Message: *ERROR* RESET DID NOT CLEAR LOWER BYTE OF ADDRESS REGISTER

Condition: Reset failed to clear address register.

C Message: *ERROR* RESET DID NOT CLEAR DATA REGISTER

Condition: Reset failed to clear data register.

D Message: *ERROR* RESET DID NOT CLEAR PULSE WIDTH REGISTER

Condition: Reset failed to clear pulse width register.

E Message: *ERROR* LOOPBACK ERROR: EXPECTED DATA=XX, ACTUAL DATA=XX

Condition: A rippling 1's pattern is placed on the address lines and looped back to the data lines by a Special Adapter. The expected and actual returned patterns are reported in this message.

F Message: *ERROR* PROM PROGRAMMER UNEXPECTED INTERRUPT

Condition: Diagnostic has detected an error interrupt at a time it was not expected to happen.

10 Message: *ERROR* REGISTER SELECT ERROR: DATA WRITTEN IMPROPERLY TO INTENDED REG.

Condition: F's are loaded into one of the registers on the Interface Module while 0's are loaded into the rest. If the data is improperly loaded into the intended register then this error appears.

11 Message: *ERROR* DATA WRITTEN TO XXXXXX REG. APPEARED AT XXXXXX REG, EXPECTED DATA=XX, ACTUAL DATA=XX

Condition: The intention of this error is to point out problems in the register select multiplexer (U32). Data is written to one of the four sets of registers while the rest are cleared. If any data written to the selected register appears at those registers which were cleared, this error appears.

12 Message: *ERROR* PROG SEQUENCE COMPARISON FAILED EXPECTED: F8 ACTUAL: XX

Condition: The three data lines which moniter voltages on the PROM II personality card should be low during the programming cycle. See the description of the E5 verb for a

table which shows the bit assignment.

13 Message: *ERROR* PROM II DATA ERROR:
EXPECTED: XX ACTUAL: XX

Condition: The E5 verb executes a loop back test on the data-in (PQDATA0-7) and data-out (PROM0-7) lines of the PROM II Personality Card. This message shows the expected and actual data.

14 Message: *ERROR* PROM II UPPER ADDRESS LOOP BACK ERROR EXPECTED: X ACTUAL: X

Condition: Using a special adaptor, the A1 verb loops back the three most significant address bits (PQADDR1-3) onto the PROM da lines (PROM5-7). On error detection, the expected and actual data are displayed.

15 Message: *ERROR* PVCCTS- LINE FAILED TO GO HIGH AFTER RESET

Condition: PVCCTS-, the program sequence status line, which comes back to the Interface Module by way of PROM7, should have gone high after receiving a reset toggle signal from PQREG12.

16 Message: *ERROR* PLSCHK- LINE FAILED TO GO HIGH

AFTER RESET

Condition: PLSECHK-, the chip enable status line, which comes back to the Interface Module by way of PROM6, should have gone high after receiving a reset toggle signal from PQREG12.

17 Message: *ERROR* POWRTST- LINE FAILED TO GO HIGH AFTER RESET

Condition: POWRTST-, the timing/power supply status line, which comes back to the Interface Module by way of PROM4, should have gone high after receiving a reset toggle signal from PQREG12.

18 Message: *ERROR* PVCCTS- STATUS LINE FAILED

Condition: PVCCTS- should be in the active (low) state after a program sequence has completed but is not.

19 Message: *ERROR* PLSCHK- STATUS LINE FAILED

Condition: PLSECHK- should be in the active (low) state after a program sequence has completed indicating CHPENA (chip enable) is functional, but is not.

1A Message: *ERROR* POWRTST- STATUS LINE FAILED

Condition: POWRTST- should be in the active (low) state after a program sequence has completed indicating that all voltage supplies in conjunction with timing controls are functional, but is not.

1B Message: *ERROR* E6/E7 TEST ERROR:
EXPECTED DATA =XX, ACTUAL DATA=XX

Condition: This error occurs during the execution of E6 or E7. Data patterns are placed on the data output lines PQDATA0-7 and are looped back by EPROM II or FPLA Personality Card diagnostic hardware to the Interface Module. If the actual data patterns do not match the expected patterns then this message is printed. This checks out the data buffers and some other hardware on the cards.

1C Message: *ERROR* E6 TEST ERR:
EXPECTED ADDRESS =XXXX, ACTUAL DATA=XXXX

Condition: This error occurs during the execution of E6. Address patterns are placed on the address output lines PQADDR0-11 and are looped back by EPROM II diagnostic hardware to the Interface Module. If

the actual address patterns do not match the expected patterns then this message is printed. This checks out the address buffers and address latch along with some other hardware on the EPROM II board.

1D Message: *ERROR* PROM1 DATA LINE STUCK LOW

Condition: This message applies only to the FPLA Card. This line is the status return line for a hardware test of the voltage supplies. The line is reset to its initial inactive state (high) by PQDATA1. If the line fails to reset, this error is printed out.

1E Message: *ERROR* PROM1 TIMING/VOLTAGE TEST LINE FAILED

Condition: After PROM1 has been reset, the voltage supplies may be tested by executing a program sequence. After completion of the program sequence, this line should be active (low). If it fails to go active this error is printed out.

7.0 PART NUMBERS

PROGRAM DESCRIPTION 2250118-9901(PD)

SRC, PD ROFF SOURCE		-2001(SRC,PD)
LIST, PD ROFF LISTING		-9001(PD)
PROMPG, MAIN MODULE		-2003(SRC)
		-1003(OBJ)
		-9003(LIST)
PROMPG, LINKED TEST		-1006(FLO)
FULLY LINKED LIST		-9006(LML)
LINK CONTROL		-7006(LC)
PROMMSG, MESSAGE MODULE	2250285	-2003(SRC)
		-1003(OBJ)
		-9003(LIST)

A

APPLICATION	
NEXT ASSY	USED ON
	7506

REVISIONS			
LTR	DESCRIPTION	DATE	APPROVED

REV
SHEET

REV STATUS OF SHEETS — REV / SHEET

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± .010
2 PLACE DECIMAL ± .02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL STD 100

DWN | DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

PD, RAM05, RANDOM ACCESS MEMORY TEST-990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	2267303-9901

SCALE | REV. C | SHEET 1 of 31

TI-0023 C

1.0 SCOPE

This document describes the RAM05 DIAGNOSTIC (PN 2267303). The diagnostic is designed to test the Random Access Memory (RAM) of the 990/5 computer. Since the 990/5 computer is unmapped, it will only test 64K bytes of memory. 32K bytes of which may be off board memory. This diagnostic is a verb oriented test and will be used in conjunction with DOCS.

2.0 DOCUMENTATION

For information beyond the scope of this document reference the following sources:

TITLE	PART NUMBER
990 COMPUTER REFERENCE MANUEL	943442-9701
990 ASSEMBLY LANGUAGE MANUEL	943441-9701
DIAGNOSTIC HANDBOOK	943400-9701

3.0 EQUIPMENT AND SOFTWARE REQUIREMENTS

The following are the minimum requirements for this diagnostic to be executed properly:

EQUIPMENT

1) A 990/5 COMPUTER WITH 64K OR 32K BYTES OF MEMORY

2) AN APPROPRIATE LOADING DEVICE

3) APPROPRIATE LOAD MEDIA

4) AN APPROPRIATE I/O DEVICE

SOFTWARE

1) FULLY LINKED OBJECT:

   a) RAM05 (PN 2267303 -1006)

2) LINKABLE PARTS:

   a) RAM05 (PN 2267303 -1003)

   b) RM5MSG (PN 2267304-1003)

4.0 LOADING

The loading procedures for all loading devices and media is covered in the DIAGNOSTIC HANDBOOK (PN 943400-9701).

5.0 TESTS

There are five tests in this diagnostic. Each test has its place or use along with certain tradeoffs as to the sufficiency of the test, complexity of the test, and the test time. The following table shows how the tests vary according to the tradeoffs.

algorithm	functionality	dc addr	dynamic addr	dc pattern	access time	write rec	test time	comments
marching ones and zeros	yes	good	poor	poor	poor	poor	10n	minimumal test
write 1s thru 0s bckgrd	yes	good	poor	yes	poor	poor	2 x (n x n)	most widely used
refresh disturb	yes	excl	excl	poor	excl	good	6 x (n x n)	
galloping 1s and 0s	yes	excl	excl	yes	excl	poor	8 x (n x n)	all pos. transitions read-read
moving inversion	yes	excl	excl	yes	fair	excl	12 x (n x n)	all pos. transitions write-read

Functionality indicates how well the test checks memory.

DC Addr indicates how well the test checks memory addressing circuitry.

Dynamic Addr indicates how well the test checks for location to location changes.

DC Pattern indicates how well the test checks for cell to cell changes.

Access time indicates how well the test checks the memory timing cycles.

Test time indicates how long the test will take to execute to completion where n is the number of memory locations to be tested. This value will then be multiplied by the memory cycle time.

5.1 MARCHING ONES AND ZEROS

This test is a minimal memory test. The test is quick and shows that the addressing circuitry is working. The test works in the following manner:

a) the lower memory address for testing is moved to a register
b) the upper memory address for testing is moved to a register
c) a background pattern of zeros is written to memory
d) the test location is read and compared to the background pattern.
e) the test location is written to ones
f) increase the lower address by two
g) the addresses are compared to see if done
h) repeat steps d - g
i) starting at the upper memory address
j) read the test location and compare to the new background
k) write zeros to the test location
l) compate the addresses to see if done
m) decrease the upper address by two
n) repeat steps j - m

An error occurring in this test indicates one of the following happened:

1) the address could not be read by the test
2) the address location could not be written to by the test.
3) the address location did not retain the pattern that was written to it.

Recommended action would be:

1) make sure the testing is in a valid area not the TILINE PERIPHERAL CONTROL SPACE or the READ ONLY MEMORY locations
2) check the memory data paths for shorts or opens
3) replace the RAM chip in error

5.2 WRITE ONES THROUGH ZEROS BACKGROUND

This test checks for cell to cell shorts. It will also do some gross checking of the refresh circuitry. The test works in the following manner:

1) MEMORY INITIALIZE

   a) the lower memory location is moved to a register X

   b) a pattern is loaded into register Y

   c) the pattern is moved to the memory location

   d) see if done (i.e. does register X equal the upper memory location)

   e) the pattern in register Y is shifted to the right 1 bit

   f) increase the memory location in register X by two

   g) repeat steps c - f

2) MEMORY COMPARISONS

a) restore the lower memory location to register X

b) restore the pattern to register Y

c) read memory to see if the pattern is correct

d) see if done (i.e does register X equal the upper memory location)

e) the pattern in register Y is shifted to the right 1 bit

f) increase the memory location in register X by two

g) repeat steps c - f

If an error occurs in this test one of the following is the probable cause:

1) cell to cell shorts

Recommended action would be:

1) replace the RAM chip in error

5.3 REFRESH-DISTURB

This test will write a background pattern into the memory under test. It will then write the inverse of the pattern to alternate rows in memory. The inverse is written for a period of 2 seconds allowing plenty of time for a refresh to occur. After the 2 second period has elaped, the rows that haven't been modified are checked to see that no change has been made. The test works in the following manner:

a) set up software timing loops

b) save the beginning address

c) set the next row

d) write the inverse of the pattern to that row

e) if timing loops are done go to step h

g) increment the memory address and repeat d - e

h) verify that all of the unchanged memory locations are correct

i) if not done with that row restore timing loops and go to step g

j) if not done with all rows restore software timing loops and go to step c

If an error occurs in this test one of the following could be the possible cause:

1) REFRESH CIRCUITRY FAILURE

2) BAD MEMORY ACCESSING

3) CHIP FAILURE

The recommended action would be the following:

1) CHECK THE REFRESH CIRCUITRY

2) CHECK THE DATA PATH CIRCUITRY

3) REPLACE THE CHIP IN ERROR

5.4 GALLOPING ONES AND ZEROS

This test checks all possible writes followed by reads at different locations. The test works in the following manner:

a) a background pattern is written to the memory locations under test

b) register Y is the starting location

c) register X is the test location

d) increment register X by two

e) write the inverse of the pattern to register X

f) verify that register Y is unchanged

g) write the inverse of the pattern to register Y

h) write the original pattern to register X

i) mov register Y to register X

j) repeat steps d - i until register X is two greater than the highest address under test

k) start at the highest location and work back to the lowest location

If an error occurs in this test one of the following could be the probable cause:

1) BAD MEMORY ACCESSES

2) CHIP FAILURE

The recommended action would be:

1) CHECK THE MEMORY DATA PATHS FOR SHORTS OR OPENS

2) REPLACE THE CHIP IN ERROR

5.5 MOVING INVERSIONS

This test checks all possible reads and writes to memory. The test works in the following manner:

a) write a background pattern to memory

b) register Y is the starting location

c) verify that the background pattern is in register Y

d) write the new pattern to register Y

e) verify that the new pattern is in register Y

f) mov register Y to register X

g) increment register X by two

h) verify that register X has the background pattern

i) repeat steps g - h until register X is two greater than the highest memory location under test

j) increment register Y by two

k) repeat steps c - i

If an error occurs in this test one of the following could be the probable cause:

1) BAD MEMORY ACCESSES

2) CHIP FAILURE

The recommended action would be:

1) CHECK THE DATA PATHS FOR SHORTS OR OPENS

2) REPLACE THE CHIP IN ERROR

6.0 VERBS

By the use of the following verbs in the following paragraphs the operator can control the diagnostic.

6.1 IT OR INITIALIZE TEST VERB

This verb is used by the operator to control the area of memory to be tested. The operator must answer the following questions:

INPUT THE BEGINNING ADDRESS DEFAULT = XXXXX

To answer this question the operator may take the default by merely pressing the "RETURN" key. Or the operator may sustitute an new value and then press the "RETURN" key.

INPUT THE ENDING ADDRESS DEFAULT = XXXXX

To answer this question the operator follows the same procedure as in the first question.

DISPLAY CHIP LOC. ON ERROR? (0=NO,1=YES) DEF = 1

If the operator takes the default on this question the diagnostic will display the location of the memory chip in error for the onboard memory.

EXECUTE EA VERB? (DEF = 1)

If the operator takes the default on this question the diagnostic will execute tests 1 - 3. If the operator doesn't take the default "VERB" will be displayed allowing the operator to choose a DOCS verb or another of the test's verbs.

6.2 EA OR LA, EXECUTE ALL OR LOOP ON ALL TESTS VERB

The EA and the LA verbs execute tests 1 - 3. If the EA verb is used, at completion "VERB" will be displayed allowing the operator to choose a DOCS verb or another of the test's verbs. The LA verb causes continuous execution of these tests until the operator stops the execution. At which time "VERB" is displayed.

6.3 E1 OR L1, EXECUTE TEST ONE OR LOOP ON TEST ONE VERB

The E1 and the L1 verbs execute test 1. If the E1 verb is used, at completion "VERB" will be displayed. This allows the operator to choose a DOCS verb or another of the test's verbs. The L1 verb causes continuous execution of test 1 until the stops the execution. At which time "VERB" is displayed.

6.4 E2 OR L2, EXECUTE TEST TWO OR LOOP ON TEST TWO VERB

The E2 and the L2 verbs execute test 2. If the E2 verb is used, at completion "VERB" will be displayed. This allows the operator to choose a DOCS verb or another of the test's verbs. The L2 verb causes continuous execution of test 2 until the stops the execution. At which time "VERB" is displayed.

6.5 E3 AND L3, EXECUTE TEST 3 OR LOOP ON TEST 3 VERB

The E3 and the L3 verbs execute test 3. If the E3 verb is used, at completion "VERB" will be displayed. This allows the operator to choose a DOCS verb or another of the test's verbs. The L3 verb causes continuous execution of test 3 until the stops the execution. At which time "VERB" is displayed.

6.6 E4 AND L4, EXECUTE TEST 4 OR LOOP ON TEST 4 VERB

The E4 and the L4 verbs execute test 4. If the E4 verb is used, at completion "VERB" will be displayed. This allows the operator to choose a DOCS verb or another of the test's verbs. The L4 verb causes continuous execution of test 4 until the stops the execution. At which time "VERB" is displayed.

6.7 E5 AND L5, EXECUTE TEST 5 OR LOOP ON TEST 5 VERB

The E5 and the L5 verbs execute test 5. If the E5 verb is used, at completion "VERB" will be displayed. This allows the operator to choose a DOCS verb or another of the test's verbs. The L5 verb causes continuous execution of test 5 until the stops the execution. At which time "VERB" is displayed.

6.8 PE OR PRINT ERROR VERB

This verb will display the cumulative total of errors.

6.9 SL OR SCOPE LOOP VERB

This verb allows the operator to input an address and data to be written to that address. Then the verb will loop on that address allowing the operator to check that address with an oscilliscope.

7.0 MESSAGES

There are three type of messages associated with RAM05. They are described in the following paragraphs.

7.1 INITIALIZATION MESSAGES

These messages ask the operator for some type of input. They are explained in detail in section 6.1 .

7.2 HEADER MESSAGES

These messages inform the operator of some action that the test has taken. The messages are below:

990/5 RANDOM ACCESS MEMORY TEST VERSION MM/YY/**

This is the test title and is displayed every time the test is loaded. MM indicates the month that the test was last upgraded. YY indicates the year the test was last upgraded. ** indicates the revision level of the test.

MEMORY UNDER TEST

XXXX TO YYYY

This message is displayed at the start of each subtest. XXXX is the beginning address of the memory under test. YYYY is the ending address of the memory under test.

START TEST XX

This message is displayed at the start of each subtest where XX is the number of the subtest.

SUBTEST COMPLETE

This message is displayed at the end of each subtest

TEST COMPLETE

This message is displayed at the end of execution of an EA or LA verb.

LOOPS COMPLETED = XXXX

This message is displayed at the end of execution of the Lx verbs.

XXXX is the number of loops completed.

7.2 ERROR MESSAGES

There are three type of error messages associated with RAM05. They are described below:

7.2.1 LEVEL TWO INTERRUPT ERROR MESSAGES

These errors are generated when for some reason the computer got one of the following:

1) PARITY ERROR
2) TILINE TIMEOUT

A PARITY ERROR is caused when the computer received bad data from memory. A TILINE TIMEOUT is caused when the computer tried to access non-existant memory or a TILINE slave device accessed didn't respond.

Recommended action would be:

1) reload and see if repeatable
   if repeatable reload DOCS and Diagnostic
   to high memory and see if error in memory
   located.
2) check the TILINE slave devices for correct operation
3) replace the TMS 9900 chip

7.2.2 IT ERROR MESSAGES

These messages are displayed when the operator has input an invalid response. They are self-explanitory and are show below:

THE ADDRESS GIVEN IS WITHIN THE PROGRAM.
PLEASE PICK AN ADDRESS OUTSIDE THE PROGRAM.

YOU MUST ENTER THE BEGINNING AND ENDING ADDRESS OF THE AREA OF MEMORY TO BE TESTED. THE ADDRESSES MUST NOT BE LESS THE 256 WORDS APART. BOTH ADDRESSES MUST BE ABOVE OR BELOW THE PROGRAM AND MUST NOT OVERLAP INTO THE PROGRAM AREA.

7.2.3 TEST ERROR MESSAGES

There are two type of error messages displayed by one of the tests. One is for onboard memory errors. The other is for off board memory errors. The message displayed for the onboard memory errors is shown below.

7.2.3.1 ONBOARD ERROR MESSAGES

This error message is displayed when the diagnostic finds a faulty component in the onboard memory.

ACTUAL DATA IS AAAA

EXPECTED DATA IS EEEE

BIT IN ERROR IS BBBB

BAD CHIP LOC. IS LLLL (LOWER 16K) OR UUUU (UPPER 16K)

AAAA INDICATES WHAT WAS READ FROM THE MEMORY

EEEE INDICATES WHAT THE MEMORY READ FROM MEMORY WAS SUPPOSED TO BE

BBBB INDICATES WHICH BIT IN THE CHIP FAILED

LLLL INDICATES THE LOCATION OF THE CHIP THAT FAILED IN THE LOWER 16K BANK OF MEMORY

UUUU INDICATES THE LOCATION OF THE CHIP THAT FAILED IN THE UPPER 16K BANK OF MEMORY.

Recommended action would be:

1) replace the chip that failed

8.0 PART NUMBERS

TITLE	PART NUMBER	
PROGRAM DESCRIPTION	2267303-2001	ROFF SOURCE
	-9001	ROFF OUTPUT
	-9901	ROFF DOCUMENT
FICHE KIT	2267303-0009	SP
RAM05 - LINKED TEST	2267303-1006	FLO
	-7006	LC
	-9006	LML
TEST PARTS		
RAM05	2267303-1003	OBJ
	-2003	SRC
	-9003	LST
RM5MSG	2267304-1003	OBJ
	-2003	SRC
	-9003	LST

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																														
SHEET																														
REV STATUS OF SHEETS	REV																													
	SHEET																													

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± .010
2 PLACE DECIMAL ± .02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL-STD-100

	DWN	DATE
	CHK	
	ENGR	
	QA	
	APVD	
	CONTR NO	
	DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

PD, RAM10, 990/10 RANDOM ACCESS MEMORY TEST -990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	2250119-9901

SCALE | REV. A | SHEET 1 of 33

TI-0923-C

990/10 RANDOM ACCESS MEMORY TEST

1.0 SCOPE

This document describes the RAM10 test module. This module is designed to operate as a linked deck operating with DOCS. The test is designed to test memory in the 990/10 computer system. The area to be tested is under operator control as is the choice of tests to be run. (See 6.0)

1.1 REFERENCES

1.1.1 DOCUMENTATION

For information beyond the scope of this document reference the following documents.

PART NUMBER	TITLE
943442-9701	Model 990 Computer Reference Manual
943441-9701	Model 990 Computer Assembly Language Manual
943447-9701	Model 990 Computer 733 Asr System Operation Guide
945253-9701	Model 990 Computer Floppy Disc Installation

	And Operation Guide
948964	48K Controller with ECC
948959	Memory Add-on Module, 128K 990/10
945400-9701	Diagnostic Handbook

1.1.2 LOADABLE TEST MODULE

The loadable test module is RAM10, 2250119-1006 (FLO).

Linkable Parts: RAM10, RAM10MSG

2.0 EQUIPMENT REQUIREMENTS

A. Model 990 Computer with 8K-32K of memory.

(1) 1 MEG on mapped systems

B. Interactive device

C. Loading device

3.0 LOADING INFORMATION

Refer to Diagnostic Handbook for loading procedures. When the test has loaded, the following messages will be output:

```
RAM10 990/10 RANDOM ACCESS MEMORY TEST VERSION MM/YY *X

THE CURRENT MEMORY CONFIGURATION IS:
ADDRESS (BYTE)
FROM TO
000000 020000

TYPE OF TEST DESIRED?
(0=16K RAM ECC ON OR 4K RAM, 1=16K RAM ECC OFF,UNCORRECTED DATA )
(2=16K UNCORRECTED DATA REPLACING ECC FIELD )  DEFAULT = 00 -

ENTER ADDRESS OF TILINE PERIPHERAL CONTROL SPACE,
ADDRESS DEFAULT = FB00 -

INPUT BEGINNING ADDRESS DEFAULT = 003098 -
INPUT ENDING ADDRESS DEFAULT = 00FFFE -

EXECUTE EA VERB? (DEF=1) -
```

These messages are also output for the IT verb and should be responded to according to the amount and type of memory to be tested. The following guidelines should be followed when initializing the test.

1. Memory of diferrent types(4K parity,4K ECC,16K ECC) should never be tested together at one time. If there

is more than one type of memory then one type should be thoroughly tested and then the next type should be tested. Do not allow the test addresses to overlap into different memories.

2. All 4K memories should be run under option 0, only this test type is valid for 4K chips. The Tiline Peripheral Control Space address default should always be set to 0000 for 4K memories, to avoid inadvertantly reading another devices control space. The beginning and ending address should follow the rules in the Test Area section(4.0.1). For 4K RAMS, the memory test should first be run with the ECC turned off on the board. The tests should then be repeated again with the ECC turned on. Therefore, if a memory error occurs with the ECC turned on, the error will reside in the ECC field. The diagnostic will try to locate the bad ECC bit by writing data patterns to toggel the six ECC bits.

3. For 16K RAMS, memory tests 1-5 can be executed in 3 different configurations. The standard method for testing 16K RAMS is with the ECC turned on. The 16K controller will then report the row and bit if a correctable error occurs. If a non-correctable error occurs, the diagnostic will report the error address, received and expected data. Option 1 allows test 1-5

to run with ECC off,errors here will report the expected and actual data for the 16 data bits and the address at the time of error. Option 2 will write the 6 MSB into the ECC field. Errors under this option also give the error address, expected and received data, but for the ECC bits(16-21) and data bits 6-15.

After the module has been properly intialized test 1-5 or any other RAM10 verb may be run.

3.1 VERBS

The available verbs are:

IT - Initialize the program

EA - Execute tests 1, 2, and 3 in order and return to DOCS

LA - Execute tests 1, 2, and 3 and loop thru this test sequence

E1-E6 - Execute a single test once. (tests 1 thru 6)

L1-L6 - Execute a single test and loop. (tests 1 thru 6)

PE - Print Errors

RM - Read Memory

SL - Execute Scope Loop.

NOTE: Interrupts 2-15 are masked out for test options 1 and 2 for 16K RAMS.

Any test may be suspended by entering any character on the keyboard of the interactive device when using option 0.

3.2 PE VERB

The PE verb may be used to find the error count when error messages have been suppressed.

EXAMPLE: VERB? - PE
ERROR COUNT = XXXX

3.3 RM VERB

The Rm verb is used to read all words between two given address while looking for parity errors. The RM verb has the following intialization messages.

ENTER ADDRESS TO START READING FROM DEFAULT = 000000
ENTER END ADDRESS, DEFAULT = 00FFFE

The addresses are set up in the same way as the beginning and ending address in the test. As the test starts the following message is output.

START READING MEMORY FOR PARITY ERRORS FROM XXXXXX TO XXXXXX

The following message shows the test finished and how many parity errors were found during test.

MEMORY READ COMPLETE, NUMBER OF ERRORS = XX

3.4 SL VERB

The SL verb sets up a scope loop. It has the following 2 messages.

INPUT ADDRESS YOU WISH TO WRITE TO - XXXXXX
INPUT DATA YOU WISH TO WRITE - XXXX

This verb continuously writes the data given into the address given. You may not give an address inside the program. The scope loop test will only end when someone enters an @.

4.0 TESTS PERFORMED

4.0.1 TEST AREAS

The area of memory to be tested is chosen by the operator during initialization, and may be changed at any time by use of the "IT" verb.
The messages displayed are:

INPUT BEGINNING ADDRESS DEFAULT=002884 -
INPUT ENDING ADDRESS DEFAULT=00FFFE -

The adresses entered are edited by the following rules:

1. Beginning address < ending address
2. Beginning address <or=256 words <or=ending address or 32K boundary
3. A0 <beginning address <ending address <program starting address

   or

   program ending address < beginning address < ending address

If any of these three edits fails the following message

will be displayed.

ERROR* ONE OR BOTH OF THE INPUT ADDRESSES IS INVALID YOU MUST ENTER THE BEGINNING AND ENDING ADDRESSES OF THE AREA OF MEMORY TO BE TESTED. THE ADDRESSES MUST NOT BE LESS THAN 256 WORDS APART. BOTH ADDRESSES MUST BE ABOVE OR BELOW THE PROGRAM AND MUST NOT OVERLAP INTO THE PROGRAM AREA, AND MUST START ABOVE MEMORY LOCATION A0.

This is the only editing done on the addresses. Testing areas containing ROM, or unimplemented memory will cause errors. To verify the memory area actually under test the following message outputs at the beginning of each test.

MEMORY UNDER TEST = 002884 TO 00FFFE

4.0.2 TEST TIMING

The timing of these tests varies directly as the size of the test area changes according to the following formulas:

TEST 1 - (10N) X A
TEST 2 - (12N) X A + (4 / TIME DELAY FOR REFRESH)
TEST 3 - (12N) X A + (2 / TIME DELAY FOR REFRESH)
TEST 4 - (12N**2) X A
TEST 5 - (192 X N X 2N) X A

N = area to be tested in words
A = access time of memory under test

NOTE: These times are minimum and do not include instruction execution times.

Due to the possible long times of program execution for tests 4 and 5, the front panel will have blinking lights as an indicator that the test is actually executing.

4.0.3 DATA CHECKING

The memory tests check data in 3 ways:

1. The interrupt vector for level 2 interrupts checks for parity errors.
2. As data is written a data comparison check is made for errors not causing parity interrupts.
3. For 16K chips the single bit error bit in the Tiline Peripheral Control Space is checked after every read.

4.1 TEST 1 (MARCHING ONE'S AND ZEROS)

This is a minimal test of a memory to reasonably assure that it is functional (that is, addressing operated and each cell can be read and written in the I/O state). The memory is first written to the all zeros state. Then, sequentially, starting at the first address, the zero is read and a one is written. This sequence is continued to the last location. The memory will be full of ones. Then starting at the highest location, a one is read and a zero is written. The address is decremented and the sequence is repeated until the lowest location is reached. This overall sequence is then repeated with the data reversed. As the memory is being scanned in the ascending direction, any effect on a location above will be detected when it is eventually read. If the effect was felt on a location below, it will not be detected until the memory is scanned in reverse. As the memory is being scanned in the descending direction, any effect on a location below will be detected when it is eventually read. This, by no means, tests everything or all interactions, but does reasonably assure that the memory is working.

4.2 TEST 2 (WRITE ONE THRU ZEROS BACKGROUND)

This test is to write a bit level pattern of alternate ones and zeros thru a background. After the pattern of 10101010 the pattern is changed to 01010101 and the test repeated. Once both patterns have completed the background is inverted and the memory test is repeated. The purpose of this test is to check for cell to cell shorts by putting bit level patterns in memory. The test also performs a basic test of the refresh logic. This test is done by a 2 second wait between the time the test pattern is written and the pattern compare, allowing the cells to decay before the cells are read. An error in this test can be the result of either refresh problems or cell to cell shorts. A check of the actual and expected data may aid in determining the cause of the error.

4.3 TEST 3 (REFRESH DISTURB)

This test checks the refresh circuits and the ability of a memory cell to maintain the proper state between refresh cycles. The test first writes a background pattern in memory. Then the inverse of the background is written in alternate rows for a period of 2 seconds. By writing in alternate rows the memory is disturbed

while allowing the test area to remain idle for a period longer than the refresh cycle. After the delay the rows that have not been written into are checked to insure that the pattern has not been disturbed. The background is rewritten and the order of the rows switched checking the rows that were not checked in the first cycle. When this completes the background is inverted and the test repeated. This test is hardware logic dependent since it depends upon the chip address lines to insure it is accessing the alternate rows properly. If the addressing logic changes it is possible that this test will give invalid results.

4.4 TEST 4 (GALLOPING ONES AND ZEROS WRITE-WRITE)

This pattern tests all possible writes followed by reads at different locations. A background B is written throughout the memory. Every pair of addresses are then checked in the following manner, starting in the first location:

Write T in the second location, read B in the first location, write B in the second location, read B in the first location, write T in the third location, read B in the first location, etc. After all locations have been checked in relation to location one, the sequence

is repeated with respect to location two, then three, etc., throughout the entire memory. At completion, the above sequence is repeated with the patterns interchanged.

4.5 TEST 5 (MOVING INVERSIONS READ-WRITE-READ)

This test executes by inverting a field of zeros to a field of ones and vice versa and generating the addresses in different sequences. The test begins execution by writing a background of o's. Every address is then checked in the following manner and starting in the first location:

Read each word and verify the previous pattern, modify a single bit in the data pattern, then read that the new pattern is there. The whole test is repeated again by incrementing through all addresses by 2,4,8 on up to 8000, then testing again all addresses that were skipped, in the same manner.

4.6 TEST 6 (ECC PATTERN TEST)

The object of this test is to verify the ECC logic for 16K RAM. First, data patterns are written with the ECC

turned on, then the ECC turned off and the stored ECC is checked for correctness. Data patterns are selected to cause a sliding 1 and sliding 0 pattern to be calculated for the ECC data. Then, a sliding 1 pattern is developed for a data pattern and a check of the calculated ECC is made. Therefore, all ECC data bits are toggled and each data bit ECC pattern is checked. Also, the ability of the 16K RAM controller to report a 1 bit error is verified by modifying the stored ECC pattern for a bit that is turned on. Test 6 is run on only one bank of the controller, the bank is dependent on where the program is loaded.

5.0 RAM10 MESSAGES

MESSAGE	DESCRIPTION
RAM10 990/10 RANDOM ACCESS MEMORY TEST VERSION = MM/YY *V	Header message
POWER RESTORED	Level 0 interrupt occurred
TYPE OF TEST DESIRED? (0=16K RAM ECC ON OR 4K RAM, 1=16K RAM ECC OFF,UNCORRECTED DATA) (2=16K UNCORRECTED DATA REPLACING ECC FIELD) DFAULT = 00	Initialization question from IT verb
ENTER ADDRESS OF TILINE PERIPHERAL CONTROL SPACE, ADDRESS DEFAULT=FB00	Initialization question from IT verb
MEMORY UNDER TEST = XXXXXX TO XXXXXX	Area of memory currently being tested
START TEST XX	Test number message
LOOP COUNT = XXXX	Loop count at end of test
SUBTEST COMPLETE	Subtest-ending message

TEST COMPLETE	Test-ending message
INPUT BEGINNING ADDRESS DEFAULT = XXXXXX	Intialization question from IT verb
INPUT ENDING ADDRESS DEFAULT = XXXXXX	Intialization question from IT verb
YOU MUST ENTER THE BEGINNING AND ENDING ADDRESSES OF THE AREA OF MEMORY TO BE TESTED. THE ADDRESSES MUST NOT BE LESS THAN 256 WORDS APART. BOTH ADDRESSES MUST BE ABOVE OR BELOW THE PROGRAM AND MUST NOT OVERLAP INTO THE PROGRAM AREA, AND MUST START ABOVE MEMORY LOCATION A0.	Directions for properly inputting test address boundaries
ERROR COUNT = XXXX	Total number of errors
THE CURRENT MEMORY CONFIGURATION IS: ADDRESS (BYTE) FROM TO 000000 XXXXXX XXXXXX XXXXXX	During intialization this message shows how memory the system contains
16K RAM CONTROLLER PRESENT	Tests are run on 16K RAM
START 16K RAM ECC PATTERN TEST	Header message for test 6
TO EXIT TEST ENTER @ START SCOPE LOOP TEST	Starting message for scope loop test

INPUT ADDRESS YOU WISH TO WRITE TO-	Initailization question from SL verb
INPUT DATA YOU WISH TO WRITE-	Intialization question from SL verb
ENTER ADDRESS TO START READING FROM DEFAULT = XXXXXX	Intialization question from RM verb
ENTER END ADDRESS DEFAULT = XXXXXX	Intialization question from RM veb
START READING MEMORY FOR PARITY ERRORS FROM XXXXXX TO XXXXXX	Starting message for RM verb
MEMORY READ COMPLETED, NUMBER OF ERRORS = XXXX	Ending message for RM verb
PARTS OR ALL OF DOCS AND RAM10 ARE LOADED INTO BANKS A & B. ECC IS LEFT ON IN THESE BANKS, USE DOCS TO RELOCATE THE PROGRAMS IF BANK A OR B IS TO BE RUN WITH ECC OFF.	Warning message about where DOCS and RAM10 are loaded
ALL LOGGED ERRORS WERE REPORTED. TOTAL ERRORS = XXXX	Informative message at test completion
THE FOLLOWING ERROR WAS LOGGED XXXX TIMES.	Informative message at test completion

5.1 ERROR MESSAGES

Error messages are displayed on the interactive or error message device. The message number is displayed on the front panel. On systems without an interactive device, the source of error message information is the front panel.

On systems with expansion chassis all memory boards in the expansion chassis may have the error lights come on during initialization. These lights cannot be easily reset and may stay on during the test without there being an actual error.

5.1.1 ERROR MESSAGE 1

ACTUAL DATA IS XXXX
EXPECTED DATA IS XXXX

This message will be displayed when data in memory is not the same as the test data, but parity is correct. This error will occur for 16K RAMS with ECC turned off (option 1), and also with the 4K RAMS, with ECC off or on.

5.1.2 ERROR MESSAGE 2

BIT IN ERROR IS XXXX

This message will occur only for 4K RAM ECC failure.

5.1.3 ERROR MESSAGE 3

PARITY ERROR

This message is displayed when a parity error occurs for 4K or 16K RAMS.

5.1.4 ERROR MESSAGE 4

LOCATION OF 16K RAM CHIP FAILURE
BIT = XXXX ROW = XXXX

This message is displayed when testing 16K RAMS with ECC turned on (option O). If the row in error is the same as the bank the program is loaded in, only the single bit error reported will print out. The chip in error must be replaced or the program must be moved to a good bank for the test to catch any other errors.

5.1.5 ERROR MESSAGE 5

TILINE TIMEOUT OCCURRED

This message is printed if a Tiline timeout occurs during the memory test.

5.1.6 ERROR MESSAGE 6

ONE BIT ERROR DID NOT OCCUR AT BIT 4

This message will be displayed if the 16K RAM controller does not respond to an invalid ECC pattern. This message from test 6 only.

5.1.7 ERROR MESSAGE 7

MEMORY BITS	16-21	6-15
ACTUAL DATA	XX	XXXX
EXPECTED DATA	XX	XXXX

This message will be displayed if a data comparison error occurs with a 16K RAM that has uncorrected data

replacing the ECC field (option 2). Or if a 16K RAM parity error has occurred.

5.1.8 ERROR MESSAGE 8

ONE BIT ERROR EXPECTED AT BIT 4 OCCURRED AT XXXX

This message will be displayed if the 16K RAM controller does not properly respond to the expected error condition. This message from test 6 only.

5.1.9 ERROR MESSAGE 9

16K RAM CONTROLLER NOT FOUND SUBTEST ABORTED

This message is printed if the 16K RAM controller cannot be located. This message from test 6 only.

5.1.10 ERROR MESSAGE 10

THE ADDRESS GIVEN IS WITHIN THE PROGRAM. PLEASE PICK AN ADDRESS OUTSIDE THE PROGRAM.

This message is printed when the address picked for the

scope loop is within the program bounderies.

5.1.11 ERROR MESSAGE 11

**ERROR* ONE OR BOTH OF THE INPUT ADDRESSES IS INVALID.

This message is printed when one or both of the addresses input for test 1-5 are incorrect.

5.1.12 ERROR MESSAGE 12

ERROR ADDRESS IS XXXXXX

This message is printed when an error at the address printed occurs.

5.1.13 ERROR MESSAGE 13

AN ERROR NOT CAUSED BY ECC LOGIC HAS OCCURRED, USE TEST 1-5 TO ISOLATE ERROR.

While running test 6 an error not caused by the ECC logic has occurred.

5.1.14 ERROR MESSAGE 14

NO MEMORY ON CONTROLLER BOARD, OR PROGRAM LOADED INTO THE ONLY BANK(S) AVAILABLE. SUBTEST 6 ABORTED.

Subtest 6 has no empty bank available to run test on.

5.1.15 ERROR MESSAGE 15

THE ERROR FILE HAS OVERFLOWED, ALL ERRORS WILL NOW BE REPORTED. DUPLICATE ERRORS NOT YET REPORTED, WILL BE REPORTED AT TEST COMPLETION.

The error file has logged 25 different errors and overflowed. All errors will now be reported by the error reporting device, no additional errors will be logged. Any duplicate errors already logged will be reported on the error device after the test has completed.

5.1.16 ERROR MESSAGE 16

SUSPECTED ERROR IN ECC BITS OR PARITY BIT. FOR ECC

MEMORY, REPLACE DATA CHIPS WITH ECC CHIPS AND RERUN TEST WITH ECC OFF.

This message is disayed only for a level 2 interrupt in 4K chips when the actual and expected data compare. The ECC chips for the row where the error occurred should be taken out and moved to 6 data chip locations. The ECC switches on the controller board should be turned off and the test rerun. This should indicate which ECC chips are bad by doing data comparisons.

6.0 PROGRAMMING CONSIDERATIONS

As this test is under operator control care and good judgement must be used to insure proper test operation. For example testing location 0-100 would destroy the interrupt vectors. Attempting to test a location which contains memory implemented as ROM or testing locations in memory that do not exist will give unpredictable results. The operator should be aware that ECC cannot be turned off in the banks that RAM10 and DOCS are loaded into.

6.1 INTERRUPTS

All interrupts with the exception of level 0, 1, and 2 are handled by DOCS. If a parity error or Tiline timeout occurs, the error address will be printed. A parity error and Tiline timeout will attempt to recover. At test completion all DOCS interrupts are restored. RAM10 interrupt traps are now set up at the start of each test instead of only during initialization.

6.2 MAPPED SYSTEMS

The test will operate with systems using mapped memory. Although the diagnostic can test memory larger than 32K, the test will execute in blocks of 32K. This will reduce the number of map changes required and therefore, shorten the execution time. Because of larger memories now being tested, tests 1,2,and 3 will now blink the front panel lights (from 5555 to AAAA) whenever the mapping file changes. This will indicate the test is still running when testing larger memories.

6.3 OPERATING SUGGESTION

When operating test 4, operating in 1K blocks will be more efficient than testing larger blocks. To verify this, see the testing timing formulas in section 4.2.

6.4 MEMORY CONFIGURATION

Use of the IT verb will determine the memory configuration in 4K blocks including non-contiguous

areas.

Example:

```
THE CURRENT MEMORY CONFIGURATION IS:
ADDRESS (BYTE)
FROM          TO
000000      010000
020000      030000
```

7.0 PART NUMBERS

OPERATING PROCEDURE	2250119-9901(PD)
FICHE KIT	-0009 (SP)
RAM1O, LINKED TEST	-1006 (FLO) -9006 (LML) -7006 (LC)
RAM1O, TEST MODULE	-1003 (OBJ) -9003 (LIST) -2003 (SRC)
RAM1OMSG, MESSAGE MODULE	2250284-1003 (OBJ) -2003 (SRC) -9003 (LIST)

PROGRAMM DESCRIPTION

UNIVERSAL TILINE CONTROLLER MAG TAPE TEST

PN 2250126-9901

TABLE OF CONTENTS

1.0 SCOPE

This document describes the Model 979 Magnetic Tape Drive Diagnostic, TAPTST. This diagnostic was written to verify the correct operation of the Universal TILINE Controller (UTC) and the associated tape units when the UTC is microprogrammed to control the tape units. This diagnostic also has some unique utility routines which can aid the operator in troubleshooting the UTC and tape units.

2.0 REFERENCES

TITLE	PART NUMBER
SPECIFICATION, TILINE 979 MAG TAPE CONTROLLER	947551
MODEL 990 COMPUTER TILINE MAGNETIC TAPE CONTROLLER DEPOT MAINTENANCE MANUAL	946237-9701
MODEL 979A TAPE TRANSPORT DEPOT MAINTENANCE MANUAL	949613-9701
DIAGNOSTIC HANDBOOK	945400-9701

3.0 EQUIPMENT AND SOFTWARE REQUIRED

This section describes the minimum equipment requirements and the available object software for the diagnostic.

3.1 EQUIPMENT REQUIREMENTS

A) A Texas Instruments 990/10 computer with 12K of more of memory.

B) An appropriate interactive device.

C) A tape kit, Model 979 Tape Unit and Controller (PN 947578-0001).

D) An appropriate loading device.

3.2 SOFTWARE REQUIREMENTS

The Fully Linked Object is:

	TAPTST	PN 2250126-1006 (FLO)
LINKABLE PARTS:	MT10SV	PN 2250126-1003 (OBJ)
	MT10PD1	PN 2250551-1003 (OBJ)
	MT10PD2	PN 2250552-1003 (OBJ)
	MT10PD3	PN 2250553-1003 (OBJ)
	MT10MSG	PN 2250296-1003 (OBJ)

4.0 LOADING INFORMATION

Loading procedures from all available media are found in the Diagnostics Handbook.

5.0 TEST EXECUTION AND DESCRIPTION

The Model 979 Tape Test verifies the correct operation of the Universal TILINE Controller (UTC) and up to four tape drives under its control. TAPTST is composed of the following parts:

PART	FUNCTION
0	Controller Test
1	Rewind and Erase Test
2	Basic Write/Read Test and Memory Addressing Test
3	Write/Backspace/Read Test
4	Forward Creep Test
5	Even/Odd Write/Read Test with Skip
6	Special Movement Test
7	Creep and Write/Read Ramp Test
8	Random Data Check with Varying Length Records
9	Power Fail Test

5.1 PARTS

The following describes Part 0 through Part 9 of the diagnostic.

5.1.1 PART 0 CONTROLLER TEST

Part 0 is a quick test of the Universal TILINE Controller. A tape unit does not have to be present because no commands are issued to a tape unit. If errors occur during the running of this part, there is a problem with the controller and the rest of the test probably won't run correctly.

A. The test attempts to write a bit into each of the 8 low order bits of register 0. If any one of the bits can be set there is a controller malfunction and an error message is printed. The rewind mask interrupts are forced and verified. If a rewind interrupt cannot be forced an error message is printed.

B. Following the testing of register 0, registers 1-6 are exercised by clearing and reading, and setting and reading the contents of each register. If the data read from the register is not the same as the data placed in the register an error message will be printed.

C. The remaining register, number 7, is tested by setting each bit. After each bit is set and reset it is read to verify its state. If the state is not correct an error message is printed. Interrupts are also checked. If an unexpected interrupt occurs an error

message is printed. If an interrupt is expected but does not occur an error message is printed.

5.1.2 PART 1 REWIND AND ERASE TEST

Part 1 tests the ability of the controller and tape unit to perform the rewind and erase commands. The commands are issued in the following order:

a. Rewind

b. Erase maximum characters (FFFF hexadecimal)

c. Rewind

The only errors which are checked for are the successful completion of the three commands by the status checked subroutine.

5.1.3 PART 2 BASIC WRITE/READ TEST AND MEMORY ADDRESSING TEST

Part 2 tests the ability of the controller and tape unit to perform the basic writing and reading of data. Also tested is the ability of the TMTC, TILINE Magnetic Tape Controller, to address all available locations of memory. Listed below are the commands and the order in which they are issued for the write/read portion of Test 2, Part 2A.

a. Rewind

b. Write a record of 256 characters

c. Rewind

d. Read a record of 256 characters

e. Compare the data

The data read in is located in a different buffer area than the data written. These two blocks of data are compared on a character basis. If there are any differences between the blocks of data an error message is printed for each use.

Listed below are the commands issued for the Memory Addressing portion of Test 2, Part 2B and 2C.

a. Erase a length of tape

b. Write data

c. Backspace

d. Read and compare data

e. Erase 1 block

f. Write data

g. Backspace

h. Read and compare data

Part 2B of the test requires that there be at least 4K of memory beyond the end of the diagnostic. If there is not 4K of memory available the following message will be printed:

INSUFFICENT MEMORY

and the test will end. If 4K of memory is available the diagnostic tests that memory by writing the address as data to the address in a buffer.

EXAMPLE Location 6F60 holds the data 6F60

The data is written to tape, the buffer is cleared and the data read back from tape to the buffer. The contents of each location is then compared to the address of its location to ensure that there has been

no memory addressing errors. If any errors are indicated an error message is printed.

Part 2C of test 2 requires that the computer have the Mapped Memory Option. If the Mapped Memory Option is not present then the following message is printed:

PART 2C SKIPPED

and the test ends. With the Mapped Memory Option present the test performs the same sequence of commands as in Part 2B except that the Map Enable switch has been set and the locations being tested are in the mapped memory. This will continue until the end of the mapped memory. If memory comparison errors are found an error message will be printed on the interactive device.

5.1.4 PART 3 WRITE/BACKSPACE/READ TEST

Part 3 of the diagnostic incorporates the backspace command in the basic write/read test. Listed below are the commands and the order in which they are issued in Part 3.

a. Write a record of 256 characters

b. Backspace 1 record

c. Read a record of 256 characters

d. Compare the data

Before the record of 256 characters is written the output and input buffers are initialized to insure that the same data is not used as in Part 2. These two blocks are compared on a character basis. If there are any differences between the blocks of data an error message is printed for each case.

5.1.5 PART 4 FORWARD CREEP TEST

Part 4 of the diagnostic issues the same commands to the tape controller as Part 3 does however, two consecutive write commands are issued. Listed below are the commands and the order in which they are issued in Part 4.

a. Write a record of 256 characters

b. Write a record of 256 characters

c. Backspace 1 record

d. Read a record of 256 characters

e. Compare the data

Before data is written on the tape the output and input buffer areas are initialized with data different than that used in Parts 2 and 3.

These two blocks of data are compared on a character basis. If any miscompares occur between the two blocks this is an indication of forward tape creep. An error message is printed for each occurrance.

5.1.6 PART 5 EVEN/ODD WRITE/READ TEST WITH SKIP

Part 5 performs the most extensive write/read operation in all the parts of the diagnostic. This part is executed in subsections. They are discussed below.

A. This section writes and reads records of various lengths and verifies the data. Listed below are the commands issued in this section.

a. Rewind

b. Write 16 records with the various lengths shown below

c. Rewind

d. Read a record with one of the lengths shown below

e. Compare the data

f. Repeat steps d and e until done

Record lengths: 16,17,32,33,64,65,128,129,512, 513,1024,1025,2048,2049,3096,3097

This section is to verify that long, short, even, and odd character length records can be written and read.

B. This section of Part 5 verifies that partial records can be read correctly.

The record that all the read operations are performed on is the last record written in section A (3097 characters). After each read operation is performed the data read is compared to the data written. If a miscompare is found an error message is printed for each occurrance.

Also the overflow count in registers 1 and 2 of the UTC are examined for the correct count. If it is not correct an error message is printed.

Case 3 is used to verify the capability of the UTC to fill with FF hexadecimal when an odd number of characters are transferred from the tape. If the UTC does not fill with FF an error message is printed.

5.1.7 PART 6 SPECIAL MOVEMENT TEST

Part 6 of the diagnostic is broken into 5 sections. Before any of the sections of Part 6 are executed the tape is moved forward by executing 4 erase commands with a maximum character count. This is done to get to a less worn position on the tape.

A. This section writes and reads data of the following character lengths: 16, 17, 32, 33, 64, 65, 128, 129, 256, 257, 512, 513, 1024, 1025, 2048, 2049, 3095, 3097.

Although the writing and reading of these lengths has been tested elsewhere in the test the data used here is unique. All possible data character patterns are tested. The data used in the section is shown partially below:

00,FF,01,FE,02,FD....

This checks the ability of the UTC and tape handler to write and read all possible data characters. After the data is read off of tape it is compared on a character basis to the data written. If there are miscompares an error message is printed.

B. This section of Part 6 checks the special flag bit (8) of register 6. In this section a record of 80

hexadecimal characters is written to tape with bit 8 of register 6 turned on. This should cause a VRC error and bit 8 of the status register should be set. If it is not set an error message is printed.

If this error occurs the test goes on to section C of Part 6. However if the VRC bit got turned on, the record that was written is read back. This should cause the VRC, LRC, CRC, and Format error bits to be set. In they are not set then an error message is printed.

****NOTE: Reference Specification for 979 TILINE Mag Tape Controller (PN 947551) Sec. 3.2.1.1.8 for explanation of above error bits.

C. Section C of Part 6 verifies that no excess data is read from tape when the data length specified in the issued command is greater than the length of the record. This is done for both even and odd length records. If more data is transferred an error message is printed.

D. Section D of Part 6 verifies the approximate amount of data that an erase of zero length would destroy. Before this section is run a flag is examined to see if this is a tape unit that is operating at 1600 bpi. If

it is this section is skipped. However, if an 800 bpi (NRZI) handler is being used the following command sequence is issued:

a. Write a record of 1600 characters
b. Write a record of 1600 characters
c. Backspace 1 record
d. Erase 0 length
e. Backspace 1 record
f. Skip forward 1 record
g. Read the second record

The remaining number of characters are checked to see if more than 900 and less than 1300 characters are remaining. If not the erase command destroyed too much or too little data and an error message is printed.

E. Section E of Part 6 verifies that a write of 0 length does not transfer data to tape. To do this the following command sequence is issued:

a. Write 2 records of 800 characters
b. Write a record of 0 characters
c. Write 2 records of 800 characters
d. Backspace 4 records
e. Read 4 records of 800 characters and verify data

If the data read is not the same as the data written in each of the 4 records an error message is printed.

5.1.8 PART 7 CREEP AND WRITE/READ RAMP TEST

Part 7 of the diagnostic is broken into 3 sections. The tape is moved forward by executing 4 erase commands with a maximum character count before any part of the test is executed.

A. This section verifies that write commands can be issued at varying time intervals between issuances. This gives the tape varying time to rest or slow down. This test is performed by issuing 16 write commands starting with approximately one millisecond wait period before the next command is issued. After each successive command, another millisecond is added to the timer controlling the wait period. After all writes have been performed the data is read back and compared for correctness. If the data does not compare on any record an error message is printed.

B. Section B in Part 7 verifies that read commands can be issued at varying time intervals between issuances. This gives the tape varying time intervals to rest or slow down. This test is performed by issuing the following sequence of commands:

a. Write 20 records of 20 characters

b. Backspace 20 records

c. Read a record of 20 characters

d. Check status register for operation complete

e. Wait

f. Increase wait period by approximately 1 millisecond and repeat steps c-f 19 times

If the operation complete bit in R7 does not get set after any read command an error message is printed.

C. Section C of Part 7 verifies that the tape handler does creep in the forward direction if a number of write/backspace/read operations are performed. This capability of the tape unit is used if there is a bad spot on the tape during a write operation. The record being written is actually moved down the tape. This test is performed by issuing the command sequence listed below:

a. Write 2 records of 800 characters

b. Erase 1472 characters

c. Write a record of 800 characters

d. Backspace 2 records

e. Write record 2

f. Read third record

g. Compare data

h. Backspace 3 records

i. Read first record

j. Compare data

k. Skip forward 1 record

l. Backspace 1 record

m. Repeat steps e-m until part of record 3
   or part of record 1 have been destroyed
   of there is no creep after 20 times through
   the loop

n. Repeat steps a-m five times saving the
   creep count after each creep test

The five attempts are performed to check any variance the tape unit may have. If the 'PRINT TAPE CREEP DATA'? question asked during initialization was answered with a 1 the following message is printed. This is not an error message.

CREEP COUNTS IN HEX

TRY	1	2	3	4	5
	XX	XX	XX	XX	XX

If the data does not compare in step g the third record got destroyed. If after the test the CREEP COUNTS vary greatly between attempts an error message is printed.

If the data does not get destroyed in part g or J in 20 iterations of the loop in any of the five attempts the tape unit has no creep and an error message is printed.

If the data does not compare in step J in any of the five attempts the first record got destroyed which indicates negative creep and is signified by an error message being printed.

The remaining check is to verify that all creep attempts fell within predefined tolerances. These tolerances vary depending on the type of tape unit being used (NRZI of PE). If any of the creep attempts do not fall within the tolerances an error message is printed.

It is suggested that the test be initialized such that the creep counts are not printed. If an error is detected the system should be reinitialized such that

the creep counts will be printed. These counts should give an indication to the type of problem.

5.1.9 PART 8 RANDOM DATA CHECK WITH VARYING LENGTH RECORDS

Part 8 verifies the capability of the UTC and tape unit to write and read random data of varying lengths. To perform this the following command sequence is issued.

a. Generate random data

b. Write random data

c. Backspace 1 record

d. Read random data

e. Compare data

The above 5 step sequence is done with the following record lengths: 8, 9, 10, 11, 12, 13, 14, 15, 16, 31, 32, 63, 64, 127, 128, 255, 256, 511, 512, 1023, 1024, 2047, 2048.

If there are any miscompares in step e an error message is printed.

After the message is printed the command sequence is repeated until all record lengths have been processed. Following this an End of File (EOF) test is performed.

An EOF is written and verified in four different ways. First a skip reverse 1 command is issued. If the EOF bit does not come on in the Tape Status Word (0) an error message is printed.

Next a read of 80 characters is attempted. If the EOF bit fails to come on in the Tape Status Word an error message is printed.

Following the read attempt a skip reverse command is issued. If the EOF bit fails to come on in the Tape Status Word an error message is printed.

Register 4 is also interrogated to verify it has the count of FFFF. If it does not, an error message is printed.

The final EOF check is done by attempting to do a skip forward command. If the EOF bit fails to come on in the Tape Status Word an error message is printed

5.1.10 PART 9 POWER FAIL TEST

Part 9 verifies that a failure of the power supply when the system has battery support will not cause data to be written on the tape.
Part 9 requires operator intervention so there are no verbs to loop on Part 9.

***CAUTION: DO NOT USE THIS PART IF THE COMPUTER DOES NOT HAVE BATTERY SUPPORT.

When Part 9 starts executing it asks the operator if

the system has battery support. If the operator should respond with a 0 (indicating no battery support) a message is printed telling the operator not to run this part of the test. However, if the operator enters a 1 (indicating battery support) the operator is asked if system power has been cycled.

Example: HAS SYSTEM POWER BEEN CYCLED?-

The operator should respond with a 0 the first time he encounters this question indicating that power has not been cycled. After seeing the zero, two records will be written and the head will be positioned between the two records. Next a message is printed telling the operator to cycle system power and control returns to the verb decoder.

Example: CYCLE SYSTEM POWER

At this time the operator should cycle system power. Of course when the system is powered back on the series of system initialization questions must be answered. After the initialization questions are answered the operator should select the E9 verb to finish the test. The operator should answer affirmative to the battery support and system power questions. At this time the test will verify, by reading and comparing data, that

the previously written records on both sides of the head did not get destroyed when system power was cycled. If record 1 or 2 are destroyed the appropriate error message is printed.

5.2 OPERATOR INTERFACE

TAPTST runs under control of DOCS. Refer to the section of DOCS in the Diagnostic Handbook for a detailed description of Docs.

5.2.1 AVAILABLE VERBS

In addition to the verbs supplied by the Docs package, TAPTST also provides the verbs listed in Table 1 and in Table 2.

TABLE 1

VERB	FUNCTION
IT	Initialize Test
E0	Execute Part 0
.	.
.	.
.	.
E9	Execute Part 9
EA	Execute Parts 0-8
L1	Loop on Part 1
.	.
.	.
.	.
L8	Loop on Part 8
(NOTE: There is no loop verb for Part 9)	
LA	Loop on Parts 0-8
CM	Execute Parts 0-8 for all available units and loop

5.2.1.1 SPECIAL UTILITY VERBS

The following verbs have been included in the UTC and Tape Test to aid in troubleshooting diagnostic problems.

TABLE 2

VERB	FUNCTION
IC	Issue Command
IM	Issue Multiple Commands
LO	Loop on Multiple Commands
DC	Display the UTC Status
CD	Compare Two Blocks of Data

5.2.2 IT VERB

The IT Verb is used to initialize the following tape parameters:

```
ENTER TILINE BASE ADDRESS, DEFAULT = F880-
TAPE UNIT TO BE TESTED -(1,2,3,4)
OTHER AVAILABLE UNITS-TERMINATE WITH FF
ENTER INTERRUPT LEVEL IN HEX, DEFAULT = 09
ENTER 0 FOR INTERRUPTS AND 1 FOR IDLE SCAN
PRINT TAPE CREEP DATA ?-
```

To set up the list of available units enter the number of the unit 1-4. The default for the second unit is FF. This allows the operator to test one unit by entering the unit to be tested and depressing the space bar for other available units. If other units (up to 3) are available they may be shown. The list is terminated with FF. This lists is used and modified by the CM

Verb.

5.2.2.1 E0-E9 VERBS

Use a verb E0-E9 to execute a specific part of the diagnostic. If the part completes and if the operator typed a 1 to the initialization question "PRINT ON ERRORS?", the number of Timing,Data, and Status errors that occurred during the test will be printed.

5.2.2.2 L0-L8 VERBS

Use the L0-L8 Verbs to loop on part 0-8 (one part per verb), of the tape test. The "L" verbs will loop until the operator terminates the execution by the entry of a keyboard character on the interactive device. After each time the loop has been completed the loop count will be printed.

5.2.2.3 EA VERB

Use the EA Verb to execute Part 0 through Part 8 of the tape test in numerical order.

5.2.2.4 LA VERB

Use the LA Verb to loop on Parts 0-8 in numerical order. After each time the loop has been completed the loop count will be printed. To terminate this verb

enter a character on the interactive device.

5.2.2.5 CM VERB

The CM Verb can be used to execute Parts 0-8 with each tape unit as the unit to be tested.

It does this by executing Parts 0 thru 8 on one unit, then it takes the next available unit as selected by the IT Verb and makes it the unit to test.

This verb will run until interrupted and each time it finishes Parts 0 thru 8 it will shift the next avialable unit to be the unit under test.

Before each unit is tested the unit number is printed out in a message.

After all units have been tested the test will start over again with the first unit.

5.3 UTILITY VERBS

The IC, IM, and LO Verbs allow the operator to execute a command or a series of commands that he has built in memory. The commands are issued to the UTC by calling the Command Issuer Subroutine (CISUER).

5.3.1 IC VERB

The IC Verb can be used to issue a command to the UTC from the 990 memory address input by the operator. The command is 8 words long and formatted as below:

FORMAT: IC ADDR-XXXX

Where : XXXX is the address of an 8 word long command

5.3.2 IM VERB

The IM Verb can be used to issue multiple commands to the UTC from the 990 memory.

FORMAT: IM ADDR-XXXX #COMMANDS-YY

Where : XXXX is the address and YY = number of commands

5.3.3 LO VERB

The LO Verb can be used to loop on a sequence of UTC commands. The commands must each be 8 words long and must be in sequential memory.

FORMAT: LO ADDR-XXXX #COMMANDS-YY CHECK ST?-Z

Where: XXXX = the address of the commands

YY = the number of commands

Z = check the status checker flag (y/n)

5.3.4 DC VERB

The DC Verb can be used to print out or display the

status of the UTC. It does this by reading the UTC slave registers, then formats and outputs the contents to the interactive device.

FORMAT: DC

5.3.5 CD VERB

The CD Verb can be used to compare two blocks of data. This verb compares the two blocks of data starting at the two addresses input by the operator for the number of words input. If any miscompares are found, the two addresses and their respective data are printed. The verb then goes on until it is done or another miscompare is found.

FORMAT: CD ADDR1-XXXX ADDR2-YYYY # OF WORDS-ZZ

WHERE: XXXX=ADDRESS OF FIRST BLOCK

YYYY=ADDRESS OF SECOND BLOCK

ZZ=NUMBER OF WORDS IN EACH BLOCK

5.4 STANDARD DOCS INITIALIZATION

Reference Diagnostic Handbook.

5.5 PROGRAM PANEL DOCS INITIALIZATION

When running TAPTST in the Front Panel Mode, besides asking the question 'IDLE ON ERRORS?', the IT Verb questions are also asked. Reference the Diagnostic Handbook for further information.

6.0 ERROR MESSAGES

ERROR NO. MESSAGE AND CONDITION

>1-3 MESSAGE: CONTROLLER FAILURE

CONDITION: Controller malfunction. Test was able to write a bit into one of the 8 low order bits of register 0. The test should not be able to set any of these bits.

>4-7 MESSAGE: ERROR X REWIND INTERRUPT TEST

CONDITION: Diagnostic test was unable to force a rewind interrupt.

>9 MESSAGE: WRONG LEVEL INTERRUPT

CONDITION: An incorrect interrupt level was entered during the answering of the Initialization questions. Reenter the correct interrupt level.

>8,A MESSAGE: ATTEMPT TO WRITE XXXX IN XXXX RESULTED IN XXXX BEING RETURNED

CONDITION: While testing registers 1-6, the data read from one of the registers is not the same as the data placed in the register.

>B MESSAGE: ADDR1=XXXX COUNT=XXXX ADDR2=XXXX COUNT=XXXXX

CONDITION: A miscompare has occurred between a block written to memory and that same block read from tape to memory.

>C MESSAGE: COMPARE ERROR-CHAR WRITTEN=XXXX CHAR READ=XXXX

CONDITION: A miscompare has occurred between a character written and the character read during execution of test 5.

>D MESSAGE: DID NOT FILL LAST CHAR WITH -FF-

CONDITION: The Universal TILINE Controller (UTC) failed to fill an area with FF hexadecimal after an odd number of characters were transferred from tape.

>E MESSAGE: TO MANY CHARACTERS WERE TRANSFERRED TO MEMORY

CONDITION: During test 5 the number of characters transferred from tape to memory from a variable length record was greater than the number expected.

>F MESSAGE: OFFSET DID NOT DECREMENT TO ZERO

CONDITION: Register used for compare loop control count in test 5 did not decrement to 0.

>10 MESSAGE: OVERFLOW COUNT WAS 000000 BUT SHOULD HAVE BEEN 00 00 00

CONDITION: The overflow count in registers 1 and 2 of the Universal TILINE Controller (UTC) were examined and did not contain the correct count.

>11 MESSAGE: ERROR PART 6A

XXXX MISCOMPARES WITH RECORD LENGTH =XXXX

CONDITION: Data read off of tape did not compare to that residing in memory.

>12 MESSAGE: ERROR PART 6B

RECEIVED VRC ERROR IN WRITE OPERATION

CONDITION: Bit 8 of Device Register 7 did not set after a record of 80 hexadecimal characters was written to tape while the controller was generating even vertical parity instead of odd.

>13 MESSAGE: ERROR PART 6B

EXPECTED R7 ERROR PATTERN=XXXX RECEIVED=XXXX

CONDITION: Bits 8,9,10, and 14 of Device Register 7 did not set after a record of 80 hexadecimal characters was read from tape. Characters were written to tape while the controller was generating even vertical parity instead of odd and should have set these bits when record was read.

>14 MESSAGE: ERROR PART 6C

WROTE RECORD OF XXXX CHARACTERS

ATTEMPTED TO READ 160 CHARACTERS

DATA TRANSFER DID NOT STOP AFTER XXXX CHARACTERS

CONDITION: More data was transferred than actually existed in a record during a read from tape when the data length specified in the issued command is greater then the length of that record.

>15 MESSAGE: ERROR PART 6D

AN ERASE OF 0 LENGTH DESTROYED LESS THAN

300 CHAR OF A 1600 CHAR RECORD

CONDITION: Not enough characters were destroyed during an erase of 0 length. At least 300 characters should have been destroyed.

>16 MESSAGE: ERROR PART 6D

AN ERASE OF 0 LENGTH DESTROYED MORE THAN

700 CHAR OF A 1600 CHAR RECORD

CONDITION: Too many characters were destroyed during an erase of 0 lenght. At most 700 characters should have been destroyed.

>17 MESSAGE: ERROR PART 6E

A WRITE OF 0 LENGTH TRANSFERRED DATA TO TAPE

CONDITION: During a write of 0 length data was transferred to tape. No data should have been transferred to tape.

>18 MESSAGE: ERROR PART 7A

FAILED TO READ RECORD XXXX

CONDITION: Data read back from a tape which was written using write commands of varying time intervals failed to compare to the original data.

>19 MESSAGE: ERROR PART 7B

FAILED TO READ FOLLOWING RECORDS WITHOUT ERROR

RECORD NO XX

CONDITION: The Operation Complete Bit, Bit 1, of Device Register 7 did not set set after a read command.

>1A MESSAGE: ERROR PART 7C

NO CREEP IN 20 BACKSPACE/WRITE OPERATIONS

CONDITION: No forward tape creep found after 20 iterations of the loop in any one of the five attempt sets.

>1B MESSAGE: ERROR PART 7C NEGATIVE CREEP

CONDITION: The data did not compare after 5 attempts indicating that the first record got destroyed and that there is negative creep.

>1C MESSAGE: E8 DATA ERROR

XXXX MISCOMPARES WITH DATA LENGTH=XXXX

CONDITION: Data written did not compare with data read.

>1D MESSAGE: PART 9 RECORD 1 ERROR

DATA WRITTEN=XX DATA READ=XX

CONDITION: Portion of record 1 was destroyed after power was cycled.

>1E MESSAGE: PART 9 RECORD 2 ERROR

DATA WRITTEN=XX DATA READ=XX

CONDITION: A portion of record 2 was written over after the system power was cycled.

>1F MESSAGE: INVALID CREEP COUNT

CONDITION: One of the creep counts did not fall within any of the predefined tolerances.

>20 MESSAGE: VARIANCE TOO LARGE BETWEEN CREEP COUNTS

CONDITION: The differences between any of the 5 creep counts are erratic and vary too greatly.

>21 MESSAGE: ERROR PART 8

END OF FILE ERROR BIT FAILED TO COME ON AFTER PERFORMING A SKIP REVERSE 1

CONDITION: The EOF Bit, Bit3, of Device Register 0 was not set after a skip reverse of 1 record.

>22 MESSAGE: ERROR PART 8

END OF FILE ERROR BIT FAILED TO COME ON AFTER PERFORMING A READ FORWARD 80 CHARACTERS

CONDITION: The EOF Bit, Bit 3, of Device Register 0 was not set after a read forward of 80 characters.

>23 MESSAGE: ERROR PART 8

END OF FILE ERROR BIT FAILED TO COME ON AFTER PERFORMING A SKIP REVERSE 0

CONDITION: The EOF Bit, Bit 3, of Device Register 0 was not set after a skip reverse of 0

records.

>24 MESSAGE: ERROR PART 8

END OF FILE ERROR BIT FAILED TO COME ON AFTER PERFORMING A SKIP FORWARD 1

CONDITION: The EOF Bit, Bit3, of Device Register 0 was not set after a skip forward of 1 record.

>25 MESSAGE: REG 4 ERROR EXP=FFFF REC=XXXX

CONDITION: Register 4 does not contain the expected count of FFFF.

>26 MESSAGE: DID NOT GET EXPECTED INT WHEN TESTING R7

CONDITION: An expected interrupt did not occur while testing R7.

>27 MESSAGE: GOT FALSE INT WHEN TESTING R7

CONDITION: An unexpected interrupt was received while testing R7.

>28 MESSAGE: DID NOT RECEIVE R7 INTERRUPT IN 17 SEC

CONDITION: An expected interrupt was not received within 17 seconds after a command was issued to the (UTC) by the Command Issuer Subroutine (C.I.S.) while Device Register 7, Bit 3 is set.

>29 MESSAGE: BOT NOT REACHED ON REWIND IN 4 MIN WITH INT

CONDITION: Bot (Beginning of Tape) was not reached within 4 minutes after the Commnand Issuer Subroutine (C.I.S.) issues a Rewind or Rewind and Unload command to the UTC.

>2A MESSAGE: OFF LINE BIT DID NOT COME ON IN RO DURING REWIND AND UNLOAD IN 4 MIN WITH INT

CONDITION: With the Interrupt Enable switch, Device Register 7 Bit 3, set to a 1 the Off Line Bit, Device Register 0 Bit 0, did not set to a 1 within 4 minutes of a Rewind and Unload command.

>2B MESSAGE: IDLE BIT FAILED TO COME ON IN 17 SEC W/O INT

CONDITION: The Idle Bit, Device Register 7 Bit 0, failed to come on within 17 seconds after the C.I.S. issued a command to the UTC while the Interrupt Bit, Device Register 7 Bit 3, was not set.

>2C MESSAGE: ** STATUS ERROR **

TMTC REGS

TAPE STAT COM STAT/CT OFFSET NO CHAR MEM AD UNIT R7 STAT

0010 06 000000 0000 0000 001EDC 02 A880

TAPE STAT COM STAT/CT OFFSET NO CHAR MEM AD UNIT R7 STAT

0000 06 000000 0000 0100 003CB8 02 1000

CONDITION: The Operation Complete Bit, Device Register 7 Bit 1, in the UTC failed to set after the C.I.S. issued a command while the status check flag and software error message flag were set.

>2D MESSAGE: *** END OF TAPE ***

OPERATOR MUST START TEST OVER

CONDITION: After an error occurred Bit 4 of Register 0, the End of Tape Bit, was found to have been set.

>2E MESSAGE: FAILED TO GET TILINE TIMEOUT WHEN WRITING TO A NON-EXISTENT MEMORY ADDRESS (1FFBFE)

CONDITION: The TILINE Timeout Error Bit, Register 7 Bit 13, failed to set after an attempted write command to a non-existent memory address.

>2F MESSAGE: DID NOT REACH BOT ON REWIND IN 4 MIN W/O INT

CONDITION: With the Interrupt Enable switch, Device Register 7 Bit 3, reset to a 0 the BOT, Beginning of Tape, bit, Device Register 0 Bit 1, did not set to a 1 within 4 minutes of a Rewind command.

>30 MESSAGE: OFF LINE BIT DID NOT COME ON IN R0 DURING REWIND AND UNLOAD IN 4 MIN W/O INT

CONDITION: With the Interrupt Enable switch, Device Register 7 Bit 3, reset to a 0 the Off Line Bit, Device Register 0 Bit 0, did not set to a 1 within 4 minutes after a Rewind and Unload command was issued.

>31 MESSAGE: TAPE UNIT ENABLED BUT DID NOT GET TAPE INT

CONDITION: Received an unexpected interrupt at the location indicated by additional error message. Device Register 7 Bit 3, Interrupt Enable, was set but tape unit did not cause the unexpected interrupt.

>32 MESSAGE: PART 2B DATA NOT EQUAL TO ADDRESS
ADDRESS=XXXX
DATA=XXXX

CONDITION: Memory was not able to correctly store a piece of data written to it from tape.

>33 MESSAGE: COMPARISON ERROR IN MAP MEMORY
MAP ADDRESS=XXXX
MAP BIAS=XXXX
TILINE ADDRESS=XXXXXX
EXPECTED DATA=XXXX
DATA READ=XXXX

CONDITION: Mapped memory was not able to correctly store a piece of data written to it from tape.

7.0 PART NUMBERS

PROGRAM DESCRIPTION	2250126-9901 (PD)
SRC, PD ROFF SOURCE	-2001 (SRC)
LIST, PD ROFF LISTING	-9001 (LIST)
FICHE KIT	-0009 (SP)
TAPTST, LINKED TEST	-1006 (FLO)
	-9006 (LML)
	-7006 (LC)
MT10SV, SPECIAL VERB MODULE	-1003 (OBJ)
	-9003 (LIST)
	-2003 (SRC)
MT10PD1, TEST MODULE 1	2250551-1003 (OBJ)
	-9003 (LIST)
	-2003 (SRC)
MT10PD2, TEST MODULE 2	2250552-1003 (OBJ)
	-9003 (LIST)
	-2003 (SRC)
MT10PD3, TEST MODULE 3	2250553-1003 (OBJ)
	-9003 (LIST)
	-2003 (SRC)
MT10MSG, MESSAGE MODULE	2250296-1003 (OBJ)
	-9003 (LIST)
	-2003 (SRC)

PROGRAM DESCRIPTION

FOR THE

733 ASR/KSR DIAGNOSTIC

P/N 02250250-9901

REVISION: *A 03/79

TABLE OF CONTENTS

1.0 SCOPE

This document describes the TST733 Diagnostic (P/N 2250250) and its capabilities. The TST733 Diagnostic tests the keyboard and printer on a KSR and the keyboard, printer and cassette tape units on an ASR. TST733 was written primarily for Model 733 ASR or KSR, but it may also be used to test the printer and keyboard on Models 742, 743, 745, 763, and 765.

2.0 REFERENCES

For information beyond the scope of this document refer to the following:

TITLE	PART NUMBER
Diagnostic Handbook	945400-9701
Model 733 ASR/KSR Data Terminal Installation and Operation Manual	945259-9701
Model 733 ASR/KSR Data Terminal Operating Instructions	959227-9701
Models 732/733 ASR/KSR Maintenance Manual	960129-9701
TTY/EIA Interface Module Depot Maintenace Manual	945408-9701
Model 990/10 Computer System Hardware Reference Manual	945417-9701
Model 990/4 Computer System Hardware Reference Manual	945251-9701
Model 990/5 Computer Hardware User's Manual	946294-9701

3.0 EQUIPMENT AND SOFTWARE REQUIREMENTS

This section describes the minimum equipment requirements and the available linked object software.

3.1 HARDWARE REQUIREMENTS

In addition to the equipment specified in the diagnostic handbook, the following equipment is required.

Model 733 KSR Data Terminal Kit 945161-0001

or

Model 733 ASR Data Terminal Kit 945162-0001

A 990 computer with 12K words of memory.

3.2 SOFTWARE REQUIREMENTS

The TST733 Diagnostic runs on the 990 Family Computers and must be run under control of DOCS. The following describes the object modules that must be linked to form the TST733 Diagnostic.

TST733 -- Main Module
TPK733 -- Printer and Cassette Tests
KEY733 -- Keyboard Test
SUB733 -- Subroutines
IOR733 -- Input/Output Routines
MSG733 -- Messages

The fully linked test module is TST733, 2250250-1006 (FLO).

4.0 LOADING

Loading procedures from all available media are found in the Diagnostic Handbook.

5.0 TEST EXECUTION AND DESCRIPTION

The TST733 Diagnostic is an Interactive Test. When the diagnostic has been loaded the following message will be output on the specified interactive device.

TST733 733 ASR/KSR DIAGNOSTIC VERSION # 03/79 *A

5.1 IT Verb -- Initialize Test

The test must be initialized before executing the test. The 'IT' verb is used to perform this initialization. When the 'IT' verb is executed, it will ask the operator the questions that follow. If the default is correct, the operator need only press the return key. If a different value is required, the operator must enter that value and then press the return key.

```
ENTER 733 INTERFACE CRU BASE, DEFAULT = 0000
ENTER 733 INTERFACE INTERRUPT LEVEL,  DEFAULT = 06
DO YOU WANT TO RUN THE INTERRUPT TEST  (0=NO, 1=YES)  DEFAULT = 1
ENTER LINE FREQUENCY (0=50HZ, 1=60HZ) DEFAULT =  1
ENTER UNIT UNDER TEST TYPE (0=ASR, 1=KSR), DEFAULT = 0
```

If the operator reply to the last question indicates that

the unit under test is an ASR, then the following additional operator prompt will be output.

INSERT 2 SCRATCH CASSETTES

During execution of the 'IT' verb, the interface for the unit under test is initialized. The interface may be a TTY/EIA card or one of the TMS 9902/9903 communication ports on the 990/5.

5.2 EA Verb -- Execute All Tests

Entering this verb causes all the tests included in the TST733 Diagnostic to be executed one after the other in order beginning with TEST 01 and ending with TEST 06. If the unit under test is a KSR then TEST 03 and TEST 06 will be skipped as both of these tests require use of the cassette units on an ASR.

5.3 LA Verb -- Loop on All Tests

Entering this verb causes all the tests included in the TST733 Diagnostic to be executed one after the other in order beginning with TEST 01 and ending with TEST 06. Upon completion of TEST 06 the number of times through the loop is output to the front panel and execution of TEST 01 begins again. The tests continue executing until the operator presses the at (@) key, the CRT 911 CMD key, or the CRT 913

HELP key.

5.4 IC Verb -- Issue Command to the ASR

This verb allows the operator to issue ASR commands to the unit under test. It is designed primarily for the operator who is trouble-shooting or repairing a defective ASR. If the unit under test is a KSR, this verb will not be executed and control is returned to 'VERB ?'. When this verb is entered on an ASR, the following is output on the interactive device.

```
0 -- REQUEST STATUS          6 -- RECORD MODE # 2
1 -- REWIND # 1              7 -- BLOCK FORWARD
2 -- REWIND # 2              8 -- BLOCK REVERSE
3 -- LOAD # 1                9 -- ADC ON
4 -- LOAD # 2                A -- ADC OFF
5 -- RECORD MODE # 1

ENTER NUMBER OF COMMAND -
```

The operator would then enter the number of the command that he wishes to have issued to the ASR unit.

If the operator enters '0' (REQUEST STATUS), the request status command is issued to the ASR and the status byte is retrieved and output to the operator in the following format.

A S R S T A T U S

PRNTR READY	RCRD READY	BOE02	BOE01	PLYBK ERROR	PLYBK READY
X	X	X	X	X	X

If the operator enters '1' (REWIND # 1), the command rewind cassette number one is issued to the ASR, and the cassette tape in cassette drive number one should rewind to the beginning of the tape.

If the operator enters '2' (REWIND # 2), the command rewind cassette number two is issued to the ASR, and the cassette tape in cassette drive number two should rewind to the beginning of the tape.

If the operator enters '3' (LOAD # 1), the command load cassette number one is issued to the ASR. If the tape is at the beginning, the tape will be moved forward to the load point on the tape. If the tape is not at the beginning, a fast forward to the end of the tape will commense.

If the operator enters '4' (LOAD # 2), the command load cassette number two is issued to the ASR. If the tape is at the beginning, the tape will be moved forward to the load point on the tape. If the tape is not at the beginning, a fast forward to the end of the tape will commense.

If the operator enters '5' (RECORD MODE # 1), the command

set cassette number one in record mode and cassette number two in playback mode is issued to the ASR.

If the operator enters '6' (RECORD MODE # 2), the command set cassette number two in record mode and cassette number one in playback mode is issued to the ASR.

If the operator enters '7' (BLOCK FORWARD), the command block forward is issued to the ASR. This causes the cassette tape in the cassette drive which is currently in playback mode to move forward one block. The block of data is input into the memory location at the end of the loaded test. The input block of data may then be examined by the operator by using the '.DM' verb in the MAXI version of DOCS.

If the operator enters '8' (BLOCK REVERSE), the command block reverse is issued to the ASR, and the cassette tape in the cassette drive which is currently in playback mode will move back one block.

If the operator enters '9' (ADC ON), the command automatic device control on is issued to the ASR.

If the operator enters 'A' (ADC OFF), the command automatic device control off is issued to the ASR.

5.5 E1 -- Execute Test 01 (Interface Test)

The interface test determines the baud rate of the interface for unit under test. It expects to find a baud rate of either 300 baud for KSR's or 1200 baud for ASR's. If the baud rate is not 300 or 1200 baud then an error message is output. Otherwise the determined baud rate is output. The determined baud rate is then checked against what the operator input as the unit under test type. If the baud rate and the unit under test type do not correspond then an error message is output. This test should always be the first test run on the unit under test.

5.6 E2 -- Execute Test 02 (Dot Matrix and Printhead Test)

This tests the printhead on the unit. First, it prints one line containing 80 complete blocks of dots. Each block should have every dot in the 5 by 7 dot matrix clearly visable. This is done by printing 80 '#' characters followed by a carriage return followed by printing 80 'H' characters followed by a carriage return followed by 80 'I' characters. By using this sequence only the center dot if printed twice.

Second, it prints one line containing alternate spaces and complete blocks. This is done by printing the following sequence 40 times: space, '#', backspace, backspace, space,

'H', backspace, backspace, space, 'I'.

Third, it prints 10 lines each containing 8 blocks followed by a line feed. This is done by printing the following sequence 10 times: 10 '#', 10 backspaces, 10 'H', 10 backspaces, 10 'I', line feed. The printed output has a staircase appearance.

This tests to insure that the dot matrix is clear and complete and that the motion of the printhead maintains proper alignment.

5.7 E3 -- Execute Test 03 (Cassette Test)

This verb tests the following functions of the cassette drives:

1. Rewinding
2. Loading
3. Writing to a tape
4. Fast forward to the end of a tape
5. Fast reverse to the beginning of a tape
6. Reading from a tape
7. Block reverse capability
8. Block forward capability

5.8 E4 -- Execute Test 04 (Ripple Print Test)

Execution of this verb causes the Ripple Print pattern to be printed on the unit under test. This test is used to insure that the printer will print all of the characters of the character set.

5.9 E5 -- Execute Test 05 (Interrupt and Keyboard Test)

The first section of this test checks to see that the interrupts are generated when Data Set Ready is changed, and that the interrupts relative to keyboard and cassette input and printer and cassette output are generated properly.

The second section tests the keyboard to insure that all the keys generate the appropiate ASCII codes in all modes. The operator is prompted to enter the keys in a specified order.

If the operator specified 'NO' in the initialization question 'DO YOU WANT TO RUN THE INTERRUPT TEST', the first section of this test will be skipped. The test requires the following operator input:

```
ENTER THE TYPE KEYBOARD ON UNIT
     0 -- 733 STANDARD ASCII (NO UPPERCASE LOCK KEY)
     1 -- 733 FULL ASCII WITH UPPERCASE LOCK KEY
     2 -- 733 LIMITED ASCII WITH CALCULATOR CLUSTER
     3 -- OTHER KEYBOARD (KSR743, KSR745, KSR763, KSR765, KSR742)
DEFAULT =  0  -
```

If the interrupt test is being run then the operator will receive the following prompts after entering the keyboard type.

```
TURN UNIT UNDER TEST TO OFF LINE        (operator does this)
TURN UNIT UNDER TEST TO ON LINE         (operator does this)
STRIKE THE KEYS IN ANY ORDER -- HIT RETURN TO PROCEED
```

The section of the test that allows keyboard input in any order is interrupt driven. The operator strikes a key and the program will input the character and then print the input it received on the printer in the following format:

X = >YY

where 'X' is the printable keyboard representation and 'YY' is the hexidecimal ASCII representation. All non- printable characters will have a blank for 'X'. The next character cannot be entered from the keyboard until the previous one has been printed. When the test proceeds from this section, the interrupt capabilities are turned off.

When the operator presses the RETURN key, the keyboard layout for the specified keyboard type will be printed on the unit, and the operator will be given instructions for the keyboard test relative to the type keyboard that was entered.

5.10 E6 -- Execute Test 06 (Tape Read/Write Test)

This test allows the operator to write and verify operator specified data to one or both of the cassette drives. It is also set up so that a verify only can be done on a tape that contains data previously placed on the tape by using this test. It is therefore possible to use this test to insure that data written by one drive can be read by another drive. The following operator inputs are required.

```
ENTER CASSETTE NUMBER (0=BOTH, 1=1, 2=2)  DEFAULT = 00
VERIFY ONLY ? (0=NO, 1=YES)  DEFAULT = 0
ENTER DATA PATTERN (00 - 7F)  DEFAULT = 00
ENTER NUMBER OF BLOCKS
```

The data pattern can be any valid ASCII character from >00 through >7F. If the operator enters a value out of that range, the input will be requested again. The operator must specify the number of blocks to be written or verified or the input will be requested again.

5.11 L1 -- Loop on Test 01

This verb causes TEST 01 to be executed continuously until the operator presses '@', CMD, or HELP.

5.12 L2 -- Loop on Test 02

This verb causes TEST 02 to be executed continuously until the operator presses '@', CMD, or HELP.

5.13 L3 -- Loop on Test 03

This verb causes TEST 03 to be executed continuously until the operator presses '@', CMD, or HELP.

5.14 L4 -- Loop on Test 04

This verb causes TEST 04 to be executed continously until the operator presses '@', CMD, or HELP.

5.15 L5 -- Loop on Test 05

This verb causes TEST 05 to be executed continuously until the operator presses '@', CMD, or HELP.

5.16 L6 -- Loop on Test 06

This verb causes TEST 06 to be executed continuously until the operator presses '@', CMD, ot HELP.

6.0 ERROR MESSAGES

When an error occurs in the execution of any part of the test, an error message will be printed if the print error flag in DOCS is equal to 1. The error message indicates which error has occurred. The number of the error will also appear on the programmer panel. The error messages for this test are listed below.

NUMBER	MESSAGE
1	BAUD RATE AND SPECIFIED UUT DO NOT MATCH

This indicates that the operator specified that the unit under test was a KSR and the baud rate was 1200, or the operator specified that the unit was an ASR and the baud rate was not 1200.

2 REQUESTED STATUS WAS NOT RECEIVED

The program requested the status byte from the ASR and the ASR did not send the status within 1.5 seconds.

3 PRINTER NOT READY

The ASR status indicated that the printer WAS not ready. Check to insure that the printer is in line mode.

4 RECORD NOT READY

The ASR status indicated that record was not ready. Check for record in line mode. Check cassette tape to insure the write enable tab is correct.

5 CASSETTE 2 NOT AT BEGINNING OR END

The ASR status indicated that cassette # 2

was not at beginning or end when it was expected to be.

6 CASSETTE 1 NOT AT BEGINNING OR END

The ASR status indicated that cassette # 1 was not at beginning or end when it was expected to be.

7 NO PLAYBACK ERROR

The ASR status indicated that no playback error occurred when it was expected.

8 PLAYBACK NOT READY

The ASR status indicated that playback was not ready. Check for playback in line mode.

9 PRINTER READY

The ASR status indicated that the printer was ready when it was expected not to be.

A RECORD READY

The ASR status indicated that record was ready when it was expected not to be.

B CASSETTE 2 AT BEGINNING OR END

The ASR status indicated that cassette # 2 was at beginning or end, and it was not expected to be. This would indicate that a tape did not load correctly or that no tape was in the drive when a load command was issued.

C CASSETTE 1 AT BEGINNING OR END

The ASR status indicated that cassette # 1 was at beginning or end, and it was not expected to be. This would indicate that a tape did not load correctly or that no tape was in the drive when a load command was issued.

D PLAYBACK ERROR

The ASR status indicated that an error occurred during playback from a cassette. Probable error in writing to or reading from a cassette tape.

E PLAYBACK READY

The ASR status indicated that playback was ready when it was not expected to be.

F UNKNOWN STATUS ERROR

An error occurred in the routine that checks ASR status against the expected status. The cause of the error could not be determined by the routine.

10 TERMINAL IS TURNED OFF OR IS OFF LINE

The Data Set Ready bit on the interface indicated that the terminal was not ready. It is assumed that the terminal is either turned off or is in the off line mode.

11 CHARACTER NOT TRANSMITTED WITHIN ONE SECOND

The transmit complete signal from the interface indicated that transmission of the last character did not complete within 1.0 second. This might indicate some problem with the interface.

12 VERIFY ERROR ON CASSETTE # XX

The program determined that the data read from the cassette did not correspond to the data that was written to the cassette. This would indicate a read or write problem with the cassette unit or a faulty cassette tape.

13 BLOCK POSITIONING ERROR ON CASSETTE # XX

The program determined that the tape was not positioned properly with the execution of block reverse and block forward commands. This would indicate that block positioning circuitry on the unit was faulty.

14 ADC OFF COMMAND DID NOT TURN OFF AUTOMATICE DEVICE

CONTROL

The program determined that ADC commands could be executed after the ADC off command had be sent to the terminal.

15 KEYBOARD INPUT ERROR -- RECEIVED - >XX, EXPECTED - >XX = X

During the keyboard test, an ASCII character was received from the keyboard which was not the character expected. Usually this is an operator error. Insure that the keyboard specified is the correct one for the unit under test. If the error does not disappear by rerunning the test, the keyboard is generating an improper character.

16 NO KEYBOARD INPUT WITHIN ONE MINUTE

The program did not receive an input from the unit under test within 1.0 minutes.

17 INTERRUPT BIT NOT SET ON INTERFACE FOR UNIT UNDER TEST

The interrupt bit on the interface was not set as it was expected to be during the execution of the interrupt section of the interrupt and keyboard test. Probable cause of error is the interface. This message is associated with input to the interface from the terminal.

18 DATA SET READY DID NOT CHANGE IN ONE MINUTE

During execution of the interrupt test, the interrupt that is generated by setting the terminal on-line or off-line did not occur in 1.0 minute.

19 WRITE REQUEST INTERRUPT DID NOT OCCUR

During execution of the interrupt test, the interrupt associated with output to the terminal did not occur when a character was sent to the terminal. Probable cause of error is the interface.

1A BAUD RATE NOT 300 OR 1200

The baud rate was checked to insure that it was within a five percent tolerance of the common baud rates of 300 and 1200. This message indicates that the interface is not at 300 or 1200. Check the jumpers if the interface is a TTY/EIA card. If the jumpers are correct then the interface needs to be replaced.

6.1 FAILURE ANALYSIS

The following failure analysis data is provided to aid the operator in isolating problems with the Model 733 ASR/KSR. The failure possibilities are listed in order of most probable to least probable in each case. The list below was taken from the Model 733 ASR/KSR Maintenance Manual and is based on maintenance experience.

The TST733 program software is not able to determine whether a verify error is caused by the write function of the cassette or the read function. The operator can use the 'E6' verb to isolate the error to writing or reading. This procedure may be used. First, use 'E6' to write a given data pattern on the tape using the faulty cassette drive. Move the tape to the other drive and attempt a verify only on the tape. If the verify only works on another drive then the probable cause of the error is the read function on the faulty drive. If the verify error occurs on the other drive also, then the probable cause of the error is the write function. It is also recommended that more than one tape be used to eliminate the possiblity of the tape being the cause of the error.

SYMPTOM	POSSIBLE CAUSES
1. Terminal completely	a. AC Power Assembly

inoperative	b. Control/Regulator (A-9) c. Reg/Amp (A-10) d. Transmit/Receive (A-5)
2. Printhead does not return when power is applied	a. AC Power Assembly b. Control/Regulator (A-9) c. Reg/Amp (A-10) d. Printhead Drive e. Printer Control (A-2)
3. Printhead returns to wrong column	a. Printhead Drive b. Reg/Amp (A-10) c. Printer Control (A-2)
4. Paper advance inoperative	a. Printer Control (A-2) b. Reg/Amp (A-10) c. Printhead Drive d. Keyboard
5. Will not print although head steps	a. Printer Code (A-1) b. Printhead Interface (XA-1) c. Printhead (XA-1)
6. Printhead Oscillates	a. Printer Control (A-2) b. Printhead Drive c. Reg/Amp (A-10)
7. Printhead steps irratically	a. Printer Control (A-2) b. Printhead Drive c. Reg/Amp (A-10)
8. Printhead does not lift on paper advance	a. Printhead Drive b. Reg/Amp (A-10) c. Printer Control (A-2)
9. Keyboard inoperative except for paper advance	a. Keyboard b. Printer Code (A-1) c. Terminal Control (A-4)
10. Specific keys will not print	a. Keyboard b. Printer Code (A-1)
11. Serial data not transmitted or received	a. Line interface b. Transmit/Receive (A-5) c. Terminal Control (A-4)
12. Carriage return too slow	a. Printhead Drive b. Printer Control (A-2)
13. Does not print all	a. Printhead (XA-1)

elements	b. Printhead Interface (XA-1) c. Printhead Drive d. Printer Code (A-1)
14. Printhead does not step	a. Reg/Amp (A-10) b. Printhead Drive c. Terminal Control (A-4) d. Printer Control (A-2)
15. Tape will not rewind	a. Tape b. Motion Control (XA-2) c. Tape Transport (XA-9)
16. Tape will not load	a. Tape b. Tape Transport (XA-9) c. Motion Control (XA-2) d. Record Buffer Control (XA-5)
17. Cannot read tape	a. Tape b. Tape Transport (XA-9) c. Tape Read/Write (XA-6) d. Playback Control (XA-4)
18. No character display	a. Record Buffer Control (XA-5) b. Display Card (XA-1)
19. RECORD ON lamp goes out	a. Motion Control (XA-2)
20. Will not write tape	a. Tape Read/Write (XA-6) b. Tape Transport (XA-9) c. Record Buffer Control (XA-5) d. Record Control (XA-7) e. Remote Cas. Control (A-6)
21. Excessive errors while reading	a. Tape Transport (XA-9) b. Tape c. Playback Control (XA-4) d. Record Buffer Control (XA-5)
22. Will not sense BOT	a. Tape b. Tape Transport (XA-9) c. Motion Control (XA-2)
23. Will not block reverse	a. Tape Transport (XA-9) b. Remote Cas. Control (A-6)
24. Reads wrong data from good tape	a. Playback Control (XA-4) b. Record Buffer Control (XA-5)
25. Will not write first	a. Record Buffer Control (XA-5)

character of a block	
26. Will not write tape in line format upon carriage return	a. Record Buffer Control (XA-5)
27. Will not print buffer	a. Record Control (XA-7) b. Record Buffer Control (XA-5)
28. Will not erase tape properly	a. Remote Cas. Control (A-6) b. Tape Transport (XA-9) c. Tape Read/Write (XA-6)

7.0 PART NUMBERS

TST733	2250250-9901	PD, PROGRAM DESCRIPTION
	-2001	DATA, PROGRAM DESCRIPTION ROFF SOURCE
	-9001	PD, ROFF OUTPUT FILE OF PD
	-0009	SP, MICROFICHE LISTING KIT
	-9003	LIST, ASSEMBLY LISTING TST733
	-2003	SRC, SOURCE TST733
	-1003	OBJ, OBJECT TST733
	-9006	LML, LINK MAP LISTING TST733
	-7006	LC, LINK CONTROL TST733
	-1006	FLO, FULLY LINKED OBJECT TST733
TPK733	2250557-9003	LIST, ASSEMBLY LISTING TPK733
	-2003	SRC, SOURCE TPK733
	-1003	OBJ, OBJECT TPK733
KEY733	2250558-9003	LIST, ASSEMBLY LISTING KEY733
	-2003	SRC, SOURCE KEY733
	-1003	OBJ, OBJECT KEY733
SUB733	2250251-9003	LIST, ASSEMBLY LISTING SUB733
	-2003	SRC, SOURCE SUB733
	-1003	OBJ, OBJECT SUB733
IOR733	2250559-9003	LIST, ASSEMBLY LISTING IOR733
	-2003	SRC, SOURCE IOR733
	-1003	OBJ, OBJECT IOR733
MSG733	2250252-9003	LIST, ASSEMBLY LISTING MSG733
	-2003	SRC, SOURCE MSG733
	-1003	OBJ, OBJECT MSG733

8.0 ALGORITHMS

This section contains the algorithms for the verbs and subroutines that make up this test. The algorithms are written in metacode.

Global Variables:

- CRU733 - CRU Base for the unit under test
- INTRLV - Interrupt level for the unit under test
- INTERS - Run the interrupt test flag
- LINEHZ - Line frequency of the CPU
- UUTTYP - Unit under test type
- KSRFLG - KSR flag
- BDRATE - Baud rate flag
- I733FG - Interface initialization done flag
- CMCHIP - Communications chip interface in use flag
- XENBFG - Transmit interrupt enable flag (comm. chip only)
- RRQFLG - Read request interrupt flag
- WRQFLG - Write request interrupt flag
- DSRHTL - Data set ready high to low interrupt flag
- DSRLTH - Data set ready low to high interrupt flag
- INTCHR - Input character from read interrupt
- INSTAT - ASR status that was input by status routine
- BUFFER - Input/Output buffer area

IT Verb - Initialize Test

```
Get CRU733
Convert to ASCII and put in message                    SSBTAA
Request CRU733                                         SSIRP
Get INTRLV
Convert to ASCII and put in message                    SSBTAA
Request INTRLV                                         SSIRP
Get INTERS
Convert to ASCII and put in message                    SSBTAA
Request INTERS                                         SSIRP
Get LINEHZ
Convert to ASCII and put in message                    SSBTAA
Request LINEHZ                                         SSIRP
Get UUTTYP
Convert to ASCII and put in message                    SSBTAA
Request UUTTYP                                         SSIRP
Set KSRFLG
Set BDRATE
If UUTTYP .EQ. an ASR
  then Begin
    Clear KSRFLG
    Clear UUTTYP
    Output 'INSERT TWO SCRATCH CASSETTES'
End
Initialize the interface                               INT733
Return
```

EA Verb -- Execute All

```
Get address of table containing pointers to tests
Do while test pointer .NE. 0
    Get test pointer
    Call test routine
    Increment test pointer to next test
End
Return
```

LA Verb -- Loop on All

```
Do until operator intervention
    Get address of table containing pointers to tests
    Do while test pointer .NE. 0
        Get test pointer
        Call test routine
        Increment test pointer to next test
    End
    Output loop count to front panel                LOOP
End
```

```
IC Verb - Issue Commands to the ASR


If KSRFLG indicates KSR then Return
Output commands with codes
Request command number
If input .GT. maximum command number
  then Request command number
Convert command number to table index
Get address of table containing command characters
Use index to get command
If command .EQ. request status
  then Begin
    Get status from ASR                                  STATUS
    Set up status mask
    Get address of status message
    Do until all bits in status are checked
        Set pointer to correct position in message
        If status bit .EQ. one
          then move '1' to message
          else move '0' to message
        Move to next bit
    End
    Output status message
    Return
End
If command .EQ. block forward
  then Begin
    Input one block from tape to buffer                  BLKIN
    Return
End
Issue command                                            COMAND
Return
```

E1 Verb - Test 01 -- Interface Test

```
Output test header message                                HMOUT
Calculate baud rate setting BDRATE                        BAUDRT
If baud rate .EQ. 1200
  then Output 'BAUD RATE = 1200'                          HMOUT
  else Output 'BAUD RATE = 300'                           HMOUT
If BDRATE .NE. KSRFLG then output error message           EMOUT
If KSRFLG indicates unit is an ASR
  then request ASR status                                 STATUS
Output test complete message
Return
```

E2 Verb - Test 02 -- Dot Matrix and Printhead Test

```
Output test header message                                HMOUT
If KSRFLG indicates an ASR
  then Check for printer ready                            CKSTAT
Output line 1 operator information                        SS7330
Output '#' 80 times                                       ONECHR
Output carriage return                                    SS7330
Output 'H' 80 times                                       ONECHR
Output carriage return                                    SS7330
Output 'I' 80 times                                       ONECHR
Output line 2 operator information                        SS7330
Output SPACE-BLOCK sequence 40 times                      SS7330
Output line 3 operator information                        SS7330
Do 10 times
    Output '#' 8 times                                    ONECHR
    Output backspace 8 times                              ONECHR
    Output 'H' 8 times                                    ONECHR
    Output backspace 8 times                              ONECHR
    Output 'I' 8 times                                    ONECHR
    Output line feed
End
Output test complete message                              HMOUT
Return
```

E3 - Test 03 -- Cassette Test

```
Output test header message
If KSRFLG indicates a KSR
  then Begin
    Output 'SKIPPED' message
    Return
End
Turn ADC on                                          COMAND
Turn record off                                      COMAND
Turn playback off                                    COMAND
Rewind # 1                                           COMAND
Rewind # 2                                           COMAND
Wait for # 1 to rewind                               CKSTAT
Wait for # 2 to rewind                               CKSTAT
Load # 1                                             COMAND
Load # 2                                             COMAND
Wait for # 1 to load                                 CKSTAT
Wait for # 2 to load                                 CKSTAT
Output 'RECORDING TEST PATTERNS'                     HMOUT
Do for cassette # 1 then for # 2
    Put in record mode                               COMAND
    Wait for record ready                            CKSTAT
    Set pointer to first test pattern
    Do while test pattern .NE. 0
        Get test pattern
        Output one block of pattern                  BLKOUT
        Move pointer to next pattern
    End
    Turn record on                                   COMAND
    Turn record off                                  COMAND
    Fast forward                                     COMAND
End
Output 'FAST FORWARD'                                HMOUT
Wait for # 1 to reach end                            CKSTAT
Wait for # 2 to reach end                            CKSTAT
Rewind # 1                                           COMAND
Rewind # 2                                           COMAND
Wait for # 1 to rewind                               CKSTAT
Wait for # 2 to rewind                               CKSTAT
Load # 1                                             COMAND
Load # 2                                             COMAND
Wait for # 1 to load                                 CKSTAT
Wait for # 2 to load                                 CKSTAT
Output 'VERIFYING TEST PATTERNS'                     HMOUT
Do for cassette # 1 then for # 2
    Put in playback mode                             COMAND
    Wait for playback ready                          CKSTAT
    Set pointer to first test pattern
```

```
    Do while test pattern .NE. 0
        Input one block from tape                    BLKIN
        Get test pattern
        Verify block                                 VERIFY
        If an error occurs then output error mss     EMOUT
        Move pointer to next test pattern
    End
End
Output 'CHECKING BLOCK FORWARD AND REVERSE'          HMOUT
Do for cassette # 1 then for # 2
    Put in playback mode                             COMAND
    Wait for playback ready                          CKSTAT
    Block reverse                                    COMAND
    Block forward with input                         BLKIN
    Verify data                                      VERIFY
    If an error occurs then ouput error message      EMOUT
    Block reverse 5 times                            COMAND
    Block forward with input                         BLKIN
    Verify data                                      VERIFY
    If an error occurs then output error message     EMOUT
    Block reverse 5 times                            COMAND
    Block forward with input                         BLKIN
    Verify data                                      VERIFY
    If an error occurs then output error message     EMOUT
End
Turn ADC off                                         COMAND
Turn playback on                                     COMAND
Input a character                                    SS733I
If a character is received then output error mss     EMOUT
Output test complete message                         HMOUT
Return
```

E4 - Test 04 -- Ripple Print Test

```
Output test header message                                    HMOUT
If KSRFLG indicates an ASR
  then Check for printer ready                                CKSTAT
Output operator information message                           SS7330
<first char> = >20
Do 95 times
    <next char> = <first char>
    Set up buffer pointer
    Do 80 times
        Move <next char> to buffer
        Increment <next char>
        Increment buffer pointer
        If <next char> .GT. >7F then <next char> = >20
    End
    Invert last character in buffer for output
    Output one line as in buffer                              SS7330
    Output carriage return line feed                          SS7330
    Increment <first char>
End
Output test complete message                                  HMOUT
```

```
E5 Verb - Test 05 -- Interrupt and Keyboard Test

Local Variables:
       KBTYPE - Keyboard type
       KEYTAB - Keyboard driver table pointers table
       HEADMG - Header message pointer
       KEYBMG - Keyboard layout message pointer

Set up CRU base from CRU733
Output test header message                              HMOUT
Get KBTYPE
Convert to ASCII and move to message                    SSBTAA
Request KBTYPE                                          SSIRP
If KBTYPE .GT. maximum then Request KBTYPE              SSIRP
If INTERS indicates run interrupt test
  then Begin
    Get INTRLV
    Index to correct interrupt trap
    Save current contents of interrupt trap
    Set up interrupr trap for test
    Enable interrupts
    Clear DSRHTL
    Clear DSRLTH
    Output 'TURN UNIT OFF LINE'                         HMOUT
    Wait one minute for DSRHTL to set
    If DSRHTL not set then output error message         EMOUT
    Output 'TURN UNIT ON LINE'                          HMOUT
    Wait one minute for DSRLTH to set
    If DSRLTH not set then output error message         EMOUT
    Output 'STRIKE THE KEYS IN ANY ORDER...'            HMOUT
    Clear RRQFLG
    Clear WRQFLG
    Do until carriage return is input
        Wait one minute for RRQFLG to set
        If RRQFLG set
          then Begin
            Get INTCHR
            Put INTCHR in message
            Convert INTCHR to ASCII and put in msg      SSBTAA
            Output 'X = >XX   '                         SS7330
            If WRQFLG not set then output error msg     EMOUT
        End
          else output error message                     EMOUT
    End
    Disable interrupts
    Restore previous interrupt trap
End
Get KBTYPE
Convert KBTYPE to an index
Use index to get correct driver table pointer from KEYTAB
```

```
Do 4 times
    Get HEADMG from driver table
    Output header message from HEADMG                SS7330
    Get KEYBMG from driver table
    Output keyboard layout message from KEYBMG       SS7330
    Get pointer to expected data from driver table
    Do until last expected data received
        Wait one minute for input                    SS733I
        If input not received
          then output error message                  EMOUT
          else Begin
            If input .EQ. expected data
              then increment pointer to expected data
              else Begin
                Move received data to message
                Move expected data to message
                Output error message                 EMOUT
            End
        End
    End
End
Output test complete message
Return
```

```
E6 Verb - Test 06 -- Tape Write/Read Test


Local Variables:
        CASNUM - Cassette number for testing
        VERONY - Verify only flag
        TSTPAT - Test pattern
        BLKNUM - Number of blocks


Output test header message                                  HMOUT
If KSRFLG indicates a KSR
  then Begin
    Output 'SKIPPED'                                        HMOUT
    Return
End
Turn ADC on                                                 COMAND
Turn record off                                             COMAND
Turn playback off                                           COMAND
Get CASNUM
Convert to ASCII and put in message                         SSBTAA
Request CASNUM                                              SSIRP
Clear VERONY
Request VERONY                                              SSIRP
Get TSTPAT
Convert to ASCII and put in message                         SSBTAA
Request TSTPAT                                              SSIRP
Clear BLKNUM
Request BLKNUM                                              SSIRP
If BLKNUM .LT. 1 then Request BLKNUM                        SSIRP
Rewind # 1                                                  COMAND
Rewind # 2                                                  COMAND
Wait for # 1 to rewind                                      CKSTAT
Wait for # 2 to rewind                                      CKSTAT
If CASNUM .NE. 2
  then Begin
    Load # 1                                                COMAND
    Wait for # 1 to load                                    CKSTAT
End
If CASNUM .NE. 1
  then Begin
    Load # 2                                                COMAND
    Wait for # 2 to load                                    CKSTAT
End
If VERONY indicates not verify only
  then Begin
    Output 'RECORDING TEST PATTERNS'                        HMOUT
    Do for cassette # 1 then for # 2
        If CASNUM .EQ. 2 then skip for # 1
        Record mode                                         COMAND
```

```
          Wait for record ready                          CKSTAT
          Do BLKNUM times
              Get TSTPAT
              Output one block to tape                   BLKOUT
          End
          Turn record on                                 COMAND
          Turn record off                                COMAND
          If CASNUM .NE. 2
            then Begin
              Rewind # 1                                 COMAND
              Wait for # 1 to rewind                     CKSTAT
          End
          If CASNUM .EQ. 1 then skip for # 2
     End
     If CASNUM .NE. 1
       then Begin
         Rewind # 2                                      COMAND
         Wait for rewind # 2                             CKSTAT
     End
     If CASNUM .NE. 2
       then Begin
         Load # 1                                        COMAND
         Wait for # 1 to load                            CKSTAT
     End
     If CASNUM .NE. 1
       then Begin
         Load # 2                                        COMAND
         Wait for # 2 to load                            CKSTAT
     End
End
Output 'VERIFYING TEST PATTERNS'                         HMOUT
Do for cassette # 1 then for # 2
     If CASNUM .EQ. 2 then skip for # 1
     Playback mode                                       COMAND
     Wait for playback ready                             CKSTAT
     Do BLKNUM times
         Get TSTPAT
         Input one block                                 BLKIN
         Verify input data                               VERIFY
         If an error occurs then output error msg        EMOUT
     End
     If CASNUM .EQ. 1 then skip for # 2
End
Output test complete message                             HMOUT
Output 'FOR COMPATABILITY TEST...'                       HMOUT
Return
```

```
SS7330 -- Output Routine

Parameters:
       Address of data to be output


Set CRU base to CRU733
Get address of data to be output
If I733FG indicates initialization is needed
  then Initialize the interface                          INT733
Do until most significant bit is set in character
    Output character                                     OUTPUT
    If BDRATE indicates an ASR
      then Wait 3 character time delay                   TDELAY
    If character is a carriage return
      then If BDRATE indicates an ASR
             then Wait 24 character time delay           TDELAY
             else Wait  6 character time delay           TDELAY
    If character is a line feed
      then If BDRATE indicates an ASR
             then Wait 4 character time delay            TDELAY
             else Wait 1 character time delay            TDELAY
End
Invert last character (MSB set)
Increment last character
Output last character                                    OUTPUT
If BDRATE indicates an ASR
  then Wait 3 character time delay                       TDELAY
If character is a carriage return
  then If BDRATE indicates an ASR
         then Wait 24 character time delay               TDELAY
         else Wait  6 character time delay               TDELAY
If character is a line feed
  then If BDRATE indicates an ASR
         then Wait 4 character time delay                TDELAY
         else Wait 1 character time delay                TDELAY
If interface is a TTYEIA card then Clear WRQ
Return
```

```
SS733I - Input Routine

Parameters:
       Address where to store the character input
       TOUTS timer count where 1 count =  8.3 ms 120 hz.
                                         10.0 ms 100 hz.

Local Variables:
       STRFLG - Input character stored flag


Set CRU base to CRU733
Get address to put input character
Get timer count for TOUTS
If timer count .EQ. 0 then Set timer count to >FFFF
If I733FG indicates that initialization is needed
  then Initialize the interface                          INT733
Start the timer                                          TOUTS
Do until STRFLG indicates a character is input
    Input one character                                  INCHAR
    If timer count .EQ. 0
      then Begin
        Turn the clock off
        If interface is a TTYEIA card
          then Begin
            Clear new status flag
            Clear write request
            Clear read request
        End
        Return
    End
End
Move character to address to put input character
Return
```

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV	
SHEET	
REV STATUS OF SHEETS	REV
	SHEET

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ± 010
2 PLACE DECIMAL ± 02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH IL STD 100

DWN	DATE
CHK	
ENGR	
QA	
APVD	
CONTR NO	
DESIGN ACTIVITY RELEASE	

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

PD, TSTS820, 820KSR DATA TERMINAL DIAGNOSTIC -990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	2250255-9901

SCALE	REV. A	SHEET 1 of 28

TI-9023-C

TABLE OF CONTENTS

1.0 SCOPE

This document describes the 820 KSR Data Terminal diagnostic (TST820). The diagnostic is designed to meet the following goals:

a) To provide factory test with the ability to thoroughly test all features of the 820 KSR Data Terminal and isolate faults to a specific kit component.

b) To provide the capability for our customer engineers at the remote site to thoroughly test the overall performance of the 820 KSR Data Terminal and isolate any faults to one of three (3) field replaceable units, some of which may contain other subassemblies.

2.0 REFERENCES

For information beyond the scope of this document refer to the followins:

PART NUMBER	TITLE
0943442-9701	Model 990 Computer Reference Manual
0943441-9701	Model 990 Assembly Language Manual
0945400-9701	Model 990 Computer Diagnostic Handbook
2262095	Model 820 KSR Data Terminal Specification
0999856-9701	User's Guide for Model 820 KSR
0999854-9701	Operating Instructions for Model 820 KSR
0945408-9701	TTY/EIA Interface Module Maintenance Manual
	TMS 9902 Data Manual
	TMS 9903 Data Manual

3.0 EQUIPMENT and SOFTWARE REQUIREMENTS

This section describes the minimum equipment requirements and the available linked object software for each of the test versions.

3.1 EQUIPMENT REQUIREMENTS

In addition to the equipment specified in the Diagnostic Handbook, the following equipment is required:

a) Model 820 KSR Data Terminal Kit P/N 2262090

b) A 990 computer with 12K words of memory.

3.2 SOFTWARE REQUIREMENTS

The following describes the linked object modules available in the TST820 release:

Loadable test module: TST820, 2250255-1006 (FLO)

Linkable parts: TST820
PRTSUB
KSRSB1
KSRSB2
KSRSB3
KSRSB4
TXT820

4.0 LOADING

Loading procedures from all available media are found in the Diagnostic Handbook.

5.0 TEST EXECUTION and DESCRIPTION

As this is a DOCS based test, reference should be made to the Diagnostic Handbook for information on DOCS start-up and initialization.

5.1 INITIALIZATION

After the test module is loaded into memory, the test will output the following message:

TST820 MODEL 820 KSR DATA TERMINAL DIAGNOSTIC VERSION MM/YY *X

where MM/YY *X is the date and revision that the test was last released. Following this, the test will ask a number of initialization questions. Refer to section 5.2.7 for detailed questions.

5.2 TEST VERB DESCRIPTIONS

The TST820 provides special verbs for testing various capabilities of the 820 KSR Data Terminal. The use of these verbs is explained in the following paragraphs.

5.2.1 EXECUTE TEST VERB (ET)

VERB ?-ET TEST # -XX

The ET verb executes the test specified by the HEX number XX.

5.2.2 LOOP ON TEST VERB (LT)

VERB ?-LT TEST # -XX

The LT verb loops on the test specified by the HEX number XX.

5.2.3 EXECUTE ALL VERB (EA)

VERB ?-EA

The EA verb executes all available tests in sequential order.

5.2.4 LOOP ON ALL VERB (LA)

VERB ?-LA

The LA verb loops on all tests in order. Tests which require operator intervention are not executed.

5.2.5 KEYBOARD ECHO TEST (KE)

VERB ?-KE

The KE verb requests the operator to press keys on the keyboard at random. The hexidecimal value of the input key is printed. The operator must then visually verify that the codes received are the ones expected. The repeat function may be tested by holding a key down. The KE test may be terminated by pressing the <@> key on the unit under test. A failure of the KE test occurs when the data received and echoed by the program does not match the value expected by the operator. The KC verb should be executed to further isolate the fault. The KE verb also does interrupt testing and checking. The appropriate messages are printed in the case of an interrupt error.

5.2.6 KEYBOARD COMPLETE TEST (KC)

VERB ?-KC

The KC verb directs the operator to strike specific keys, thoroughly testing the operation of each and every key. A diagram of the keyboard is printed with each key numbered. The operator is directed to press these keys in numeric order. This test is done in three parts. The first part checks the unshifted function. The second part checks the shifted keys. The third part checks the control function. In each part, directions of which keys to press are given along with a keyboard diagram. This test may be terminated by pressing the <@> key on the unit under test except when this key is being tested. If parameter 86 is enabled an error message will be issued upon typing of the <RETURN> key or <CONTROL M> to the effect that an unexpected code was received. This is because typing of <RETURN> will transmit <carriage return/line feed> and only <carriage return> is expected. Failure of this test is indicated by an error message. Possible causes are: 1) Operator did not type any key, 2) The key was typed out of order, 3) The keyswitch is faulty, 4) The electronics generating the codes are faulty, or 5) The interface to the computer is faulty.

5.2.7 INITIALIZE TEST VERB (IT)

TST820 provides the IT verb so the 820 KSR Data Terminal parameters can be made available to TST820. Location of information concerning installed options and parameters may be found in the 820 KSR Data Terminal User's Guide.

```
VERB ? -IT
COMM DEVICE? (0=TTY/EIA, 1=COMM CHIP) DEF = 00  -
(If the default is taken the following are asked:)
CRU BASE? DEF = 0060  -
INTERRUPT LEVEL? DEF = 0E  -
(If COMM DEVICE=1 is taken the following are asked:)
CRU BASE? DEF = 1740  -
INTERRUPT LEVEL? DEF = 0E  -
DO YOU WANT TO TEST INTERRUPTS? DEF = 00  -
COLUMN WIDTH (0=132, 1=80)? DEF = 01  -
LINE FREQUENCY (0=50HZ, 1=60HZ)? DEF = 01  -
DOES TERMINAL HAVE NUMERIC PAD? DEF = 01  -
IS DFC OPTION INSTALLED? DEF = 01  -
CHARACTER SET SUBOPTION? (0=USF,1=UKF,2=FRF,3=GRF) DEF = 00  -
ENTER THE TERMINAL CONFIGURATION PARAMETERS.
DEF = 14,27,37,81,86
-
```

After these questions are answered, TST820 is ready to execute the specified sub-test or verb.

5.3 NUMBERED SUB-TEST DESCRIPTIONS

5.3.1 Test 01 tests the communications interface baud rate by sending null characters, counting the number of characters transmitted in a given time, and comparing to a table of acceptable values for the baud rate specified in the initialization step (IT verb). Failure is indicated by an error message. Refer to the appropriate error message for possible causes of failure.

5.3.2 Test 02 prints one line of 20 characters for each printable character. Failure is indicated by incorrect printing of one or more characters. Possible causes: 1) Faulty printing mechanism or 2) Faulty electronics driving the printing mechanism.

5.3.3 Test 03 prints a ripple dump of all characters in the character set. Each line contains 80 (or 132) characters and each character appears once in each print position. Failure is indicated by incorrect printing of one or more characters in one or more print positions. Refer to 5.3.2 for possible causes.

5.3.4 Test 04 prints a diamond pattern of characters to test the printer buffer. Failure is indicated by incorrect printing of the diamond pattern. Possible causes: 1) Printing mechanism too slow for buffer or 2) Buffer electronics faulty.

5.3.5 Test 05 is the carriage return test. An English sentence is printed with every other character omitted, giving a carriage return and printing the characters which were omitted before. If the carriage return works correctly the following message is printed on the printer:

THIS LINE IS PRINTED IN TWO PARTS; VERIFY THAT ONLY ONE LINE IS PRINTED.

Parameter 85 must be disabled for this test to run. If not it is skipped. Failure is indicated by the printing of an unintelligible sentence. Possible causes: 1) Parameter 85 was enabled on the 820 but not included in the parameters entered in the IT verb, or 2) <carriage return> electronics are faulty.

5.3.6 Test 06 tests the programmable forms length. First the length is set to 33 lines by transmitting: <esc>[33t. Then a <form feed> command is given and the following message is printed on the 820 under test:

THIS IS THE CENTER OF FORM PAGE ONE OF TEST 06

The forms length is then reset to 66 by transmitting: <esc>[66t and the following message is printed on the top of the next page:

FORM FEED THIS IS TOP OF FORM PAGE TWO OF TEST 06

Failure of this test is indicated by the printed information being incorrect. Possible causes: 1) DFC option is not installed, 2) Parameter 81 is not set (enables DFC commands from computer) or, 3) DFC electronics are faulty.

5.3.7 Test 07 tests the tab to line command. The following message is printed ten(10) times on the 820 under test:

THIS IS LINE XX ON THIS FORM YY LINES FROM TOP

This is done by transmitting: <esc>[XXd where XX takes the values 1, 2, 4, 8, 16, 32, 40, 44, 46, and 47; and YY takes the values 4, 5, 7, 11, 19, 35, 43, 47, 49, and 50. This test is skipped if the DFC option is not installed. Refer to 5.3.6 for failure and possible-causes information.

5.3.8 Test 08 tests the vertical set and tab commands by transmitting <esc>[4g to clear all existing tabs, <esc>[4;5;7;11;19;35;43;47;49;50v to set tabs, and using the <vertical tab> command to test the tabs. Refer to 5.3.6 for failure and possible-causes information.

5.3.9 Test 09 tests the horizontal set and tab commands by transmitting <esc>[2g to clear existing tabs, <esc>[10;15;20;40;80;90;115;117;119;128u to set the tabs, and using the <horizontal tab> command to test the tabs. Refer to 5.3.6 for failure and possible-causes information.

5.3.10 Test 0A is the tab to column test. <esc>[M` is transmitted where M is the column to tab to. M takes on the values 10,15,20,40,80,90,115,117,119,128 in turn. This test is skipped if the DFC option is not installed (indicated to TST820 by parameter 81 not enabled. Refer to 5.3.6 for failure and possible-causes information.

5.3.11 Test 0B tests the setting of left and right margins. This is done by sending <esc>[N;Ms where N is the left margin and M is the right margin. N takes on the values 10, 20, 30 and M takes on the values 40, 50, 60 in turn. Three lines of "B" are then sent to the printer to verify the new settings. Refer to 5.3.6 for failure and possible-causes information.

5.3.12 Test 0C tests the setting of top and bottom margins. This is done by sending <esc>[N;Mr where N is the top margin and M is the bottom margin. N takes on the values 10, 20, 30 and M takes on the values 40, 50, 60 in turn. Thirty lines of "C", ten characters per line are sent to verify the new margins. Refer to 5.3.6 for failure and possible-causes information.

5.3.13 Test 0D tests the lines per inch options. Sending <esc>PA\ sets 6 lines/inch and <esc>PB\ sets 3 lines/inch. Six lines of "D", sixteen characters per line are sent to verify settings. This test is skipped if the DFC option is not installed (indicated to TST820 by parameter 81 not enabled). Refer to 5.3.6 for failure and possible-causes information.

5.3.14 Test 0E tests the characters per inch options. This test is skipped if the DFC option is not installed (indicated to TST820 by parameter 81 not enabled). Sending <esc>PC\ sets 10 chars/inch and six lines of "E", 10 characters per line are sent to verify setting. Then <esc>PD\ sets 16.5 chars/inch and six lines of "E", seventeen characters per line are sent to verify. Refer to 5.3.6 for failure and possible-causes information.

5.3.15 Test OF tests the answer back memory (ABM) remotely triggered. This is done by sending an <ENQ> character (HEX 5) and echoing the message as it is received. The ABM must be programmed for this test to run properly. Failure is indicated by reciept of incorrect ABM contents. An error is issued if less than 21 characters are recieved. Possible causes are: 1) The ABM is not programmed or 2) The internal electronics for storage and transmittal of ABM are faulty.

THE FOLLOWING TESTS REQUIRE OPERATOR INTERVENTION:

5.3.16 Test 10 tests the auto <cr/lf>. Parameter 86 must be enabled for this test to run. Four lines of data are sent to the terminal. The operator is directed to press <RETURN> at the end of each and verify that four separate lines were typed. Failure of this test is indicated by less than four lines being typed. Possible causes are: 1) <RETURN> key is faulty, 2) Parameter 86 is not enabled, or 3) Internal electronics controlling auto <cr/lf> are faulty.

5.3.17 Test 11 tests the operation of the programmable <ENTER> key sequence. The operator is directed to configure the <ENTER> with <TAB><CR><LF>, then to press the key. Failure is indicated by an error message. Possible causes are: 1) Incorrect configuration of <ENTER> key, 2) Faulty <ENTER> key, or 3) Faulty internal storage for <ENTER> key.

6.0 MESSAGES

This section describes all messages which TST820 can produce. It is hoped that such messages will be self explanatory. However, should this not be the case, each message is described fully. TST820 issues three types of messages. They are 1) Error messages, 2) Informative messages, and 3) Directive messages. Each type is discussed in a separate section which follows.

6.1 ERROR CODES AND MESSAGES

These messages are generated as the result of an error condition in a test. They are designed to aid the operator in locating a fault in the unit under test. Each message is given below with the reason and possible cause of failure.

CODE	MESSAGE

01 INVALID CONFIGURATION PARAMETER ENTERED

This message is generated when the configuration parameter entered in response to the initialization question is not a valid value. The entire configuration report must be reentered.

02 CODE NOT RECEIVED, DELAY TIMED OUT

This message is generated when the test is expecting an input from the keyboard and does not receive one. Possible causes are: 1) A bad keyswitch, 2) Operator did not press any key in the allowed time, or 3) A bad interface circuit which prevents receipt of input.

03 EXPECTED = XX, RECEIVED = YY

When a key input is received it is compared to the expected value. If these two are not equal this message is generated. Possible causes are: 1) Bad code generated by keyswitch, or 2) Bad interface circuitry.

04 TRANSMIT IN PROGRESS TIMED OUT

The terminal busy signal should have been removed within a specific amount of time and was not. Possible cause: Busy signal generating electronics faulty.

05 TERMINAL IS OFF LINE. DSR LOW AFTER TIME OUT

The Data Set Ready signal is low following a specific time. Possible causes: 1) Terminal is off line, 2) Terminal is not powered up, or 3) electronics generating DSR signal are faulty.

06 EXPECTED NEW STATUS INTERRUPT (DSR HIGH TO LOW) DID NOT OCCUR

TST820 expected DSR to go from high to low and it did not. Possible cause: electronics generating DSR signal are faulty.

07 EXPECTED READ REQUEST INTERRUPT DID NOT OCCUR

TST820 expected a read request interrupt and it did not receive one. Possible cause: Faulty read request signal generating electronics.

08 EXPECTED WRITE REQUEST INTERRUPT DID NOT OCCUR

TST820 expected a write request interrupt and it did not receive one. Possible cause: Faulty write request signal generating electronics.

09 UNEXPECTED INTERRUPT AT TERMINAL INTERRUPT LEVEL

An interrupt came at the specified terminal interrupt level but not from the terminal. Possible cause: Improper interrupt level specification.

0A TEST 01 MUST RUN WITHOUT ERROR BEFORE RUNNING OTHER TEST

An error was generated during the execution of test 01. This error should be cleared up to assure the validity of subsequent tests.

0B BAUD RATE LOW

The calculated baud rate is lower than the specified allowable limit for the unit under test. Possible causes: 1) Terminal baud rate and interface baud rate are incompatible or 2) Faulty electronics generating baud rate on interface.

0C BAUD RATE HIGH

The calculated baud rate is higher than the allowable limit for the unit under test. Refer to error 0B (above) for possible cause.

0D EXPECTED <CR>; NOT RECEIVED

The test was expecting to receive a carriage return character from the <RETURN> key depression and did not. Possible causes are: 1) The operator did not press <return> key, 2) Terminal configuration parameter 86 was not enabled, 3) A bad keyswitch is present, or 4) The electronics generating <carriage return/line feed> are faulty.

0E EXPECTED <LF>; NOT RECEIVED

The test was expecting to receive a line feed character from the <RETURN> key depression and did not. Refer to error 0E (above) for possible causes.

0F INVALID CHARACTER SET SUBOPTION

The character set suboption entered in the IT verb sequence is not a valid entry. The parameter should be reentered.

6.2 INFORMATIVE MESSAGES

These messages are given to indicate the starting and ending of tests. Also included here are messages which indicate completion status of tests. General examples follow:

TEST XX COMPLETE

Indicates completion of the specified test.

TEST XX SKIPPED

Indicates that the test was not run because a parameter was disabled during test initialization.

BEGIN TEST XX...

Indicates beginning of the specified test. Usually printed on the unit under test. This message will also describe the objective of the specified test.

TEST NOT AVAILABLE

Indicates that the test number requested was invalid.

6.3 DIRECTIVE MESSAGES

These messages give directions to the operator on how to perform the test. These are primarily for the manual intervention tests to specify the sequence of operations to perform in order to complete the test. For example TEST 10, Auto cr/lf test generates the following message:

**** BEGIN TEST 10 AUTO CR/LF TESTING ****

DEPRESS RETURN AT THE END OF EACH PRINTED LINE

VERIFY THAT FOUR SEPARATE LINES ARE PRINTED

7.0 PART NUMBERS

TITLE	NUMBER	
PROGRAM DESCRIPTION	2250255-2001	ROFF Source
	-9001	ROFF Output
	-9901	PD Document
FICHE KIT	-0009	SP
TST820 - Linked Test	-1006	FLO
	-7006	LC
	-9006	LML
TST820 - Main Module	-1003	OBJ
	-2003	SRC
	-9003	LIST
PRTSUB - I/O Service	2250256-1003	OBJ
	-2003	SRC
	-9003	LIST
KSRSB1 - Test #s	2250257-1003	OBJ
- 1,2,3,4,5	-2003	SRC
	-9003	LIST
KSRSB2 - Test #s	2250258-1003	OBJ
- 6,7,8,9,A	-2003	SRC
	-9003	LIST

KSRSB3 - Test #s	2250259-1003	OBJ
- B,C,D,E,F	-2003	SRC
	-9003	LIST
KSRSB4 - Manual Tests	2250260-1003	OBJ
- 10,11,KE,KC	-2003	SRC
	-9003	LIST
TXT820 - Text Module	2250261-1003	OBJ
	-2003	SRC
	-9003	LIST

A

APPLICATION		REVISIONS			
NEXT ASSY	USED ON	LTR	DESCRIPTION	DATE	APPROVED
	7506				

REV																													
SHEET																													

REV STATUS OF SHEETS — REV / SHEET

UNLESS OTHERWISE SPECIFIED DIMENSIONS ARE IN INCHES
TOLERANCES
ANGLES ±1°
3 PLACE DECIMAL ±.010
2 PLACE DECIMAL ±.02

IDENTIFYING NUMBERS SHOWN IN PARENTHESES FOR REFERENCE ONLY

INTERPRET DWG IN ACCORDANCE WITH MIL STD 100

DWN | DATE
CHK
ENGR
QA
APVD
CONTR NO
DESIGN ACTIVITY RELEASE

TEXAS INSTRUMENTS
INCORPORATED
Equipment Group Dallas, Texas

PD, TTYEIA, FULL DUPLEX TTY/EIA INTERFACE MODULE TEST-990

SIZE	CODE IDENT NO	DRAWING NO
A	96214	2250131-9901

SCALE | REV. A | SHEET 1 of 39

TI-0025 C

TABLE OF CONTENTS

1.0 SCOPE

The TTYEIA Test is a Texas Instruments computer program, which is used to test the Full Duplex TTY/EIA Interface Module.

The TTYEIA Test program determines whether the board is operating correctly. If it is not, the TTYEIA test determines the type of error and the exact signal or signals which are malfunctioning. The output from the TTYEIA Test is an aid in determining what repair or replacement should be performed to correct a problem.

The TTYEIA Test is run under control of DOCS (Diagnostic Operational Control System). DOCS is a simple operating system under which the diagnostic test programs can run. It contains routines which handle the operator interface devices and provide service functions to the test programs.

2.0 REFERENCES

Part No.	Document
945408-9701	Model 990 Computer TTYEIA Interface Module Depot Maintenance Manual
945400-9701	Model 990 Computer Diagnostics Handbook
02250131-1006	TTYEIA, Full Duplex TTYEIA Interface Module Test--990 (Computer Program)
2262130	Loopback Connector TTY/EIA Module (Test) Drawing
945077	Logic Diagram, Full Duplex TTY/EIA Module

3.0 Equipment and Software Requirements

3.1 Test Equipment and Software

The following equipment is needed to perform a test:

(a) 990 Computer with 12K or more of memory.

(b) Appropriate interactive device.

(c) The TTY/EIA Module to be tested. Note: The TTY/EIA Module to be tested cannot also be the interface used for the interactive device.

(d) TTY/EIA Loopback Test Connector.

(e) Floppy Disc or other I/O device suitable for loading the test program.

In addition, the operator must have the TTYEIA Test computer program to be loaded into the 990 computer.

The loadable computer program is TTYEIA, 2250131-1006 (FLO).

In setting up the hardware, the TTY/EIA board (to be tested) should be placed at an interrupt level not used by the remainder of the equipment.

3.2 TTY/EIA Jumper Configurations

The following chart gives information on how to set TTY/EIA board jumpers, in order to enable specific capabilities.

(1) Baud Rate:

	75/110	300	1200	2400	4800	9600
E1 to	E2	E3	E4	E5	E6	E7

(2) Device Logic Levels:

	EIA Level	TTY Level
E9 to	E8	E10

(3) Code Format:

		10-Bit Code	11-Bit Code
(a) Receive	E12 to	E11	E13
(b) Transmit	E15 to	E14	E16

(4) Receive during "Request to Send":

	Enable Receive	Disable Receive
E18 to	E17	E19

(5) 110 Baud Rate:

	110 Baud	All other baud rates
(a) E21 to	E22	E20
(b) E23 to	E24	No jumper installed

(6) RTSE Output: 306/588 and MC-810

	Line Printers	All Other Devices
E26 to	E27	E25

The following jumper settings must be accomplished prior to running the TTYEIA Test:

(a) Set the baud rate for which the board is to be tested. The baud rate is set by placing a jumper from E1 to the contact indicated at the baud rate in the aforegoing chart. If a rate other than 110 is desired, jumper E21 to E20, and ensure that there is no jumper from E23 to E24. If baud rate 110 is desired, jumper E1 to E2, E21 to E211 and E23 To E24.

(b) Contact E9 must be jumpered to E8, for the EIA interface.

(c) The code format (10-bit or 11-bit) must be specified. For 10-bit format, jumper E12 to E11 and E15 to E14. For the 11-bit format, jumper E12 to E13 and E15 to E16.

(d) Contact E18 must be jumpered to E17, to enable the board to receive data which is looped back.

(e) Jumper E26 to E25, to maintain DTR (Data Terminal Ready) and RTS (Request to Send) as separate signals. The jumper should not be placed from E26 to E27, because this will set DTR to RTS and cause the Interrupt Test to fail in Loopback Mode.

3.3 Configuration of TTY/EIA Loopback Test Connector

The Loopback Test Connector wires EIA interface outputs back to inputs as follows:

(a) Pin 3 to pin 2, Transmitted Serial Data to Received Serial Data

(b) Pin 16 to Pin 14, Reverse Channel Transmit to Reverse Channel Receive (Serial data)

(c) Pin 6 to pin 20, Data Terminal Ready to Data Set Ready

(d) Pin 8 to pin 18, Request to Send to Data Channel Detect

4.0 Loading

Loading information for each medium is contained in the Model 990 Computer Diagnostics Handbook. The TTYEIA Test is loaded by DOCS (Diagnostic Operational Control System.)

When the TTYEIA Test is loaded, the following message is output:

TTY/EIA MODULE TEST VERSION # = MM/YY *X

where MM/YY is the release date and *X the revision level of the test. The release date is used as a version number to allow the object program to be related to the correct source listing.

5.0 Test Execution and Description

The procedure to operate the TTYEIA Test consists generally of the following steps:

(a) attaching the Loopback Test Connector to the EIA interface of the TTYEIA board,
(b) loading the TTYEIA Test program into the 990 computer, and
(c) operating the program and noting the test results, through an interactive device.

5.1 Loopback Test Connector Usage

The TTYEIA Loopback Test Connector, part no. 2262130, should be plugged into the EIA socket on the TTYEIA board. The jumper configuration (on the board) for Device Logic Level should be set for the EIA Level (see section 3.2, TTYEIA Jumper Configurations).

If it is not possible to use the loopback connector, the board may still be tested. It is preferable to use connector, because the entire board may be tested by the program in "loopback mode." When the computer program is in "diagnostic mode" (loopback connector not used), several components of the TTYEIA board are not tested.

The circuits in the loopback connector are specified in part 3.3, Configuration of TTYEIA Loopback Test Connector.

5.2 Verbs for Program Execution

The first step in operating the TTYEIA Test, after loading by DOCS, is to initialize the program by inputing values which relate to

the hardware configuration. By executing the IT verb, the operator is able to input (or modify) the required values.

The remaining verbs are used to execute specific types of tests. As tests are performed, messages are output to the interactive device to indicate which test is currently executing. The specific tests, mentioned in the definition of these verbs, are described in detail in section 5.3, TTYEIA Test Software Description.

5.2.1 IT Verb--Initialization of Program

The IT verb sets up a question-and-answer sequence, in which the operator is prompted to input required data. When the TTYEIA program begins execution after loading, it automatically executes the IT Verb to obtain needed inputs. At any time during a test, the operator may execute the IT Verb again, in order to modify an input.

The IT verb messages, and appropriate operator responses, are defined in the following paragraphs. If the response is terminated (i.e. by striking the Return key) without entering a value, the default value will be used.

(a) Message:

LOOPBACK CONNECTOR INSTALLED ON TTY/EIA BOARD? (YES=1 NO=0 DEF= 01) -

Response: The operator should plug the loopback test connector onto the EIA interface of the TTYEIA board. If he cannot do this, he should enter a zero before terminating the response. The response to this message should be 1 if the loopback connector is attached, or zero if the loopback connector is not attached.

(b) Message:

TTY/EIA CRU BASE? (DEF= 0000) -

Response: The operator must enter the CRU Base Address of the slot where the TTYEIA board has been installed.

(c) Message:

TEST INTERRUPTS? (YES=1 NO=0 DEF= 01) -

Response: If the TTY/EIA Module is placed at a CRU Base Address which is equipped with an interrupt level (interrupt capability), then 1 should be entered. If there is no interrupt capability, a zero must be input in order to inhibit the Interrupt Test.

(d) Message:

TTY/EIA INTERRUPT LEVEL? (DEF= 00) -

Note--this message occurs only if a "Yes" response was given to the previous message, regarding the interrupt test.

Response: The effective interrupt level of the TTYEIA board (to be tested) must be entered.

(e) Message:

LINE FREQUENCY? (60=0 50=1 DEF= 00) -

Response: The operator must enter 0 or 1 to indicate the the frequency (60 Hertz or 50 Hertz) of the AC power supply to the 990 computer. (A real time clock in the 990, used in the TTYEIA Test, is based upon the AC frequency.)

(f) Message:

ENTER BAUD RATE (75, 110, 300, 1200, 2400, 4800, or 9600 DEF= 0075) -

Response: The operator should enter the baud rate which is expected to result from the current TTYEIA jumper configuration. (Section 3.2 presents the TTYEIA jumper configurations). This message will repeat if the operator inputs a value other than an allowed baud rate.

(g) Message:

BITS PER CHARACTER? (10 BITS=0 11 BITS=1 DEF= 00) -

Response: The operator should indicate whether the TTYEIA board jumpers are set for the 10-bit code or the 11-bit code. .(Section 3.2 defines the TTYEIA jumper configurations.) Note--the Receive Code Format and Transmit Code Format must be the same (they must both be 10-bit code or must both be 11-bit code).

5.2.2 EA Verb--TTY/EIA Module Test

The EA verb causes a complete TTYEIA Module Test to be performed. The TTY/EIA Module Test consists of the following series of tests:

(a) initializing and testing of status lines,

(b) the baud rate test,

(c) the transmit, receive, and compare test,

(d) the timing error test, and

(e) the interrupt test.

The interrupt test will be performed only if this option (Test Interrupts) was specified at program initialization. The tests in (a) through (e) above are described in detail in section 5.3, TTYEIA Software Description.

5.2.3 E1 Verb--Initialize and Test Status Lines

The E1 verb causes the TTY/EIA status lines to be initialized and tested (as described in section 5.3.1).

5.2.4 E2 Verb--Baud Rate Test

The E2 verb causes the status lines to be initialized and tested, and a test to be performed which determines the baud rate of data transmission. If the test determines a baud rate which differs by less than 2 percent from the operator input, the message,

BAUD RATE= XXXX

where XXXX is the correct baud rate, will be output on the interactive device. Otherwise, an error message will be output. (The baud rate test error messages, and recommended action, are discussed in section 6, Messages).

5.2.5 E3 Verb--Transmit, Receive, and Compare Test

The following operations are performed upon execution of the E3 verb:

(a) The status lines are initialized and tested, and
(b) the Transmit, Receive, and Compare Test is performed.

If the TTYEIA Test was initialized to be in Loopback Mode, the Transmit, Receive, and Compare Test is performed first in Diagnostic Mode and then in Loopback Mode. (See section 5.2.1, IT Verb.) If the program was initialized for Diagnostic Mode, the test will be performed in Diagnostic Mode only.

5.2.6 E4 Verb--Timing Error Test

The E4 verb causes the following to occur:

(a) The status lines are initialized and tested, and

(b) the Timing Error Test is performed.

If the program was initialized for Loopback Mode, these computations will be performed twice, once each for the Diagnostic and Loopback Modes. If the program was initialized for Diagnostic Mode, then the test will be performed only in that mode.

5.2.7 E5 Verb--Interrupt Test

The E5 verb causes the following to occur:

(a) Status lines are initialized and tested, and
(b) the Interrupt Test is performed.

If during initialization the operator specified Loopback Mode, the Interrupt Test is performed in both Diagnostic and Loopback Modes. If the operator specified Diagnostic Mode, the Interrupt Test is executed in diagnostic mode only.

An error message will occur if the interrupt level was not specified at program initialization.

5.2.8 LA Verb--Loop on TTY/EIA Module Test

The LA verb causes a continuous, repetitive execution of the TTYEIA Module Test. This continues until an external interruption. Entry of the LA verb is equivalent to the continual, repeated execution of the EA verb.

A count of the TTY/EIA Module Tests performed is displayed in the binary number on the on the front panel. This Loop Count is also output on the interactive device.

5.2.9 L1 Verb--Loop on Initialize and Test Status Lines

The L1 verb causes the TTY/EIA status lines to be initialized and tested. The test is repeated until the program is interrupted.

5.2.10 L2 Verb--Loop on Baud Rate Test

Entry of the L2 verb causes the following series of operations to occur:

(a) TTYEIA status lines are initialized and tested,

(b) the Baud Rate Test is performed, and is repeated continually until the program is interrupted.

Execution of the L2 verb is similar to the repetitive execution of the E2 verb. However, the initializing and testing of status lines is performed only at the beginning of the test and is not repeated. The Loop Count is output on the display panel and on the interactive device.

5.2.11 L3 Verb--Loop on Transmit, Receive, and Compare Test

The following sequence of computation results from the L3 verb:

(a) The TTYEIA status lines are initialized and tested,
(b) the Transmit, Receive, and Compare Test is performed, and is repeated continually until the program is interrupted.

Entry of the L3 verb is similar to repeatedly entering the E3 verb, except that initializing and testing of status lines is not repeated.

If the operator specified Loopback Mode when initializing, the Transmit, Receive and Compare Test will alternate between Diagnostic and Loopback Modes. If the operator specified Diagnostic Mode, the Transmit, Receive, and Compare Test will be performed only in Diagnostic Mode.

The number of Transmit, Receive and Compare Tests which have been done are indicated by the binary number on the display panel. In addition, this number is output as the Loop Count on the interactive device.

5.2.12 L4 Verb--Loop on Timing Error Test

Entry of the L4 verb causes the following sequence of computations:

(a) The TTYEIA status lines are initialized and tested,
(b) the Timing Error Test is executed, and it is repeated continually until the TTYEIA Test is interrupted.

Execution of the L4 verb is similar to the continual, repetitive

execution of the E4 verb, except that initializing/testing status lines is not repeated.

The Timing Error Tests are counted in the binary number on the display panel. This value is output as the Loop Count to the interactive device.

5.2.13 L5 Verb--Loop on Interrupt Test

Entry of the L5 verb causes the following sequence of operations to occur:

(a) TTYEIA status lines are initialized and tested, and
(b) the Interrupt Test is performed and repeated continuously.

The execution of the Interrupt Test is halted when an interrupt external to the TTYEIA Test occurs, such as entry of another verb.

Entry of the L5 verb is equivalent to repeatedly entering the E5 verb, with the exception that the initializing/testing of of status lines is not repeated.

The lights on the display panel indicate the number of times the Interrupt Test has been performed. This value, the Loop Count, is also output on the interactive device.

5.2.14 LP Verb--Loop-on-Pattern Test

Upon entering the LP verb, the message,

LOOP PATTERN? (DEFAULT= FF) -

is output on the interactive device. The operator may accept the default hexadecimal value of the bit pattern, or he may input another value. The Loop-on-Pattern Test will begin when the operator responds. The operator input bit pattern will become the default pattern when the LP verb is re-entered.

Upon beginning the test, the binary number on the display panel is set to zero. The display panel is incremented by one for each

100 transmit-receive-compare tests which occur.

The Loop Count, the total number of transmit-receive-compare tests performed, is output to the interactive device.

5.3 TTYEIA Test Software Description

The TTYEIA program consists of the following main test modules:

(a) Initialize and Test Status Lines

(b) Baud Rate Test

(c) Transmit, Receive, and Compare Test

(d) Timing Error Test

(e) Interrupt Test

(f) Loop-on-Pattern Test

The following utility routines are used by the aforegoing tests:

(a) Interrupt Service Routine

(b) Transmit, Receive, and Compare Routine

(c) Error Message Calling Sequences and Message Text Module

The programs and routines listed above are described in more detail in the following paragraphs.

5.3.1 Initializing and Testing Status Lines

This program initializes and tests status lines. The following computations are performed, in this order:

(a) The interrupt capability is disabled and the interrupt status lines are cleared. A DOCS service call is performed, which does the reset and reenables the interactive device.

(b) The transmit-in-progress, timing error, write request, and interrupt status signals are checked for correctness.

(c) The interrupt status signal is forced to go high by changing

the DCD (data carrier detect) status. This is done in diagnostic or loopback mode depending on operator input at program initialization. After verifying the interrupt line works, the new status flag interrupt signal is cleared.

(d) Each of the eight status signals is tested for its correct value.

Any variance from the expected value of a status line, in the aforegoing tests, results in an error message.

5.3.2 Baud Rate Test

The main purpose of the baud rate test is to check the timing mechanism which controls the bit transfer rate.

The baud rate test is accomplished by performing write operations as rapidly as possible over a period of time, and computing the number bits per second which were transmitted. The program writes (transmits) a character, waits until the WRQ (Write Request) signal goes high, and then writes another character, etc. The read (receive) operation is not used in performing the baud rate test.

Write (transmit) operations are executed continually for 1.25 second, a count being kept of the number of characters transmitted. If 100 or more characters were transmitted during the 1.25 second, the word count is multiplied by 8 (if 10-bit character code format if being used) or by 8.8 (if the 11-bit character code form is in effect). This value is the measured baud rate. (The number of bits per character is set by TTYEIA board jumpers, and this information is provided by the operator during program initialization.)

If fewer than 100 characters were transmitted during 1.25 second, the test is repeated for a period of ten seconds to obtain a new character count. This is the measured baud rate, if the 10-bit character code is in effect. If the 11-bit code is in effect, the ten-second character count is multiplied by 1.1 to obtain the measured baud rate.

After obtaining a value for measured baud rate, computations are made to determine whether it falls within two percent of one of the following rates: 75, 110, 300, 1200, 2400, 4800, or 9600.

If the measured rate is within two percent of one of these specified rates, the specified rate (not the exact measured rate) is output to the operator.

If the measured rate does not lie within a two percent range of one of these specified correct rates, an error message is output along with the measured baud rate. If the measured rate is a permissable rate but not the rate specified by the operator at program initialization, an error message and the baud rate are output.

A table look-up procedure is used in determining whether the baud rate is good; another table look-up is used in formatting the baud rate message.

It is important that the AC power supply frequency be correctly input at program initialization (see section 5.2.1). The time period of the baud rate test is measured on the basis of AC frequency.

The baud rate test is the same regardless of whether or not the loopback connector is used. This test does not involve looped-around data.

5.3.3 Transmit, Receive, and Compare Test

The Transmit, Receive, and Compare Test sends a character through the transmitting circuitry with a write operation, and reads the character back from the receiving circuitry with a read operation. The character has been looped around from the transmitting to the receiving circuitry, either with internal board logic or via the loopback connector attached at the EIA interface.

The write, read, and compare operations are performed numerous times, with the characters being varied. The series of characters transmitted in this test are described as follows:

(a) 256 characters, ranging from hexadecimal 00 to FF, are transmitted. The character used begins with 00 and is incremented by one for each successive transmission.
(b) Hexadecimal 33 is transmitted 50 times.
(c) Hexadecimal FF is transmitted 50 times.
(d) Hexadecimal 55 is transmitted 50 times.

Error messages are output when the character received differs from that which was transmitted.

A detailed description of the transmit, receive, and compare test computation performed for each character, and the formulation of error messages, is given in section 5.3.8, Transmit, Receive and Compare Routine.

When the loopback connector is attached, the aforegoing series of characters is tested first with the board in Diagnostic Mode, and then tested a second time with the board in Loopback Mode.

Diagnostic mode only is used if the connector is not attached.

5.3.4 Timing Error Test

The purpose of the Timing Error Test is to test the setting of the TIMERR (timing error) status signal and the XMTING (transmit-in-progress) status signal.

This test begins by clearing all interrupts and setting the real time clock for a 250 ms countdown. Five consecutive write (transmit) operations are performed. After the 250 ms period has elapsed, the timing error status signal is tested. The timing error signal should have gone high because no read commands were given to clear the receiver buffer. An error message is output if this is not the case.

The transmit-in-progress status signal is tested after each write command. An error message occurs if this signal fails to go high. In addition, the transmit-in-progress signal is checked to ensure that it has gone low after the 250 ms has elapsed.

The Timing Error Test is performed twice, with Diagnostic Mode on and with it off, if the operator is using Loopback Mode. The status lines are initialized and tested prior to each timing error test. If the operator is using Diagnostic Mode, the test is performed once with the diagnostic mode signal on.

5.3.5 Interrupt Test

The interrupt test checks the write request (WRQ), read request (RRQ), and new status flag (NSF) interrupts. The interrupt trap vector is enabled, so that control will be transferred to the TTYEIA Interrupt Service Routine.

The test begins by enabling interrupts, and clearing the read request, write request, and new status flags.

The real time clock countdown is set for 250 ms, and a command to write (transmit) a character is given. The Interrupt Service Routine maintains a count of each type of interrupt as it occurs. (See section 5.3.7 for a description of the Interrupt Service Routine.) After the 250 ms has elapsed, the program checks to verify that:

(a) One (and only one) write request interrupt occurred,
(b) one (and only one) read request interrupt occurred, and
(c) no new status flag interrupt has occurred.

Any deviation from these conditions results in an error message.

The new status flag (NSF) interrupt is tested with the following procedure. The DSR (Data Set Ready) status signal is set high by a command to the DTR (Data Terminal Ready) signal, which is looped back to DSR. The DCD (Data Carrier Detect) signal is toggled by clearing and setting the RTS (Request to Send), which is looped back to DCD. The DCD signal is toggled (on and off) twice.

The program delays for 100 ms after each change of DCD and then checks whether the new status flag interrupt has occurred. Failure

of an interrupt to occur will result in the output of an error message.

A requirement for the new status flag interrupt to occur, in Loopback Mode, is that a jumper be placed from E26 to E25, not from E26 to E27. A jumper from E26 to E27 sets RTS equal to DTR, and causes the NSF interrupt test to fail.

If the operator did not indicate the TTYEIA interrupt level at program initialization, then an appropriate message is output and the test is aborted.

If the operator indicated Loopback Mode at initialization, the Interrupt Test is executed twice (once in Diagnostic Mode and again in Loopback Mode). If Diagnostic Mode was indicated, the Interrupt Test is performed once.

5.3.6 Loop-on-Pattern Utility

The Loop-on-Pattern Utility continually performs transmit, receive, and compare tests, with the transmitted character being looped back to the receiver. The pattern, an eight-bit character, is input by the operator at the beginning of the test. The pattern remains the same for the duration of the test, which continues until an external interrupt occurs. This utility makes it possible to observe the transmitted bit pattern on an oscilloscope.

The Loop-on-Pattern is performed only in the mode for which the TTYEIA Test was initialized (Diagnostic Mode or Loopback Mode). It does not alternate between two modes.

The binary number on the display panel is set to zero at the beginning of the test. It is incremented by one for each 100 transmit, receive, and compare operations. This provides the operator with a visual verification that the test is running, and an indication of the number of transmissions done.

The loop count (number of transmissions performed) is output, in increments of 100, to the interactive device.

The Loop-on-Pattern Utility uses the Transmit, Receive, and Compare Routine (described in section 5.3.8) to perform the details of the write, read, and compare operations, and the output of error messages.

5.3.7 Interrupt Service Routine

The Interrupt Service Routine serves the purpose of testing TTYEIA board interrupts. In addition, it outputs error messages when spurious, unexpected interrupts occur.

The Interrupt Service Routine performs the following sequence of computations:

(a) It determines whether the interrupt was caused by the TTYEIA Module or by something else. If the interrupt was not from TTYEIA, an appropriate error message is output. If it was from TTYEIA, then the computations in (b) and (c) below are done.

(b) A check is made to determine whether a read request, write request, or new status flag interrupt occurred. A flag is set to indicate the type of interrupt.

(c) If the interrupt was not expected, an error message indicating that fact is output.

Upon completion of its computations, the Interrupt Service Routine transfers control back to the point in the program where the interrupt occurred.

5.3.8 Transmit, Receive, and Compare Routine

The Transmit, Receive, and Compare Routine performs the function of writing eight-bit characters through the transmission circuitry and reading them back from the receiving circuitry (where the characters are looped back, either at the EIA interface, or internally via the diagnostic mode circuitry).

This routine will output a message, explaining the error, if any of the following situations occurs:

(a) The interrupt status line fails to go high within 250 ms after a write command is issued,
(b) the write request status line fails to go high,
(c) the read request status line fails to go high, or
(d) the character input does not equal the character output.

Inputs to this routine consist of an initial character, a character increment value, and a loop count. The routine begins testing with the initial character, and increments it for each successive test. This is done until the number of characters specified in the loop count have been tested.

5.3.9 Error Message Output Routine and Message Text Module

There is a routine which contains all calling sequences for error messages. In addition, there is an independent, linkable module containing all message text; it is separate from the remainder of the program.

When an error is detected by one of the test programs, the Error

Message Output Routine is called. This routine then executes the output of the message, using text from the Message Text Module.

6.0 Messages

This section discusses messages output through the interactive device. Section 6.1 explains messages related to TTY/EIA errors, while section 6.2 itemizes header messages routinely output during tests. Those messages which apply to program initialization and operation are explained in section 5.2, Verbs for Program Execution.

6.1 Error Messages and Recommended Action

Hexadecimal Error No. | Message and Recommended Action

1 STATUS ERROR--EXPECTED=XX RECEIVED=YY

This message results from an erroneous value of a status signal after the program has initialized the TTYEIA status lines. XX is the hexadecimal value of correct status, and YY is the hexadecimal value of status as received by the test. (See Appendix C, TTY/EIA Module Input/Output at Computer Interface, for definition of status signals.) Examine that part of TTY/EIA Module logic which sets the erroneous signal.

2 INTERRUPT LINE NOT SET HIGH

If this error occurred during the Initialize/Test Status Lines Test, then a change in the state of the DCD (Data Carrier Detect) signal failed to cause the interrupt line to be set. If the error occurred in any other

test, a write request or read request either failed to occur or failed to set the interrupt line. Investigate that part of module logic which sets the interrupt status signal when a read request, write request, or new status flag signal goes high.

3 READ REQUEST LINE NOT SET IN 250 MS

The Read Request line fails to go high after a write command has been issued. If the TTYEIA Test is in Loopback Mode, check whether the Loopback Test Connector is installed. If not, use the IT verb to re-initialize the program in Diagnostic Mode, and repeat the test. If the test still fails, examine that part of Module logic which sets the Read Request signal.

4 TIMING ERROR LINE NOT SET HIGH

Several characters were loaded into the receive buffer register and no read operation was performed by the computer; the Timing Error status signal failed to go high within 250 ms. Examine that part of TTYEIA Module logic which sets the Timing Error status line.

5 TRANSMIT/RECEIVE ERROR--OUTPUT= XX INPUT= YY

The eight-bit character transmitted by a write command does not equal the eight-bit character received by a corresponding read command, in a transmit-receive-compare operation.

Recommended action:

(a) If the test is in Loopback Mode, check whether the Loopback Test Connector is attached.

(b) Check the jumper configuration on the TTYEIA board. The receive and transmit code formats should be both 10-bit code or both 11-bit code. Also, a jumper should be set for "enable receive". (See section 3.2, TTYEIA Jumper Configurations.)

(c) If the transmit-receive-compare operation works correctly in Diagnostic Mode but not in Loopback Mode, then examine the module logic which is not tested in Loopback Mode. Also, check the jumper configuration to verify that the EIA interface, not the TTY interface, is being used.

(d) If the transmit-receive-compare test fails in both Loopback and Diagnostic Modes, examine the circuitry which stores and retrieves the eight-bit character.

6 UNEXPECTED TTYEIA INTERRUPT

A spurious TTYEIA interrupt has occurred. The interrupt status line went high and the interrupt trap vector executed. This interrupt should not have been caused by any action of the

TTYEIA Test program. Examine that part of TTYEIA board logic which sets the interrupt status line.

7 CHECK BAUD RATE JUMPERS--BAUD RATE= XXXX

The baud rate calculated by the test does not equal the operator input baud rate. It is one of the baud rates which can be obtained from the various jumper configurations. Examine the

TEXAS INSTRUMENTS
INCORPORATED
DIGITAL SYSTEMS DIVISION
POST OFFICE BOX 2909 AUSTIN, TEXAS 78769