TITLE: CR80 AMOS
TDX TEST SYSTEM

DOCUMENT NO: CSS/714/USM/0066

PREPARED BY: Tim Lund Markussen

APPROVED BY: Jørgen Høg

AUTHORIZED BY: Jørgen Høg

DISTRIBUTION:

ISSUE:	1	2						
DATE:	810327	810427						

400-571-2

CR80 AMOS TDX TEST SYSTEM	sign/date TLM/810327	page i
	repl	project

PAGE RECORD AND ISSUE LOG.

PAGE	ISSUE							
	1	2	3	4	5	6	7	8
01								
02								
03								
04								
05								
06								
07								
08								
09								
10								
11								
12								
13								
14								
15								
16								
17								
18								
19								
20								
21								
22								
23								
24								
25								
26								
27								
28								
29								
30								
31								
32								
33								

PAGE	ISSUE							
	1	2	3	4	5	6	7	8
34								
35								
36								
37								
38								
39								
40								
41								
42								
43								
44								
45								
46								
47								
48								
49								
50								
51								
52								
53								
54								
55								
56								
57								
58								
59								
60								
61								
62								
63								
64								
65								
66								

PAGE	ISSUE							
	1	2	3	4	5	6	7	8
67								
68								
69								
70								
71								
72								
73								
74								
75								
76								
77								
78								
79								
80								
81								
82								
83								
84								
85								
86								
87								
88								
89								
90								
91								
92								
93								
94								
95								
96								
97								
98								
99								
100								

ISSUE	DATE	PREPARED BY	APPROVED BY	AUTHORIZED BY
1	810327	TLM	JHØ	JHØ
2	810427			

A

TABLE OF CONTENTS

		PAGE
1.	SCOPE	1
1.1	Introduction	1
2.	APPLICABLE DOCUMENTS	2
3.	OVERVIEW	3
3.1	Environment	3
3.2	Functions performed	4
3.3	Syntactical Rules	5
4.	FUNCTION DESCRIPTION	7
4.1	Assign	8
4.2	Deassign	9
4.3	Create	10
4.4	Dismantle	11
4.5	Read	12
4.6	Initread	13
4.7	Append	14
4.8	Initappend	15
4.9	Driver define	16
4.10	Console In	17
4.11	Console Out	18
4.12	Define pattern	19
4.13	Dump Buffer or pattern	20
4.14	Delay	21
4.15	Procedure definition	22
4.16	Repeat	23
4.17	Cancel	24
4.18	List operations	25
4.19	Wait for init. operations or timeout	26
5.	ACTIVATION	27
6.	ERROR MESSAGES	28
7.	EXAMPLES	29

1. SCOPE

This document describes the use of CR80 AMOS TDX Test System.
This issue applies to version 0102 of the program.

1.1 Introduction

The TDX Test System is a program which eases the access to the CR80 TDX Host Interface, via the CR80 AMOS TDX Driver.
All access to the TDX system is performed using the AMOS I/O-system and the AMOS TDX Driver.

The program is able to interpret commands from either a terminal or an AMOS disk file.

Input commands may be:

- TDX Driver commands like
  ASSIGN, DEASSIGN, CREATE etc.

- Commands making the program repeat any of the other commands a number of times.

- Commands making the program dump data buffers, or check contents of data buffers.

- Commands making the program change from file input to terminal input, or file output to terminal output.

2. APPLICABLE DOCUMENTS

[1] CR80 Minicomputer Handbook
CSD/HDBK/0001.

[2] CR80 AMOS Command Interpreter.
User's Manual
CSS/381/USM/0037.

[3] CR80 AMOS Terminal Operating System.
User's Manual.
CSS/380/USM/0026.

[4] CR80 AMOS TDX Driver
Product Specification
CSS/314/PSP/0013.

3. OVERVIEW

This section contains:

- Software and Hardware requirements to facilitate use of the program.

- Syntactical rules for input commands.

3.1 Environment

Minimum hardware system:

A standard software development system

With at least 64K words of RAM, and a TDX Host Interface.

Minimum software system:

A standard software development system and a TDX Driver.

3.2 Functions performed

The functions performed are the following:

- Communication to the TDX driver to perform:
  - assign TDX driver
  - deassign TDX driver
  - create log-lines
  - dismantle log-lines
  - read, init-read
  - append, init-append

- Definition of the TDX driver name to be used by the above mentioned commands.
- Switch from file input to terminal input.
- Switch from file output to terminal output.
- Definition of testpatterns to be used for read and append.
- Dump of data buffers as hexadecimal words and ASCII text.
- Delay execution for a specified time.
- Definition of procedures with parameters, containing any of the other commands.
- Definition of repeat statements, making the program repeat commands or procedure calls a number of times.
- Checking of data received by read or init-read against a test pattern.

Input to the program is, either an AMOS file containing commands, or commands entered directly from a terminal.

Output from the program is either to an AMOS file or to a terminal.

3.3 Syntactical Rules

This section contains the syntactical rules for input commands, denoted in EBNF. Reserved words are written in capital letters.

```
<COMMANDFILE>      -->  <PROCDECLS> <BODY> <CR>
                   -->  <BODY> <CR>
                   -->  END <CR>

<PROCDECLS>        -->  <PROCDECLS> <PROCDECL>
                   -->  <PROCDECL>

<PROCDECL>         -->  PROCEDURE <PROCNAME> <PARAMLIST> <CR> <BODY> <CR>
                   -->  PRODEDURE <PROCNAME> <CR> <BODY> <CR>


<PROCNAME>         -->  <IDENTIFIER>

<PARAMLIST>        -->  ( <PARAMETERS> )

<PARAMETERS>       -->  <PARAMETERS> ; <IDENTIFIER>
                   -->  <IDENTIFIER>

<BODY>             -->  BEGIN <CR> <STATEMENTS> END

<STATEMENTS>       -->  <STATEMENTS> <STATEMENT>
                   -->  <STATEMENT>

<STATEMENT>        -->  <PROCEDUPECALL>
                   -->  <REPEATSTATEMENT
                   -->  <SIMPLECOMMAND>

<PROCEDUPECALL>    -->  <PROCNAME> ( <ACTUALPARAMS> ) <CR>
                   -->  <PROCNAME> <CR>

<ACTUALPARAMS>     -->  <ACTUALPARAMS> , <CONST_OR_VAR>
                   -->  <CONST_OR_VAR>

<REPEATSTATEMENT>  -->  REPEAT <REPEATS> <CR> <BODY> <CR>
                   -->  REPEAT <REPEATS> <STATEMENT>


<REPEATS>          -->  <CONST_OR_VAR>  "TIMES TO REPEAT"

<SIMPLECOMMAND>    -->  <DRIVERDEFINECOMMAND>
                   -->  <ASSIGNCOMMAND>
                   -->  <DEASSIGNCOMMAND >
                   -->  <CREATECOMMAND>
                   -->  <DISMANTLECOMMAND>
                   -->  <READCOMMAND>
                   -->  <INITREADCOMMAND >
                   -->  <APPENDCOMMAND>
                   -->  <INITAPPENDCOMMAND>
                   -->  <CONSOLECOMMAND>
                   -->  <DEFINECOMMAND>
                   -->  <DUMPCOMMAND>
                   -->  <DELAYCOMMAND>
```

```
<DRIVERDEFINECOMMAND>   -->  DRIVER = <STRING> <CR>

<ASSIGNCOMMAND>    -->  ASSIGN <CR80ADR> <HOSTNO> <CR>

<DEASSIGNCOMMAND>  -->  DEASSIGN <CR>

<CREATECOMMAND>    -->  CREATE <PROTOCOL> <CRID> <SPEED> <MAXPACKET> <CR>

<DISMANTLECOMMAND>-->  DISMANTLE <CRID> <CR>

<READCOMMAND>      -->  READ <READPARAMS> <CR>

<INITREADCOMMAND>  -->  INITREAD <READPARAMS> <CR>

<READPARAMS>       -->  <CRID> <NOOFBYTES> EXPECT PATTERN <PATTERNNO>  ( DUMP )
                   -->  <CRID> <NOOFBYTES>  ( DUMP )

<APPENDCOMMAND>    -->  APPEND <APPENDPARAMS> <CR>

<INITAPPENDCOMMAND> -->  INITAPPEND <APPENDPARAMS> <CR>

<APPENDPARAMS>     -->  <CRID> <NOOFBYTES> PATTERN <PATTERNNO>
                   -->  <CRID> <NOOFBYTES> ( <INTEGERS> )

<INTEGERS>         -->  <INTEGERS> <INTEGER>
                   -->  <INTEGER>

<INTEGER>          -->  <CONSTANT>

<CONSOLECOMMAND>   -->  CONSOLE IN <CR>
                   -->  CONSOLE OUT <CR>

<DEFINECOMMAND>    -->  DEFINE PATTERN <PATTERNNO> = <PATTERN> <CR>

<DUMPCOMMAND>      -->  DUMP BUFFER <BUFFERNO> FROM <STARTADDR> <CR>
                   -->  DUMP BUFFER <BUFFERNO> FROM <STARTADDR> TO <ENDADDR><CR>

                   -->  DUMP PATTERN <PATTERNNO> <CR>

<DELAYCOMMAND>     -->  DELAY <TIME> <CR>

<LISTCOMMAND>      -->  LIST OPERATIONS <CR>

<CANCELCOMMAND>    -->  CANCEL CRID <CRID> <CR>
                   -->  CANCEL OPERATION <OPREF> <CR>

<WAITINITCOMMAND> -->  WAITINIT <MAXTIME> <CR>
```

```
<CR80ADR>          -->  <CONST_OR_VAR>
<HOSTNO>           -->  <CONST_OR_VAR>
<PROTOCOL>         -->  <CONST_OR_VAR>
<CRID>             -->  <CONST_OR_VAR>
<NOOFBYTES>        -->  <CONST_OR_VAR>
<SPEED>            -->  <CONST_OR_VAR>
<MAXPACKET>        -->  <CONST_OR_VAR>
<PATTERNNO>        -->  <CONST_OR_VAR>
<BUFFERNO>         -->  <CONST_OR_VAR>
<STARTADDR>        -->  <CONST_OR_VAR>
<ENDADDR>          -->  <CONST_OR_VAR>
<TIME>             -->  <CONST_OR_VAR>
<MAXTIME>          -->  <CONST_OR_VAR>
<OPREF>            -->  <CONST_OR_VAR>
<TIME>             -->  <CONST_OR_VAR>
<CONST_OR_VAR>     -->  <IDENTIFIER>
                   -->  <CONSTANT>
<PATTERN>          -->  <INTEGERS>
                   -->  <STRING>
<CR>               -->  (:10:)  "CARRIAGE RETURN"
```

In a procedure, a <const-or-var> may be the name of a parameter, or an integer constant. In the 'main program' it must be an integer constant.
Identifiers may contain letters, numbers and underline, but must start with a letter.
Constants may be decimal constants or hexadecimal constants preceeded by '#'.

4. FUNCTION DESCRIPTION

This section contains a detailed description of the functions performed by the program.

This includes:

- input command syntax.
- possible output messages.
- examples of input and output.
- possible output messages.

NOTE:

In general, correct execution of a command will not invoke any output, except an arrow ('--> ') which indicates that the program expects a new command. This includes commands which generate communication to the AMOS I/O system, where the I/O system returns an OK answer.

4.1 Assign

Input Syntax:

ASSIGN <CR80 address> <host number>

Function performed:

An assign command is issued to the TDX-driver.
The CR80 address is:
(Hardware device number *4)+priority.
Host number is the TDX device number of the host.

Output:

If OK: no particular message.
If not OK:

*** BAD COMPLETION CODE*** CC=<CC>

<CC> meanings may be looked up in
[4] section 4.1.4.

Examples:

ASSIGN 124 3 <CR>

In procedure with formal parameters:
ASSIGN CR80ADR HOST <CR>

4.2 Deassign

Input syntax:

DEASSIGN

Function performed:

A deassign command is issued to the TDX driver.

Output:

If OK: no particular message.

If not OK:

*** BAD COMPLETION CODE *** CC=<CC>

<CC> meanings may be looked up in [4] section 4.1.4.

Example:

DEASSIGN <CR>

4.3 Create

Input syntax:

CREATE <protocol><crid><speed><max packet>

Function performed:

A create-log-line command is issued to the TDX driver.

<protocol> is the protocol index in the TDX driver.

<crid> is an identification of the destination subdevice.

<speed> is a speed index for the line.

<maxpacket> defines the maximum packet size for the line.

Output:

If OK: no particular message.

If not OK:

*** BAD COMPLETION CODE *** CC=<CC>

<CC> meanings may be looked up in

[4] section 4.1.4.

Examples:

CREATE 0 # 343 7 16 <CR>

CREATE 1 # 502F # 808 128 <CR>

4.4 Dismantle

Input syntax:

DISMANTLE <crid>

Function performed

A dismantle of the actual crid is issued to the TDX driver.

Output:

If OK: no particular message.

If not OK:

*** BAD COMPLETION CODE *** CC=<CC>

<CC> meanings may be looked up in [4] section 4.1.4.

Examples:

```
DISMANTLE  #343   <CR>
DISMANTLE  #502F  <CR>
```

4.5 Read

Input syntax:

1. READ <crid><no of bytes> {DUMP}
2. READ <crid><no of bytes> EXPECT PATTERN<n> {DUMP}

Function performed:

A read command is issued to the TDX driver. Then the program waits for this command to finish. If a pattern is expected, and the read completed successfully, the program checks the received data against the specified test pattern and reports the result. If DUMP is specified, the received data is output at completion.

Output:

If OK: no message

If bad completion:

*** BAD COMPLETION CODE *** CC=<CC>

If less transferred than requested:

*** REQUESTED *** <x> BYTES
*** TRANSFERRED*** <y> BYTES

If expected data not equal to received data:

*** DATA CONFLICTS ***
*** FIRST IN BYTE *** <x>
*** WRONG BYTES *** <y>

Examples:

READ # 343 2048 <CR>
READ # 502F 2048 EXPECT PATTERN 5<CR>

4.6 Initread

Input syntax:

1. INITREAD <crid><no of bytes> {DUMP}
2. INITREAD <crid><no of bytes> EXPECT PATTERN<n> {DUMP}

Function performed:

An init-read command is issued to the TDX driver. The program does not wait for completion, however, the completion of init-operations is checked approximately once every 100 msec's. If a specific pattern is expected, the program checks the data when the operation completes.

If DUMP is specified, the received data is output at completion.

Output:

At completion:

*** INIT-READ COMPLETED *** CRID =<crid>

If bad completion:

*** BAD COMPLETION CODE *** CC=<CC>

If less transferred than requested:

*** REQUESTED *** <x> BYTES
*** TRANSFERRED *** <y> BYTES

If expected data not equal to received data:

*** DATA CONFLICTS ***
*** FIRST IN BYTE *** <x>
*** WRONG BYTES *** <y>

Examples:

INITREAD # 343 2048
INITREAD # 502F 200 EXPECT PATTERN 5

4.7 Append

Input syntax:

1. APPEND <crid><no of bytes> (<integers>)
2. APPEND <crid><no of bytes> PATTERN <n>

Function performed:

An append command is issued to the TDX driver. The data to be output is either the integers in the brackets or the pattern identified by <n>.

Output:

If OK: no message

If bad completions:

*** BAD COMPLETION CODE *** CC=<CC>

If less transferred than requested:

*** REQUESTED *** <x> BYTES
*** TRANSFERRED *** <y> BYTES

Examples:

```
APPEND #343  2048 (1 2 3 4 5)
APPEND # 502F 128 PATTERN 7
APPEND # 502F   1 (# CO)
```

4.8 Initappend

Input syntax:

1. INITAPPEND <crid><no of bytes> (<integers>)
2. INITAPPEND <crid><no of bytes> PATTERN<n>

Function performed:

An init-append command is issued to the TDX driver. The program does not wait for completion, however, completion of init-operations is checked approximately once every 100 msec's.
The data to be output is either the integers in the brackets or the pattern identified by <n>.

Output:

At completion:

*** INIT-APPEND COMPLETED *** CRID=<crid>

If bad completion:

*** BAD COMPLETION CODE *** CC=<CC>

If less transferred than requested:

*** REQUESTED *** <x> BYTES
*** TRANSFERRED *** <y> BYTES

Examples:

```
INITAPPEND #343  2048 (1 2 3 4 5)
INITAPPEND #502F  128 PATTERN 7
INITAPPEND #502F    1 (#CO)
```

K

4.9 Driver define

Input syntax:

DRIVER = <string>

Function performed:

When this command has been executed, all commands invoking communication to the TDX driver, will be issued to the driver <string>.
The driver name may be redefined at any time.
The default driver name is
'TDXOOO-INTR'.

Output:

Nothing.

Example:

DRIVER = 'TDXOO1-FILE' <CR>

4.10 Console in

Input syntax:

CONSOLE IN

Function performed:

When this command is executed, input continues from the terminal from which the program was started.
It should be quite useful, in cases where procedures have been defined in a file and the user wants to call these procedures from his terminal afterwards.

Output:

On the terminal:
TDX TEST CONTINUING FROM CONSOLE
-->

Example:

In the file:

```
PROCEDURE AS
BEGIN ASSIGN 124 3
DELAY 600
END
BEGIN
AS
CONSOLE IN
```

Then the procedure AS may be used from the terminal.

4.11 Console out

Input syntax:

CONSOLE OUT

Function performed:

When this command is executed, output switches from a file to the terminal from which the program was started.

Output:

On the terminal

TDX TEST CONTINUING TO CONSOLE

-->

Example:

CONSOLE OUT.

4.12 Define pattern

Input syntax:

DEFINE PATTERN <n> = <integers>
DEFINE PATTERN <n> = <string>

Function performed:

This command is used to define testpatterns to be used as output for append, initappend or as expected input for read, initread.
The number n must be in the range 1..10.

Output:

Nothing.

Examples:

```
DEFINE PATTERN 1 = #1 #101 #1203
DEFINE PATTERN 7 = 'TDX TEST (:10:)'.
```

4.13 Dump buffer or pattern

Input syntax:

DUMP BUFFER <n> FROM <x> {TO <y>}
DUMP PATTERN <x>

Function performed:

Buffer dump:

This command is used to dump one of the buffers used for input or output.

<n> identifies the buffer number
buffer 1 is used for read
" 2 - " - append
" 3 - " - initread
" 4 - " - initappend

<p> identifies the start address within the buffer.

<y> identifies the stop address

The buffer size is 2048 words.

To stop the dump one must 'break' to the process.

Pattern dump:

The specified pattern (64 words) content is output.

Output:

Buffer or pattern contents as hexadecimal words, and buffer contents in readable ASCII.
In the ASCII field the bytes are swapped to make it readable for a human being.

Example:

```
DUMP BUFFER 2 FROM 0 .
DUMP BUFFER 3 FROM 0 TO 256
DUMP PATTERN 4
```

```
->DUMP BUFFER 2 FROM 0
#0000L #4454 #2058 #4554 #5453 #5320 #5359 #4554 #204D  TDX TEST SYSTEM
#0008L #4241 #4443 #4645 #4847 #4A49 #4C4B #4E4D #504F  ABCDEFGHIJKLMNOP
#0010L #A7A7 #A7A7 #A7A7 #A7A7 #A7A7 #A7A7 #A7A7 #A7A7
#0018L #A7A7 #A7A7 #A7A7 #A7A7 #A7A7 #A7A7 #A7A7 #A7A7
#0020L #A7A7 #A7A7 #A7A7 #A7A7 #A7A7 #A7A7 #A7A7 #A7A7
#0028L #A7A7 #A7A7 #A7A7 #A7A7 #A7A7 #A7A7 #A7A7 #A7A7
```

4.14 Delay

Input syntax:

DELAY <time>

Function performed:

When this command is executed, the program waits for the specified time before execution is resumed. The time waited is in slots of 100 msec's, so DELAY 10 will make the program wait for 1 second.

Output:

Nothing.

Example:

```
DELAY  600   "wait 1 minute".
```

4.15 Procedure definition

Input syntax:

1. PROCEDURE <name> (<formal parameters>)
   BEGIN
       <commands>
   END
2. PROCEDURE <name>
   BEGIN
       <commands>
   END

Function performed:

When a procedure definition is met, the commands in it are compiled, and not executed before the procedure is called by its name.
If the procedure has formal parameters these may be used by the commands.
The procedure code is saved until the program terminates.

Output:

There is no particular output for the procedure, but the commands in it may cause output when it is executed.

Example:

```
FM: TDXTST
TDX TEST SYSTEM VER.810327.
-->PROCEDURE HOST_TO_HOST(HOST; BYTES)
-->BEGIN
-->    CREATE 0 HOST 7 128              "CREATE THE LINE"
-->    INITREAD HOST BYTES              "INIT-READ THE BYTES"
-->    APPEND   HOST BYTES (0 1 2 3 4)" APPEND SOME BYTES"
-->END "HOST_TO_HOST"
-->BEGIN "EXECUTION"
-->    HOST_TO_HOST(#343, 128)
*** BAD COMPLETION CODE *** CC = #020D
*** BAD COMPLETION CODE *** CC = #0203
*** REQUESTED   *** 128 BYTES.
*** TRANSFERRED *** 0 BYTES.
*** BAD COMPLETION CODE *** CC = #0203
```

4.16 Repeat

Input syntax:

1. REPEAT <n> <command>
2. REPEAT <n>
   BEGIN
   <commands>
   END

Function performed:

When a repeat command is met, the commands in it are compiled, and executed when the 'END' is met. If the repeat is used in a procedure the commands are executed when the procedure is called.
Repeating may be stopped by making a 'break' to the TDX TEST process.

Output:

Only the output from the commands in it, or if it is stopped:

```
*** REPEATING BREAKED ***
    REPEAT PERFORMED <x> TIMES
```

Examples:

```
REPEAT 300 APPEND #502F 50 (O)
REPEAT 300
BEGIN
INITREAD #343  256
APPEND   #343  256  PATTERN 3
END
```

4.17 Cancel

Input syntax:

1. CANCEL OPERATION <operation>
2. CANCEL CRID <crid>

Function performed:

CANCEL OPERATION:
A cancel of the specified operation is issued to the TDX Driver.
<operation> is an identification of an init-read operation previously issued to the TDX Driver. Operation identifications may be output by the command LIST OPERATIONS.

CANCEL CRID:
Using this command, the program issues a cancel of all not completed init-read operations, previously issued to the TDX Driver, using the specified CRID.

Output:

For each cancelled operation the following message is output:
OPERATION <operation> ON <crid> CANCELLED

Examples:

CANCEL OPERATION #4024 <cr>
CANCEL CRID #120F

4.18 List operations

Input syntax:

LIST OPERATIONS

Function performed:

A list of not completed init-read and init-append operations is output. The list contains the following fields:

CRID : CRID of the operation

INIT-KIND: Operation kind (read or append)

OPREF : Logical identification of the operation. This is the field to use by CANCEL OPERATION.

BYTES : Number of bytes requested to input or output.

BUFFER : Buffer number, from which data is taken, or in which data is delivered.

Example:

LIST OPERATIONS

Output

CRID	INIT-KIND	OPREF	BYTES	BUFFER
#343	READ	#4028	#0100	3
#120F	APPEND	#4024	#0006	4

4.19 Wait for init operations or timeout

Input syntax:

WAITINIT <maxtime>

Function performed:

This command makes the program wait until a previously submitted init-read or init-append is completed, or until <maxtime> passes.
<maxtime> is in slots of 100 msec's.
If all init-operations are completed when execution of this command starts, no waiting is performed.

Example:

WAITINIT 600 <CR>
Waits for an init-operation to complete or one minute to pass.

5. ACTIVATION

The program call syntax is like this:

TEST_TDX {I: <file-id>} {O:<file-id>} <CR>

I: <file-id> is the file id of the input file.
If omitted, current input is used.

O: <file-id> is the file id of the output file.
If omitted, current output is used.

At call, system directory must contain the file 'TDXTEST_SYNTAX.A'.
otherwise the program will terminate.

6. ERROR MESSAGES

In case of syntax errors in the input file, the program will display a '?' and wait for a new command.

If a procedure is called, but not defined, the program will display:

*** UNDECLARED PROCEDURE ***

and wait for a new command.

If a parameter is used, but not declared, the program will display:

*** UNDECLARED PARAMETER ***

and wait for a new command. The current command is omitted.

Completion codes from communication with the TDX driver may be looked up in [4] section 4.1.4.

7. EXAMPLES

Example 1:

Simple loop from a host back to itself.

Contents of input file:

```
PROCEDURE HOST_TO_HOST
BEGIN
   INITREAD #343 30
   APPEND   #343 20 (0)
END
BEGIN
   ASSIGN 124 3
   DELAY 650
   CREATE 0 #343 7 128
   HOST_TO_HOST
   CONSOLE OUT
   CONSOLE IN
END
```

Related output:

```
TDX TEST SYSTEM VER.810402.
-->PROCEDURE HOST_TO_HOST
-->BEGIN
-->   INITREAD #343 30
-->   APPEND   #343 20 (0)
-->END
-->BEGIN
-->   ASSIGN 124 3
-->   DELAY 650
-->   CREATE 0 #343 7 128
-->   HOST_TO_HOST
-->
*** INIT-READ   COMPLETED ***
   CONSOLE OUT
```

K

Example 2:

Loop from host back to itself, repeated a number of times:

Contents of input file:

```
PROCEDURE HOST_TO_HOST
BEGIN
   INITREAD #343 30
   APPEND   #343 20 (0)
END
BEGIN
   ASSIGN 124 3
   DELAY 650
   CREATE 0 #343 7 128
   REPEAT 130 HOST_TO_HOST
   CONSOLE OUT
   CONSOLE IN
END
```

Related output:

```
TDX TEST SYSTEM VER.810402.
-->PROCEDURE HOST_TO_HOST
-->BEGIN
-->   INITREAD #343 30
-->   APPEND   #343 20 (0)
-->END
-->BEGIN
-->   ASSIGN 124 3
-->   DELAY 650
-->   CREATE 0 #343 7 128
-->   REPEAT 130 HOST_TO_HOST
*** REQUESTED   *** 20 BYTES.
*** TRANSFERRED *** 0 BYTES.
*** BAD COMPLETION CODE *** CC = #0607
*** REQUESTED   *** 20 BYTES.
*** TRANSFERRED *** 0 BYTES.
*** BAD COMPLETION CODE *** CC = #0607
-->   CONSOLE OUT
```

Comments:

This test shoved that the host started to time out after a repeat of 128 times, which is an error.

K

Example 3:

Host to host loop at 7 channels simultaneously, using procedures with parameters.

```
PROCEDURE HOST_TO_HOST(CRID)
BEGIN
   INITREAD CRID 30
   INITAPPEND   CRID 30 (0 1 2 3 4)
END
PROCEDURE CREATE_LINES
BEGIN
   CREATE 0 #303 7 128
   CREATE 0 #313 7 128
   CREATE 0 #323 7 128
   CREATE 0 #333 7 128
   CREATE 0 #343 7 128
   CREATE 0 #353 7 128
   CREATE 0 #363 7 128
END
PROCEDURE REPEAT_TEST(TIMES)
BEGIN
   REPEAT TIMES
   BEGIN
      HOST_TO_HOST(#303)
      HOST_TO_HOST(#313)
      HOST_TO_HOST(#323)
      HOST_TO_HOST(#333)
      HOST_TO_HOST(#343)
      HOST_TO_HOST(#353)
      HOST_TO_HOST(#363)
   END   " REPEAT "
END " REPEAT_TEST "
BEGIN
   ASSIGN 124 3
   DELAY 650
   CREATE_LINES
   HOST_TO_HOST(#303)
   HOST_TO_HOST(#313)
   HOST_TO_HOST(#323)
   HOST_TO_HOST(#333)
   HOST_TO_HOST(#343)
   HOST_TO_HOST(#353)
   HOST_TO_HOST(#363)
   CONSOLE OUT
   CONSOLE IN
END
```

Related output:

```
TDX TEST SYSTEM VER.810402.
-->PROCEDURE HOST_TO_HOST(CRID)
-->BEGIN
-->    INITREAD CRID 30
-->    INITAPPEND   CRID 30 (0 1 2 3 4)
-->END
-->PROCEDURE CREATE_LINES
-->BEGIN
-->    CREATE 0 #303 7 128
-->    CREATE 0 #313 7 128
-->    CREATE 0 #323 7 128
-->    CREATE 0 #333 7 128
-->    CREATE 0 #343 7 128
-->    CREATE 0 #353 7 128
-->    CREATE 0 #363 7 128
-->END
-->PROCEDURE REPEAT_TEST(TIMES)
-->BEGIN
-->    REPEAT TIMES
-->    BEGIN
-->       HOST_TO_HOST(#303)
-->       HOST_TO_HOST(#313)
-->       HOST_TO_HOST(#323)
-->       HOST_TO_HOST(#333)
-->       HOST_TO_HOST(#343)
-->       HOST_TO_HOST(#353)
-->       HOST_TO_HOST(#363)
-->    END  " REPEAT "
-->END " REPEAT_TEST "
-->BEGIN
-->    ASSIGN 124 3
*** BAD COMPLETION CODE *** CC = #0616
-->    DELAY 650
-->    CREATE_LINES
-->    HOST_TO_HOST(#303)
-->
*** INIT-READ    COMPLETED ***

*** INIT-APPEND COMPLETED ***
   HOST_TO_HOST(#313)
-->
*** INIT-READ    COMPLETED ***

*** INIT-APPEND COMPLETED ***
   HOST_TO_HOST(#323)
-->    HOST_TO_HOST(#333)

*** INIT-READ    COMPLETED ***
```

Comments:

The error in 'ASSIGN' occured bacause the TDX driver was assigned previously.