FIELD SERVICE

GUIDE TO PLURIBUS DIAGNOSTICS

REV B

January 30, 1983

Pete Marti
Technical
Support

Table of Contents

1 INTRODUCTION

1.1 Purpose

The Field Service Guide to Pluribus Diagnostics has been generated to:

1- Reduce the number of diagnostics in existence to an adequate and supportable quantity.

2- Fully explain and simplify running the diagnostics.

3- Interpret the error information generated by the diagnostics.

4- Reduce the indecision of "which one do I run" by forming a complete document of "the latest and greatest" diagnostics needed to support the Pluribus product line.

Many diagnostics are obsolete, either because they no longer apply or, a later developed diagnostic performs the same function. From a Field Service point of view, we must constantly review and analyze our diagnostics to ensure that they support our maintenance philosophy and aid in installing and maintaining pluribus systems. We must also provide a document that explains the supported diagnostics/software and how to get the most out of each.

We must point out that the term "diagnostic" is being used loosely, because the present software does not diagnose problems, but is actually an exerciser of some subset of the system configuration and will not report a failure as who failed, but what operation failed.

Future REVS to this document will include not only new or replacement diagnostics, but troubleshooting hints/shortcuts. Forward any suggestions that any of you users of the diagnostics have to improve this document to the Field Service Technical Support Staff, BBNCC.

1.2 Pluribus Diagnostics And Their Use

Up to this point in time, many diagnostics at BBNCC have been written, packaged and distributed individually or linked together under a new name for a specific purpose. Pluribus diagnostics exist in two main categories as depicted below, Single-bus and Multi-bus. Single-bus diagnostics are used on Single-bus machines (VDA,PLI) where the processors, memory, and I/O are on the same bus. Multi-bus diagnostics are used on Multibus-bus machines where the processors, memory, and I/O may or may not share the same bus. Some of the diagnostics are present on both lists, because their operation is not dependent on the configuration.

If these diagnostics do not exist on the proper media for that site or they do not match the REV level shown (paper-tape, cassette) order deficiencies or worn media from the list below. An explaination of the REV designation os in the following section.

1.3 BBNCC Diagnostics And Related Software

Single Bus	Multi- Bus
PWRTST.012	PWRTST.012
TTYTST.012	TTYTST.012
INV-SBA.003	INV-MB.004
MEMCHK-SBA.006	MEMCHK-MB.007
PIDTST.031	PIDTST.031
DEVTST-SBA.003	DEVTST-MB.003
DMACHK-SBA.003	DMACHK-MB.002
RLDCHK.002	RLDCHK.002
RTCTST.010	RTCTST.010
AMLTST.067	AMLTST.067
RPLI-257.001	MLCTST.010
BVPLI-257.001	MSPMTST.058
GI34.60429	EXHIT.007
CONS.51209	HIT.233
MEMCLR.114	MEMCLR.114
PSTOPL.003	PSTOPL.003
DDT-12.328	DDT-14.111
	DDT-41.516

1.4 BBNCC Diagnostic Tapes And Cassettes

Each Pluribus diagnostic is punched individually on paper tape or are all contained on a cassette. Pluribus diagnostic tapes are stored at each Pluribus IMP, TIP X.25 TIP, PLI and VDA site. These are for the use of field service personel. Each kind of site requires a different set of diagnostic tapes or cassette, according to the SRN configuration. PLI and VDA sites require a set of the Single-bus diagnostics. The IMP and TIP X.25 TIP require a set of the Multi-bus diagnostics.

The diagnostic names have been changed to some extent at release of this document to not only be able to tell the name of the diagnostic, but also the REV level of the diagnostic.

EXAMPLE: INV-MB.004

This is the name of the Inventory(INV) diagnostic for a multi-bus(-MB) system and is at REV level 4. All tapes as of release of this document will be in this format. The header on the paper-tape generated diagnostics can be read sideways and will coincide with this format.

A future REV of this document will include information as to what should be found at the various types of sites, such as IMP,VDA, etc.

The cassettes will still have the 1 bel on their case. Future diagnostics will hopefully will be internally labeled and print out what they are and REV level at start-up.

1.5 Synopses Of The Diagnostics

The follow pages give a synopses of each diagnostic. They are intended to help a person decide whether or not the particular diagnostic is appropriate to his need. But, they are not a substitute for reading the instructions for the diagnostic which can be found in each diagnostic detailed operation section. Special attention to the default parameter sections will save a lot of time in determining why the diagnostic doesn't run the first or subsequent tries.

Most of the diagnostics do not require the use of DDT, but DDT can be loaded with the diagnostics. The tendency is to not load DDT unless extensive program location changes are required. One or two changes can be made rather easily after the diagnostic is loaded via the control panel. Restart the program by pushing "RESET" then "ATTEN" on the control panel.

Special attention should be made to the default addresses of the diagnostics. Many of them are not the addresses in present configurations. This will be corrected in future REVS of the diagnostics to make the diags easier to run.

1.6 DDT

DDT is used as a mini-monitor or operating system for some of the more old fashioned diagnostics. It is also useful by itself for general poking around a system. But many of the more recent diagnostics e.g., EXHIT have basic DDT features built into them. The result is that DDT as a main program is becoming a thing of the past. There are many versions of DDT each one supporting a different machine configuration. Which versions are which are documentented in the beginning of the DDT listing. The version used with a particular diagnostic is documented within that diagnostic write-up.

1.7 DEVTST

Purpose

The purpose of the DEVTST program is to find and test the DMA devices in a PLURIBUS system, and to print detailed information on the TTY.

In the most general case, the program finds all DMA devices, and for each device it tries to transfer data from every common memory page to itself using the device.

For every transfer unique information is stored in the transmit buffer (normally, the lower half of the page), the destination buffer (normally, the upper half of the page) is cleared, and the device is activated. The process is repeated twice (the results of the first transfer are ignored).

A program timer is set for each activation of the device. The transfer is terminated in one of two ways:

1. The two bits in the device register block- Transmit active and Receive active- are both reset before the program timer reaches the end of its period, or

2. The timer times out with one of the active bits still set.

In any case, after the second transfer is terminated, the program prints the results of the transfer.

The results of the transfer are printed in a table and include information about the status of different error bits in the device registers block, the success of the transfer, the success of writing to the PID, etc.

The program interacts with the user and allows him to select specific I/O devices, to select specific memory busses, to specify which memory pages are to be ignored, to specify which I/O busses are to be ignored, or to check every DMA device and every memory page (this is the default). Only the necessary changes have to be typed in when the user wants to rerun the program.

There are two versions of the program.

VERSION		DEVTST-MB
DEVTST-SBA		
LOCAL CORE	4 KW	8
KW		
COMMON MEM	4 KW	-

MULTI/SINGLE		MULTI
SINGLE		
TTY ADDRESS		FA00
FA00		

DEVTST-MB is a multi-bus version, whereas DEVTST-SBA is a single-bus version.

Note that the single-bus version of the program transfers data to the local memory. The starting addresses of the checked pages are: 2000, 4000, ... , A000.

1.8 DMACHK

Purpose

The purpose of the DMACHK program is to check one DMA device, or two connected DMA devices. The program handles either modems or HLC's. In the normal operation mode, the program tests the two devices and their connection by transferring data from memory pages to themselves through the devices.

There are two versions of the program.

VERSION	DMACHK-MB	DMACHK-SBA
LOCAL CORE	8 KW	12 KW
COMMON MEM	4 KW	----
MULTI/SINGLE	MULTI	SINGLE
TTY ADDRESS	FA00	FA00

DMACHK-MB is a multi-bus version, whereas DMACHK-SBA is a single-bus version. The program assumes that the TTY is not a hard copy device, and waits for the user to type a character when the screen is full.

Note that the single-bus version of the program transfer data to the local memory. The starting addresses of the checked pages are: 4000, 6000, ... , C000.

1.9 EXHIT

Purpose

The purpose of the EXHIT program (HIT executive) is to provide an interface between the HIT program and the user. HIT is the general Pluribus system test program. It primarily tests shared memory, I/O devices, and the bus couplers. All (or some of) the processors in the system execute the tests simultaneously but independently.

Previously, HIT ran under DDT which was used both for preparing to HIT the system configuration tables, and for interpreting its error tables. EXHIT eliminates the need for DDT in order to run HIT and instead provides a convenient way for interacting with HIT.

Environment

The EXHIT program is divided to two logical parts. The first part is the conversation part in which the user selects the parts of the system to be tested (memory busses, I/O busses, DMA devices, BLI devices), the processors to participate in the tests, as well as the tests themselves. The selection is easy since the program finds and prints the configuration of the system, and thus helps the user in the selection task.

The second part is an on-line command interpreter which allows the user to control the tests while HIT is running: e.g., to halt or restart processors, to modify some of the test parameters in any of the processors, to print readable error tables, and even to access and modify any location in the whole system address space.

1.10 HIT

Purpose

HIT is the general Pluribus system test program. It primarily tests shared memory, I/O devices, and the bus couplers. All processors execute the tests simultaneously but independently. By setting parameters, HIT may be tailored to run on many Pluribus system configurations.

Environment

HIT assumes a multibus configuration. There can be from 1 to 14 processors, each with at least 4K of private memory, of which HIT will use 4K. There may be up to two processors on each bus, but each should have its own private memory (keyed). There may be up to two memory busses and two I/O busses. Combined M/I busses can be handled; they are treated logically as a memory bus and an I/O bus. There may be up to 26 DMA-type I/O devices (Host, Modem, CBT, etc.) and 12 polling devices (e.g., BLI). Each bus that has a DMA-type device must also have a PID. RTCs may also be tested. The RLD card is not currently tested. Each memory bus may have up to 128K words of memory. It is not essential but desirable that an operator panel be located on one processor bus for lights display. Line frequency interrupts should be enabled on all processor busses. HIT does not do any initial processor starting parameter setting, or error typeout.

1.11 INVENTORY(INV)

Purpose

The purpose of the INVENTORY program is to find the configuration of a PLURIBUS system, and to print the results on the TTY.

The configuration of the whole system is found by each processor in the system, one at a time, and then concentrated tables are printed. The first activated processor is called the master, and it activates the other processors (slaves) one at a time.

The program finds the processor busses, memory busses, and I/O busses and prints what they contain. In order to be sure that a device or a memory page is plugged in the bus where it is expected to be plugged according to its address, the program instructs the operator to power off busses from time to time. In order to get maximum information, the operator has to power off all the busses when the program suggests it. In any case this is optional.

The program finds the connections between the busses. The busses connected to a processor bus are those busses that can be "seen" by a processor on the processor bus. The connections between a memory bus and I/O busses are found by trying to transfer data from the memory bus to itself using devices on the I/O busses.

The program finds the quit time on every bus in the system. The time is printed in micro-seconds and its accuracy is about 5%.

The program is a diagnostic one, but it does not perform any thorough test. It is supposed to be the first program to run on a new system (after there is at least one good processor and a TTY).

There are two versions of the program dependent on whether it is a multi-bus "MB" or single-bus "SBA" configuration.

VERSION	INV-MB	INV-SBA
LOCAL CORE	8 KW	4 KW
COMMON MEM	8 KW	----
MULTI/SINGLE	MULTI	SINGLE
TTY ADDRESS	FA00	FA00

The program consists of two logical parts: the search program, and the printing program. The search program finds the configuration and builds

tables for use by the printing program. The tables are stored in common memory (multi-bus systems) or in local memory (single bus systems). The printing program prints the tables found by the search program on the TTY.

The Different Versions

INV-MB: This version of INVENTORY occupies 8 KW of local memory. The program can run repeatedly without reloading. Since the program destroys common memory, no DDT can be used in this case.

INV-SBA: This version of INVENTORY occupies 4 KW of local memory. The program can run repeatedly without reloading. The program uses locations in the first ^h100 bytes.

1.12 KGBTST

Purpose

KGBTST is the basic card test for the Key-Generator-Black (KGB) card. It also includes several small subprograms which are useful for testing specific kinds of failures. The final section of this document contains information about checkout of new, previously untested cards. It is important to realize that the KGB and KGR are components in a subsystem, and KGBTST does NOT provide a complete test of a KGB. PLIOPT is available to test the optical-isolator Red-to-Black path logic, and PLITST is available to be run with the KG simmulator (or a real KG) to provide a complete data-path check. The operational PLI programs provide the ultimate test, and EACH of the programs must be run on every card before it can be given a clean bill of health. The last page of this document is a checkout sheet to guide you and provide a record of tests run.

Environment

KGBTST requires a single processor bus with an unkeyed or Key-3 0-8K memory. A PID is not required except to F-stick the bus. If no F-Stuck PID is available, address bits 16-19 may be easily forced to decode as specified on KGB-00. The TTY interface (PSB) address should be FC10 (hex). The Operator Panel is expected at FF80, and Line Frequency, Power Fail, and Power Restore interrupts should be enabled with the appropriate panel switches. The KGB address may be set to any unused address.

1.13 KGRTST

Purpose

KGRTST is the basic card test for the Key-Generator-Red (KGR) card. It also includes several small subprograms which are useful for testing specific kinds of failures. The final section of this document contains information about checkout of new, previously untested cards. It is important to realize that the KGR and KGB are components in a subsystem, and KGRTST does NOT provide a complete test of a KGR. PLIOPT is available to test the optical-isolator Black-to-Red path logic, and PLITST is available to be run with the KG simmulator (or a real KG) to provide a complete data-path check. The operational PLI programs provide the ultimate test, and EACH of these programs must be run on every card before it can be given a clean bill of health. The last page of this document is a checkout sheet to guide you and provide a record of tests run.

Environment

KGRTST requires a single processor bus with an unkeyed or Key-3 0-8K memory. A PID is not required except to F-stick the bus. If no F-Stuck PID is available, address bits 16-19 may be easily forced to decode as specified on KGR-00. The Operator Panel is expected at FF80 (hex), and Line Frequency, Power Fail, and Power Restore interrupts should be enabled with the appropriate panel switches. The KGR address may be set to any unused address.

1.14 MEMCHK

Purpose

The purpose of the MEMCHK program is to find all processors and memory pages in a PLURIBUS system, to check each page by all processors, and to print test results on the TTY.

In the most general case, the program finds all processors, all processor memories, all common memory pages, and all I/O busses. Then the program activates the processors, one at a time, each of them tests the whole memory.

The first activated processor is called the master, it activates the other processors which are called slave processors. The master's buddy is called the buddy, and the remote processors are called remote processors.

Each processor starts with the common memory pages. Each page is tested through all map registers (0, 1, 2, 3). Then the processor tests processor bus memories. Memories of remote processors are tested through all I/O busses via BBC's. The processor tests all its pages, but doesn't test the memory of its buddy.

Each page is tested by a sequence of up to 11 tests, and the process continues indefinitely. Each test consists of 2 parts. First the whole page is written with unique information, determined by the current test, then the information is read and compared to the expected information. All found errors are printed on the TTY. Errors are also accumulated in the TEST LOG which is printed when the user requests. The TEST LOG contains counters of several kinds, a list of bad common memory pages, a list of complaining processors, etc. It is described in a later section.

The program interacts with the user and allows him to select a subset of the processors to run the tests, a subset of the processor memories to be tested, which common memories are tested, what map registers are used, what I/O busses are used in the BBC, and what tests to run. During the testing, the user can see the TEST LOG, can halt the program, and can restart the program by typing special characters on the TTY.

Versions

There are two versions of the program which are derived from the same source file by assembly parameters.

VERSION		MEMCHK-MB
		MEMCHK-SBA
LOCAL CORE	8 KW	8 KW
COMMON MEM	8 KW	--
		--
MULTI/SINGLE		MULTI
		SINGLE
TTY ADDRESS		FA00
		FA00
DDT VERSION	COMN	--
		--

MEMCHK-MB is a multi-bus version, whereas MEMCHK-SBA is a single-bus version.

1.15 MEMCLR

MEMCLR is used today as a header program on diagnostic and operational tapes. It finds all of memory, erases it, and prints on the TTY what pages of memory it found in both processor and common busses.

It displays the I/O bus used for backward buscoupling and the address of the PSB used for printing in the DATA and ADDR lights of the fron panel. It will use F bus and FA00 if possible and E bus and EA00 if necessary.

1.16 MLCTST

Purpose

MLCTST is the test program for the hardware portion of the PTIP. It is strongly oriented toward testing the terminal handling hardware and does not test common memory or most of the I/O system. MLCTST attempts to test all the hardware which is used by the operational PTIP program.

This version of MLCTST is used to test the original MLC and the C30 presently being used to emulate the MLC.

Environment

MLCTST is designed to run in any PLURIBUS with 8K of local memory. MLCTST itself occupies the lower 4K and expects to find a DDT starting at 2000 HEX. The program pushdown stack starts at 3E00, but this is patchable to be elsewhere if necessary. Since all PTIP local memories are 8K, this is not usually a hardship. No common memory is used and no I/O except for the AML under test is required. The AML may be located on the local processor bus, on an

attached I/O bus or on an M-I bus.

1.17 MSPMTST

MSPM tests the MSPM (also called BSO) card using a DDT-41 as a mini-operating system. It contains the means to talk to the card's micro DDT.

1.18 PIDTST

Purpose

PIDTST is the basic card test for the Pseudo-Interrupt Device (PID) card.

Environment

PIDTST will work in either a single or multiple bus environment. If a single bus, it must be F-stuck and the memory must be keyed. A processor with 0-4K local memory is required. See the detailed operation section for default parameter settings.

1.19 PLI-KG-TST

Purpose

The purpose of the PLI-KG-TST program is to test the KG, the KGR, and the KGB in a PLI system. The system is viewed as having two paths: the transmit path (TX path) which connects the RED processor to the BLACK processor through the KG; and the receive path (RX path) which connects the BLACK processor to the RED processor through the KG. The program gathers information on the behavior of both paths.

1.20 PLI-OPTCAP-TST

Purpose

The purpose of the PLI-OPTCAP-TST program is to test the two direct paths in a PLI system: the "Black to Red" (optical isolators) path, and the "Red to Black" (capacitors) path; and to gather information on the behaviour of both paths.

1.21 PSTOPL

PSTOPL is a small program that halts the processors in a multibus Pluribus. It is used before running other diagnostics to be sure that rampant processors don't intefere with the diagnostic to be run. It is usually combined with the memory clearing program MEMCLR and called PSTOPZ. PSTOP and MEMCLR are routinely added to the front of diagnostic (and operational) tapes to ensure a quiescent machine before the main program is loaded.

1.22 PWRTST

Purpose

PWRTST tests the processor bus power fail and restart interrupts, including the auto-reset timer facility. A rudimentary check of 60 Hz interrupts is also done.

Environment

The bus under test must have a processor connected to the BCU (to receive interrupts), at least 4K words of memory, and a control panel. The bus may or may not be part of a multibus environment; no assumptions are made about this. Line

frequency, power fail, and power restore interrupts should be enabled per the switches on the operator panel, if any.

1.23 RLDCHK

Purpose

The purpose of the RLDCHK program is to find and test the RLD devices in a PLURIBUS system, and to print detailed information on the TTY.

In the most general case, the program finds all modem devices, with connected RLD, and for each device it tries to transfer data from every common memory page to itself using the device. The size of the transferred block is F00 words

For every transfer unique RLD packet information is stored in the transmit buffer (the lower F00 words of the page),the destination buffer (the upper F00 words of the page) is cleared, and the device is activated.

A program timer is set for each activation of the device. The transfer is terminated in one of two ways:

1. The two bits in the device register block- Transmit active and Receive active- are both reset before the program timer reaches the end of its period, or
2. The timer times out with one of the active bits still set.

In any case, after transfers are terminated, the program prints the results of the transfers.

The results of the transfer are printed in a table and include information about the status of different error bits in the device registers block, the success of the transfer, the success of writing to the PID, etc. The program interacts with the user and allows him to select specific I/O devices, to select specific memory busses, to specify which memory

pages are to be ignored, to specify which I/O busses are to be ignored, or to check every MODEM/RLD device and every memory page (this is the default). Only the necessary changes have to be typed in when the user wants to rerun the program.

RLDTST will test the RLD (Magic Modem) to see that it can actually force a reload into a Pluribus.

1.24 RTCTST

Purpose

RTCTST is the basic card test for the Real Time Clock (RTC) module.

Environment

RTCTST will work in either a single bus or multiple bus environment. If a single bus, it must be F-stuck and the memory must be keyed. A processor with 0-4K local memory and a PID are required.
Line frequency, power fail and power restore interrupts should be enabled on the processor bus. See the detailed operation section for default parameters.

1.25 TTYTST

Purpose

TTYTST tests a Teletype like device (e.g., a VISTAR) and its associated serial interface (PSB). The classical interrupt capabilities of the PSB are not tested.

Environment

The PSB under test may either be on the same bus as the processor or on a separate I/O or MI bus. The processor must have 0-4K local memory and an operator panel should be located on the processor bus. See the detailed operation section for default parameters.

1.26 Hexadecimal Numbering

Throughout the Domestic Data Network the hexadecimal numbering system is the most common one in use, because PLURIBUS is the standard machine. For this reason, knowledge of it is essential.

These are the hexadecimal digits with their binary, octal, and decimal equivalents:

Hex.	Decimal	Octal	Binary
0	0	0	0000
1	1	1	0001
2	2	2	0010
3	3	3	0011
4	4	4	0100
5	5	5	0101
6	6	6	0110
7	7	7	0111
8	8	10	1000
9	9	11	1001
A	10	12	1010
B	11	13	1011
C	12	14	1100
D	13	15	1101
E	14	16	1110
F	15	17	1111

Just as in octal, the binary bits are grouped to form the digits of the hexadecimal number; the difference is that they are grouped by fours instead of in triplets:

Hex		Binary (fours)		Binary (threes)		Octal
53B4	=	0101 0011 1011 0100	=	0 101 001 110 110 100	=	051664

The bits in a PLURIBUS word are numbered differently than in Honeywell machines. It is important to understand this as site people will be reporting register contents using bit numbers, and you must be careful not to be confused. This is the format of the PLURIBUS bit numbering scheme:

15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0

Sometimes one refers to a "byte" or half word. In PLURIBUS using the hexadecimal system this is easy: one byte is exactly two hex digits. For example, a byte of all ones is FF; as another example, if a word contains the (hex) number 10A3, one would say the left byte is 10 and the right byte contains A3.

The PLURIBUS is a byte addressed machine, so when a word is being referred to its address is always even (ending in 0, 2, 4, 6, 8, A, C, or E). Bytes (half words) may also be addressed: the even (word) address always refers in this case to the left byte; the word address plus one then refers to the right byte. Hence they are often called the even and odd bytes of a word.

Finally, for reference purposes, here is a table of the correspondence between octal bit values and hexadecimal bit values, with the Honeywell and PLURIBUS bit nu berings corresponding.

PLURIBUS Bit no.	Hex Value	Octal Value	Honeywell Bit no.
0	0001	000001	16
1	0002	000002	15
2	0004	000004	14
3	0008	000010	13
4	0010	000020	12
5	0020	000040	11
6	0040	000100	10
7	0080	000200	9
8	0100	000400	8
9	0200	001000	7
10	0400	002000	6
11	0800	004000	5
12	1000	010000	4
13	2000	020000	3
14	4000	040000	2
15	8000	100000	1

1.27 The Pluribus Control Panel

The control panel can be used to access the processor(s) on the bus to which it is attached and to access associated local memory. It can also be used to access common memory and I/O devices. The following description assumes that the panel is enabled.

The switches on the face of the PLURIBUS control panel consist of small buttons beneath the markings on the panel. A gentle push there should result in a small click and the changing of a light above the switch. Use only fingers to push the buttons; do not use pencils or other sharp objects.

RESET: Causes system reset of the processor bus. Also lights LOAD and IDLE indicators.

LOAD: When the auto-load is armed, indicated by the LOAD indicator being lit, pressing LOAD will cause the auto-load device to begin loading (this is usually the paper tape reader). Any activity on this bus will turn off the LOAD light.

INHIBIT: Pushing this switch toggles the INHIBIT light. When it is on, all system interrupts on the bus are inhibited.

REGISTER_switches: These select for examination or change any of the sixteen processor registers on a processor. Only one of these can be selected at a time. Register 0 is the program counter, registers 1-7 are the general registers R1-R7, and register 8 is the status register. The rest are not generally used or referred to.

ADDRESS_switches: These sixteen toggle switches are used to represent addresses using the binary (and hexadecimal) numbering system described in section 1.1.

CLEAR_(address): Next to the ADDRESS switches, this will clear all the ADDRESS lights.

DATA_switches: These sixteen toggle switches are used to represent a sixteen bit binary (or hexadecimal) data word.

CLEAR_(data): Next to the DATA switches, this will clear all the DATA lights.

READ_(register): Next to the REGISTER switches, this causes the contents of the register selected in the REGISTER lights to appear in the DATA lights. (Applies only to selected processor. See below.)

WRITE_(register): Next to the REGISTER switches, this causes the contents of the DATA switches to be written to the register selected in the REGISTER lights. (Applies only to selected processor.)

READ_(address): Next to the ADDRESS switches, this causes the contents of the memory word or I/O device register addressed in the ADDRESS lights to appear in the DATA lights. If the address is below 4000(hex) the memory word is in the selected processor's local memory. If the address is greater than C000, it is an I/O device register or a processor hardware register. Otherwise the address references a word on some page of common memory (see section 1.3.3).

WRITE_(address): Next to the ADDRESS switches, this causes the contents of the DATA switches to be written to the memory or device register addressed by the ADDRESS lights. See the above paragraph for more on addresses.

BUSY: This light is on if there was any activity on the processor bus within the last 0.1 second.

IDLE: This light is on when no processor on the bus is running and one is waiting for an interrupt to proceed.

DONE: When any register, memory, or processor action performed from the panel is completed successfully, the DONE light is lit. If such an operation is not completed, the DONE light stays off. The controls which can turn DONE on are: both WRITEs, both READs, RUN, HALT, ADH, and STEP.

RUN: The light will be on when any processor on the bus is executing instructions (running). Pushing RUN will cause the selected processor to begin (or continue) executing instructions.

HALT: This light is on when all processors on the bus are stopped, either because they executed an instruction that halted them, or because the HALT button was used. Pushing the HALT switch halts only the selected processor.

ADH: This is the Address-Halt toggle switch. When this is on and any operation on the bus uses the address shown in the ADDRESS lights, the selected processor will be halted. This is usually used only as a debugging aid.

STEP: This switch causes the selected processor to execute one instruction (halting the processor if it was running). The contents of the register selected in the REGISTER lights are shown in the DATA lights automatically.

ATTN: Pushing and releasing this button generates a system interrupt on the processor bus. If a processor responds to it, the ATTN light will stay off. Generally, if a processor is running, or if IDLE is on because you have just pushed RESET, the system will respond. If not, the ATTN light will stay on and the panel will be lo bed. This may be cleared by pushing the ATTN button a second time.

The next set of switches to be described is located behind the control panel at the bottom. They can be felt and changed from the front. However most of them are labeled on the back (the panel can be hinged at either end). Since these switches can be changed without swinging out the panel they will be discussed viewing the panel from the front.

PROCESSOR_SELECT: This UNLABELED switch is the rightmost one In its left position it selects the even processor on the bus; in its right position it selects the odd processor (not present on all processor buses). Check your configuration!!!

PWR_FAIL: The second switch from the right determines whether a power failure causes an interrupt to the processor. If it is pushed to the right it is ON and should be ON.

PWR_RCVRY: This switch is the third from the right and controls action when power recovery is detected. There are three positions: OFF(to the lft.), meaning no action; IN(straight down), meaning a power recovery interrupt will occur; and AL(to the right), meaning an auto-load will occur. This switch should be on IN.

LINE_FREQ: This switch is the leftmost switch and allows or inhibits the 60 hertz line frequency interrupt from the power supply (also called jiffy interrupt). It should be ON(to the right).

CONT: This hardware debugging aid should always be OFF.

1.28 Inspecting Or Changing Memory Or Registers

Processor Registers

To examine or change the contents of a processor register from the control panel it is necessary to have selected the correct processor, if there are two on the bus.

To examine a register:
1. Select the desired register using the REGISTER switches.
2. Push READ (register).
3. The contents of the selected register will appear in the DATA lights. If the selected processor does not exist on the bus, the DONE light will remain dark. If the processor is IDLE (has been RESET) it must be first be put in HALT.

To change the contents of a register:
1. Select the desired register using the REGISTER switches.
2. Enter the new contents desired for the register into the DATA switches.
3. Press WRITE (register). If the operation is successful, the DONE light will now be on. You may verify the correctness of the write by reading the contents of the register.

Local Memory And I/O Device Registers

The following applies to local memory belonging to the selected processor and to registers in I/O devices: addresses below hex 4000 or above C000.

To read the contents of a location:

1. Enter the address of the location into the ADDRESS switches.
2. Push READ (address).
3. The contents of the location will appear in the DATA lights and DONE will come on. If the memory location or the device does not exist, then DONE will remain off.
4.

Each successive push of READ (address) will increment the ADDRESS shown by 2 (i.e., one word) and read the next word into the DATA lights. To reread the same location without the auto-increment, toggle one of the ADDRESS bits twice.

To write data into a location:

1. Enter the address of the location

into the ADDRESS switches.

2. Enter the new contents intended for the location into the DATA switches.

3. Push WRITE (address). If the operation was successful the DONE light will come on. You may verify the correctness of the operation by pressing READ (address) to check the contents of the location. Pushing READ (address) a second time will advance the ADDRESS lights to the next word.

Common Memory

Assuming the location and the physical page address in common memory desired are given, one must apply the above operation an extra time to set a memory map register to the required page. The map register depends on which common memory window is referenced:

Address range	Map register
4000 - 5FFE	FC00
6000 - 7FFE	FC02
8000 - 9FFE	FC04
A000 - BFFE	FC06

It is highly inadvisable to use the fourth window (address A000 through BFFE), as it is a special purpose window which zeroes a word as it reads the word.

To read or modify common memory locations on a given page in common memory: first, using the procedure given above, write the physical address of the page into the appropriate map register. Then access the desired memory locations as descr bed above.

1.29 Hints To Pluribus Diagnostics

1. Before loading any diagnostic, load PSTOPL to ensure quiescent state of the machine.

2. Any program can be restarted, after it has been initially loaded, by pressing "reset", then "atten" buttons on the control panel. WARNING: The locations and/or tables that were modified during that run have NOT been changed.

2 DDT

Many diagnostics require the use of DDT to set parameters such as lines to be tested, baud rate, and device to be tested. A few DDT commands are presented below and are a subset that are needed to effectively run the diagnostics. DDT provides a means to control/monitor control/monitor processes (programs) by inspecting/changing registers of the machine.

The following conventions will be used to designate keyboard entry:

used to enclose what has to be typed in

^ denotes control key held down while striking next key in command

Pxx processor xx executed read/write

yy value read out of location

Whenever a register is "opened", its con tents are typed out in the current mode (except as noted for certain commands). When a register is "closed", the last value typed in while open, if any, is written to that register. If nothing or <delete> is typed in, nothing is written.

2.1 Addresses_Opening And Closing Commands

nn/ Opens register nn. The processor in whose address space the reference was made types out as "Pnn" immediately following the /. The contents of the requested register then are typed in the current type out mode.

<cr> Closes current register, if any open.

<lf> Closes current register, if any open, and opens next "instruction"; that is, if type out mode is symbolic and the current register is a double-word instruction, skip one register.

. By itself, is the value of the address of the current register, if any open; if none, then the last current register.

/ Types out the contents of the register addressed by the current register but does not open it or change ".". The address used is the "effective" address of the symbolic instruction, if the current type-out was symbolic. For instructions that have no effective address (HLT for example), or for type out in constant or ASCII mode, the actual memory contents are used for the "effective" address.

2.2 Type Out Mode Commands

There are two orthogonal type out modes. One controls the radix of type out.

^H Numbers are typed out in hexadecimal (base 16)- default

^O Numbers are typed out in octal (base 8).

The other controls how register contents are interpreted:

^S Type out symbolically, that is, try to interpret as an instruction, including next word if a two-word instruction code.

^K Constant) type out as a number.

^A Type out as two ASCII characters.

2.3 Type In Commands

nn. a decimal number

nn' an octal number

nn! a hexadecimal number

2.4 Control Commands

nn^G starts the selected processor at nn. If the selected processor is running STAGE, nn is copied to its simulated R0 and its R15 is set to 2.

^G starts the selected processor at the address last specified by a ^G command. If no argument is given and no address has been specified previously, a "#" prints and nothing else happens.

^X Stop the selected processor is running and types out the contents of the program counter. If not running, types out "HALTED". For processors running in the STAGE system, sets their simulated R15 to a 1, and prints the contents of simulated R0.

^P Causes the selected processor to proceed from its current state. If a processor running STAGE is selected, its R15 is set to 2.

^Z Steps the selected processor one instruction and types that instruction. Processors running STAGE set their R15 to 3. In this case, single-stepping is meaningless and should not be tried.

^L Clears the screen and repaints the display in the bottom portion of the screen.

^O Complements value of the override switch and echoes "ON" or "OF" as appropriate.

3 DEVTST

3.1 Loading And Starting The Program

Before loading the program all busses should have power turned on. The bus reset timers of all busses should be in the OFF (up) position.

In this version of DEVTST(REV.3) PSTOPL has been included in the compulation of DEVTST. Therefore, it is not necessary to run PSTOP prior to loading DEVTST. RESET-ATTENTION can be used to restart the program.

A. Load correct DEVTST tape or cassette according to system configuration.

B. Press "Reset" on control panel.

C. Set start address in data lights on control panel if loading from cassette.

D. Press "Load" on control panel. Check the console to see if all processors and common memory are listed as per the SRN. If not, you are not running the correct version. If no display on console, check to see if your console is at the address the diagnostic expects to see it.

E. Verify next on the screen is start of questions.

3.2 The Interaction With The User

The program starts by interacting with the user, which has to define the program's mode of operation. The answers to the questions are:

Y- yes, N- no, or E- exit (to the end of the program).

Each of the above characters must be followed by a carriage return. If an illegal pair of characters is typed by the user, the program prints the string "? (Y/CR,N,E)" and waits for a legal response.

The program interprets a single carriage-return as the character "Y" followed by a carriage-return. The messages and questions printed by the program are described next.

1. DEVTST-MB The program name is the first message printed by the program.

2. CHANGES? (Y/CR,N,E)

This is the first question asked by the program. If the user does not want to make any change in the program's mode of operation, then the character N should be typed. If this is the first iteration of the program, the answer N denotes the default mode, which is described later. If there are no changes, the program starts activating the devices according to the current mode of operation. The format of the printed tables is described in the next section. If there are changes, the conversation continues in step 3.

3. HARD COPY? (Y/CR,N,E)

If the user replies negatively, implying a CRT device, then he should type any character to cause the printing of each table. If the typed character is E (exit), message 16 is printed. If the user replies positively, all tables are printed without pauses between tables. RESET-ATTENTION can be used to abort printing in this case and to restart the program. If the operator replies with E, then message 16 is printed.

4. I/O DEVICES, CHANGES? (Y/CR,N,E)

The user has to reply whether there is a change in the list of I/O devices to be tested. If the answer is Y, then message 5 is printed next. If the answer is N message 10 is printed. The default list of I/O devices contains all I/O devices.

5. XPATCHED? (Y/CR,N,E)

If the I/O devices should be tested, internally cross-patched, the answer Y should be given (this is the default). If the answer is N, then the devices are tested, externally looped. A looping plug can be used in this case to enable the test.

6. TEST ALL I/O DEVICES? (Y/CR,N,E)

If all devices are to be tested, then the answer should be Y (this is the default for the first iteration). If the answer is N, then the current table of I/O devices to be tested is cleared, and the user has to type up to 5 device addresses (e.g. E150, F220) as directed by the program in the iterated sequence of steps: 7-9.

7. MORE? (Y/CR,N,E)

The user has to answer if more entries are to be added to the currently discussed table. If the answer is Y, the program prints message 8. If the answer is N, then the table is complete and the conversation continues. This message is given even at the start of building the table. It should be noted that the current table is cleared before the sequence of steps 7-9 is started, and therefore the user has to type the complete table (and not only the modifications).

8. ADDRESS=

Following this message the user has to type a 4 digit hexadecimal address (including leading 0's) as the next entry to the currently open table. After 4 characters have been typed by the user, message 9 is printed by the program.

9. CONFIRM? (Y/CR,N,E)

The user is required to confirm an address typed as an answer to the previous message. If the answer is N, then question 7 is repeated, and the typed address is ignored. If the answer is Y, then the address is added to the table. Then, if there is still room in the table, question 7 is printed, else the sequence of steps: 7-9 is terminated and the conversation continues.

10. MEM BUSSES, CHANGES? (Y/CR,N,E)

The user has to reply whether there is a change in the table of memory busses included in the test. If the answer is Y, then message 11 is printed. If the answer is N, message 12 is printed and the table is not changed.

11. TEST ALL MEMORY BUSSES? (Y/CR,N,E)

If all memory busses should take part in the test, then the answer should be Y (this is the default for the first iteration). If the answer is N, then the current table of memory busses to be checked is cleared, and the user has to type map addresses of up to 3 pages (e.g. 0000, 0200) which are interpreted as starting addresses of the memory busses to be checked. A memory bus is a contigous sequence of (_^h2000 byte) memory pages. The addresses are typed by the user as directed by the program in the iterated sequence of steps: 7-9. After this step is completed, question 12 is printed.

12. IGNORE-I/O BUSSES, CHANGES? (Y/CR,N,E)

The user has to reply whether there is a change in the list of I/O busses to be ignored in the test. The list is initially empty, i.e. no I/O bus is initially ignored. The I/O busses address ranges are: E000-E7F0, E800-EFF0, F000-F7F0, and F800-FBF0. The first _^h80 bytes are the BCM area, and no devices are searched there. If the answer is Y, then the table of I/O busses to be ignored is cleared, and the user has to fill it by typing up to 3 I/O busses starting addresses (e.g. E000, E800). The addresses are typed by the user as directed by the program in the iterated sequence of steps: 7-9. If the answer is N, then the table is not changed. After this step is completed, question 13 is printed.

13. IGNORE-PAGES, CHANGES? (Y/CR,N,E)

The user has to reply whether there is a change in the table of memory pages to be ignored during the test. The table is initially empty, i.e. no memory pages are initially ignored. If the answer is Y, then the table of memory pages to be ignored is cleared, and the user has to fill it by typing up to 8 memory busses starting addresses (e.g. 0000, 0200). The addresses are typed by the user as directed by the program in the iterated sequence of steps: 7-9. If the answer is N, then the table is not changed. After this step is completed, question 14 is printed.

14. SLOW DEVICES? (Y/CR,N,E)

The user has to reply whether there are slow devices in the system, for which the transfer can last more than 1 second. A 50 Kb modem should complete the transfer of the 2000 words in approximtely 2/3 second. The answer to this question determines the length of the period to which the program timer is set. The default for the first iteration is 1 second, i.e. no slow devices. If unexpected results are obtained, it would be a good practice to rerun the program with the longer timer set. The long timer is set to about 2.5 seconds. If longer timouts are required the timer loop can be increased (see P130-P140 in the listing).

15. TEST GOOD PAGES ONLY ONCE? (Y/CR,N,E)

If a positive reply is given, then each transfer (sucessful or unsuccessful) is performed once; otherwise, the user has to specify a number in the next step. The default is to test each good page only once.

16. TIMES=

Following this message, the user has to type a non-zero four digit hexadecimal number specifying how many times each sucessful transfer should be repeated. If the user asks for n times, each transfer between a common memory page and a device will be performed n times if all of them are successful, or until the first error occurs.

17. DMA TRANSFERS

This title is printed before the actual transfers starts. It is followed by the tables, as described in the next section.

18. RESTART? (Y/CR,N,E)

The question is printed when all selected devices have been checked. Y causes restarting of the program with the current mode of operation which may be changed by the user, N causes reprinting of question 18, E causes a transfer of control to DDT (which should be loaded in this case, since the program does not check whether DDT is in core).

19. UNEXPECTED QUIT AT: XXXX

The message is printed for the obvious reason. The program should be restarted and if the message is consistent then the program listing must be used to analyze the cause of the quit.

PRINTING THE DMA TRANSFERS RESULTS

After the program's mode of operation is set as described in the previous section, the devices are activated one at a time. For each device the transfer is done sequentially for each selected memory page. The transfers' results are printed in tables, one per device.

The format of each table is as follows:

```
DEV ADDR: XXXX   TYPE: YY   NAME: NNN   TPID: TT   RPID: RR   NUM: ZZ
-------------------------------------------------------------------------

MAP TCOM RCOM TPID RPID DATA NELB MQIT DQIT TQIT RQIT TERR RERR TSRS CNT
V
VVV  F    F    F    F    F    F    F    F    F    F    F    F   ttrr cccc
  .
  .
  .
WWWW F    F    F    F    F    F    F    F    F    F    F    F   ttrr cccc
```

The header of each table consists of the device address, the device type, a three character nmemonic name, the transmit PID level, the receive PID level, and the device number (byte 2 in the device registers block).

Each row summarizes the transfer's results for one memory page, whose map address is denoted by VVVV or WWWW. A blank line is printed between memory busses.

The entry in most of the table fields is a single digit denoted by F. F can be one of the characters: "1", "0", or "?".

1 means that the event associated with the corresponding field has happened during the transfer.

0 means that the event has not happened.

? means that the program does not have the necessary imformation (e.g. because of a quit).

The meaning of each event is as follows:

TCOM- The transmit part of the device had terminated the transfer before the end of the program timer period was reached (i.e. bit 13 in word 6 of the device registers block was 0).

RCOM- The receive part of the device had terminated the transfer before the end of the program timer period was reached (i.e. bit 13 of word 3 of the device registers block was 0).

TPID- The transmit PID level was written into the correct PID.

RPID- The receive PID level was written into the correct PID.

DATA- The expected data was found in the receive buffer of the checked memory page. For an HLC device the character "P" can follow the digit denoted by F, this means that the padding word (_^h8000) was found at the end of the receive buffer.

NELB- The bit indicating no error and last buffer was on (i.e. bit 15 in word 2 of the device registers block was 1).

MQIT- The processor got a quit from the checked memory page.

DQIT- The processor got a quit from one of the device registers.

TQIT- The device transmit quit bit was set (i.e. bit 8 in word 6 of the device registers block was 1).

RQIT- The device receive quit bit was set (i.e. bit 8 in word 3 of the device registers block was 1).

TERR- The device transmit error bit was set (i.e. bit 0 in word 5 of the device registers block was 1).

RERR- The device receive error bit was set (i.e. bit 0 in word 2 of the device registers block was 1).

TSRS- The last field in the table contains the hexadecimal contents of two status bytes in the device registers block: the transmit status (i.e. the contents of byte c in the device registers block), and the receive status (i.e. the contents of byte 6 in the device registers block).

CNT- This field is printed only when a not hard copy terminal is used. It specifies the value of the counter which counts the transfers to the page described in the printed line.

3.3 Interpretation Of The Results

A successful transfer for a common memory page is reflected in the printed table by a line whose first 6 fields (TCOM-NELB) are 1, and the next 6 fields (MQIT-RERR) are 0. The contents of the last 2 fields (TSTT-RSTT) is device dependant.

A successful transfer for a common memory bus is reflected in the printed table by a group of identical consecutive lines, each representing a successful transfer.

Note that the program tries to transfer data for every memory bus (unless the user has selected another mode of operation), and therefore if a device is not connected to some memory bus (as in the case of combined M/I busses), then it will be reflected in the printed table as an unsuccessful transfer.

4 DMACHK

For our discussion, one of the two devices under test is called the TST (test) device; the other (if exists) is called the REF (reference) device. The assumption is that the REF device is a good, working, device which therefore can help the user in isolating problems in the TST device; although, if the REF device causes errors they are reported and the program functions correctly.

Each of the two devices consists of two parts: the TX (transmit) part, and the RX (receive) part. If the devices are connected there are two paths; each path containing a TX part of a device, a RX part of a device, and a cable connecting them. The program can handle either one path or two paths. Each path has a TX device and a RX device associated with it. If only one device is tested, there is only one path.

The program tests each path by transferring data from a memory page to itself using the TX device and the RX device of that path; this test is called the DMA test. The program gathers information on the behavior of both paths during the DMA test in an error log which can be examined by the user.

The user can specify to the program various parameters which define the DMA test to be performed. Those parameters are defined during the conversation with the program before the tests begin. The user can select the DMA devices, the memory pages used in the test, the data transfer parameters, and the configuration of the devices connection. Several configurations are possible. If one device is tested, it can be crosspatched, looped (modems only), or externally looped. If two devices are tested, there are two possible paths to be tested. The user can specify that only one of these paths is to be tested (simplex); both paths are tested by concurrent transfers on the two paths (full duplex); or both paths are tested by alternately transferring data on the two paths, one path at a time (half duplex).

In each iteration, the program clears the buffers used in the previous iteration (by writing 0 in each location), selects memory buffers for the current transfer, loads the TX buffers with unique information, clears the RX buffers (by writing the word DEAD in each location), and loads the devices registers. Then the program waits until the relevant device parts have completed (by examining their active bits) or a timeout occurred. After this waiting period, the program checks the various device bits, the data in the RX buffer, and the PID's. If any error is detected, a new entry is pushed into the error log, and a new iteration begins.

In case of modems, the contents of the TX buffer is transferred to one RX buffer. In case of HLC's, another mode is possible (the default mode) in which the first two words are transferred to RX buffer 1, and the rest of the TX buffer is transferred to RX buffer 2.

During the test itself, the user can communicate with the program by typing on-line commands on the TTY. He can see the current summary of the test, examine (and modify) memory locations, force errors, cause the program to enter or exit a loop, restart the test, or begin the conversation again.

Before the DMA test begins, the program can perform pre-transfer tests. In these tests the program measures the watchdog timers of both devices, and allows the user to modify the setting of various switches of the TST device (PID levels, device number, and device address) and displays the switches setting on the console lights. Additional test is the registers test. In this test, the program writes various patterns to all device registers and checks that the appropriate bits are changed, and that unexpected changes do not occur. If an error is detected, the program can enter a loop (if the user specified it during the conversation) thus allowing the user to use a scope for examining the device behavior. Information on the errors is printed as they are detected; the error log does not contain any information on the registers test.

4.1 Loading And Starting The Program

Before loading the program each processor bus should have power turned on. Power restart interrupts should be enabled (S71 in the console in the middle position which enables the power recovery int). Line frequency interrupts (JIFFY) should be enabled (S70 in the console in the ON position). The bus reset timer of each bus should be in the OFF (up) position.

First PSTOP must be used to halt all processors. Then DDT may be loaded if the operator wants to use it later (the DMACHK program does not need DDT for its operation).

A. Load correct DMATST tape or cassette according to system configuration.

B. Press "Reset" on control panel.

C. Set start address in data lights on control panel if loading from cassette.

D. Press "Load" on control panel. Check the console to see if all processors and common memory are listed as per the SRN. If not, you are not running the correct version. If no display on console, check to see if your console is at the address the diagnostic expects to see it.

E. Verify next on the screen is start of questions.

Then the program is loaded and starts automatically. The conversation can always be restarted by one of the following ways: using RESET-ATTENTION, restarting the program at ^H100, replying "E" to one of the program questions during the conversation, or via the on-line command "B" (begin conversation) during the test itself. Causing a power restart interrupt to the processor (by turning its power OFF and ON) before the DMA test begins can be used to restart the conversation (although using "E" is more appropriate).

Causing a power restart interrupt to the processor (by turning its power OFF and ON) after the DMA test has begun can be used to restart the test. A similar effect can also be achieved by the on-line command "R".

For all the versions of the program, if the TTY address must be changed it can be done after the program is loaded. Location TTYADD (currently ^H104) contains the TTY address.

4.2 The Initial Interaction With The User

The program starts by interacting with the user, which has to define the program's mode of operation.

The answers to the questions are:

Y- yes, N- no, or E- exit.

"E" means exit and restart the program at the beginning of the conversation. Each of the above characters must be followed by a carriage-return. If an illegal pair of characters is typed by the user, the program prints the string "? (Y/CR,N,E)" and waits for a legal response. The program interprets a ^&single carriage-return as the character "Y" followed by a carriage-return. The messages and questions printed by the program are described next.

1. DMACHK-MB

The program name is the first message printed by the program.

2. SUMMARY? (Y/CR,N,E)

This question is asked by the program only if the conversation is restarted after the test has begun. If the user replies positively, the current status and the error log are printed (see a later description). Typing the character "X" or the character _^0 (control 0) during the summary printing aborts the printing.

3. CLEAR LOG? (Y/CR,N,E)

The error log from previous tests is cleared when a positive reply is given. The various counters are also cleared in this case. A negative reply causes keeping of the error log and the counters from previous tests; in this case, the conversation proceeds in step 5. The error log and the various counters can be cleared during the test via the on-line command "C".

4. STATISTICS TOO? (Y/CR,N,E)

A positive reply causes clearing some statistics which the program maintains about data errors.

5. TYPE COMMANDS (A- ABORT):

At this point the user can type most of the on-line commands (which are described later); e.g., he can examine the summary (if it was not cleared earlier) by typing "S", or open memory locations by means of the "O" (open) command. The character "A" must be type in order to proceed in the conversation.

6. DEVICE CHANGES? (Y/CR,N,E)

If the user does not want to make any change in the device address(es), then the character "N" should be typed. In this case, the program prints the details of the previously specified device(s) and continues in step 14.

7. NEED DEVICE LIST? (Y/CR,N,E)

If a positive reply is given, the program prints information on all I/O devices it finds; thus helping the user in determining the desired device address(es).

8. TST DEVICE ADDRESS=

Here, the user is required to type a 4 digit hexadecimal address of the TST device (e.g., E140). Typing an escape character will cause the program to use the previously selected value.

9. REFERENCE (REF) DEVICE? (Y/CR,N,E)

The user is asked whether a REF device exists.

10. REF DEVICE ADDRESS=

Here, the user is required to type a 4 digit hexadecimal address of the REF device (e.g., E160). Typing an escape character will cause the program to use the previously selected value.

11. IDENTICAL DEVICES

This error message indicates that the same device was specified both as a TST and as a REF device. The program then continues in step 7.

12. CHOOSE OTHER DEVICES

This message is printed when the program is not satisfied with the selected device(s). This can happen if a device which is not a modem or an HLC is specified, if the two devices are not compatible (e.g., one modem and one HLC), or if the program gets a quit from the device while trying to determine if the devices are acceptable. The program then continues in step 7.

13. TOO MANY DMA DEVICES

After the devices are selected, the program builds a table of all I/O devices. This table is used during the address test (which begins after step 52). This message is printed if the table cannot contain all device addresses. Currently the table can contain at most 20 device addresses.

14. MEMORY CHANGES? (Y/CR,N,E)

The user has to reply whether there is a change in the list of memory pages to be tested. If a negative reply is given before any page has been specified, all pages are tested. After a negative reply, the program continues in step 22.

15. ALL MEMORY PAGES? (Y/CR,N,E)

The user has to specify whether he wants all memory pages to participate in the test, or only some of them. If the answer is "Y", then all memory pages except those specified in step 21, participate in the test, and the program continues in step 20. If the answer is "N", then the user has to specify pages in the following steps. The user selects pages by defining ranges of pages. The first page and the last page in each range are specified.

16. MORE? (Y/CR,N,E)

A positive reply indicates that more pages are needed; the program continues in the next step.

17. FROM PAGE

Following this message, the user has to type the page address of the first page in a group of pages to be used. Typing an escape character will cause the program to use the previously selected value. A page address is a four digit hexadecimal number. For a multi-bus version of DMACHK the addresses are 0000, 0200, 0400, ... For a single-bus version the addresses are 4000, 6000, ... , C000.

18. TO PAGE

Following this message, the user has to type the page address of the last page in a group of pages to be used. Typing an escape character will cause the program to use the previously selected value.

19. CONFIRM? (Y/CR,N,E)

A positive reply is needed in order to confirm the range of pages just specified in steps 17, 18. The program continues in step 16.

20. IGNORE PAGES? (Y/CR,N,E)

The user can specify that some of the pages selected so far do not participate in the test; i.e., their contents is not modified by the program. In order to do it a positive reply should be given here. The user specifies pages to be ignored by repeatedly executing step 21.

21. PAGE=

Following this message, the user has to type the address of a page to be ignored. Typing an escape character will cause the program to use the previously selected value.

22. MEMORY PAGES:

After all memory pages to be used have been specified, the program prints the addresses of the existing selected pages. Non-existing memory pages are ignored.

23. NO PAGES

This error message is printed if no existing memory page has been specified. The program then continues in step 14.

24. TRANSFER PARAMETERS CHANGES? (Y/CR,N,E)

The user has to reply whether there is a change in any of the parameters specified in the following steps. If a negative reply is given, the program continues in step 31. A negative reply given in the first time the program is run, causes default parameters to be used.

25. RANDOM LENGTH? (Y/CR,N,E)

This question deals with the length of the TX buffer. If a positive answer is given, pseudo-random values are used. In fact, the lengths used start with 1, 2, ... , 10 (words) and only afterwards pseudo-random values are selected. If a negative reply is given, the program continues in the next step. The default answer is "Y".

26. FIXED LENGTH=

Following this message, the user has to type the fixed length to be used; the number of words in the TX buffer. The length is a non-zero four digit hexadecimal number. The maximum number is 127 decimal (which should be typed as 7F in hex). Typing an escape character will cause the program to use the previously selected value.

27. RANDOM DATA? (Y/CR,N,E)

If a positive reply is given, pseudo-random data is used for each RX buffer. If a negative reply is given, the user has to specify two words which will be used for loading the TX buffer in the following way: WORD 1, WORD 2, WORD 1, WORD 2, etc. The default answer is "Y".

28. WORD 1=

Following this message the user is required to type a 4 digit hexadecimal number defining the first of the two constant data words described above. Typing an escape character will cause the program to use the previously selected value.

29. WORD 2=

Following this message the user is required to type a 4 digit hexadecimal number defining the second of the two constant data words described above. Typing an escape character will cause the program to use the previously selected value.

30. SPLIT RX BUFFER? (Y/CR,N,E)

This question is asked only when HLC's are used. A positive answer specifies that two RX buffers are used for each transfer on a path. The length of the first buffer is two words, and the length of the second is determined according to the current length of the TX buffer. The default answer is "Y".

31. CONFIGURATION CHANGES? (Y/CR,N,E)

A negative reply indicates that there is no change in the test configuration. If this is the first time the program is run, the default configuration is used. After a negative reply, the program continues in step 40.

32. DEFAULT CONFIGURATION? (Y/CR,N,E)

If only one device has been specified (the TST device), the default configuration is crosspatched. If two devices have been specified, the default is full duplex. If a positive reply is given, the default configuration is used and the program continues in step 40.

33. BOTH DEVICES CONNECTED? (Y/CR,N,E)

This question is asked only if two devices have been specified. If only one device has been specified the program proceeds in step 37. A positive reply should be given in order to test the path(s) consisting of the two devices. Appropriate cable should connect the two devices in such case. In case of modems, additional device is needed (such as a "silver box"). If a negative reply is given, the DMA test will be conducted as if there is only one device; in this case the conversation continues in step 37.

34. CHECK BOTH PATHS? (Y/CR,N,E)

This question is asked if a positive reply was given to the question in step 33. If a negative reply is given to the current question, only one path is tested; the program continues in step 36. If a positive reply is given, the program continues in the next step.

35. FULL DUPLEX? (Y/CR,N,E)

This question is asked if both paths are to be tested. A positive reply indicates full duplex mode, a negative reply means half duplex mode. The program proceeds in step 40.

36. TX BY REF DEVICE? (Y/CR,N,E)

This question is asked when only one of the two paths is to be tested. The user has to define whether the TX device on that path is the REF device or the TST device.

37. TST DEVICE XPATCHED? (Y/CR,N,E)

This question is asked when only one device is to be tested. The three possible configurations are: crosspatched, looped (modem only), or externally looped. The default answer is "Y" (crosspatched).

38. TST DEVICE LOOPED? (Y/CR,N,E)

This question is asked only in case of a modem.

39. TST DEVICE EXTERNALLY LOOPED? (Y/CR,N,E)

If a negative reply is given the program repeats from step 37 until the appropriate configuration is selected.

40. KEEP EARLIEST ERRORS (INSTEAD OF LATEST)? (Y/CR,N,E)

In case of a positive reply the program does not push more entries into the error log after it is filled with 5 entries (thus the first 5 entries can be saved). In case of a negative reply, the latest 5 entries are kept in the error log, and previous entries are forgotten. In any case, the error log can be cleared during the test via the on-line command "C", or in the conversation as described in step 3. The answer is stored in a program switch whose status can be complemented during the test via the on-line command "K".

41. LOOP IF ERRORS? (Y/CR,N,E)

The answer to this question applies both to the registers test, and to the DMA test.
A negative reply (probably the normal case) means that different parameters are used in each iteration of the test regardless of the outcomes of the previous iteration (success or failure). A positive reply means

that if an error occurs, the test is retried once with the same parameters (e.g., same page, same buffers, same length, and same data, in case of DMA test; or same pattern stored in same registers in registers test). If the retry succeeds, the program proceeds normally. If the retry fails the program enters a loop mode as selected in the following steps (quick loop or slow loop). When the program is in loop mode, it uses the same parameters constantly.

•42. QUICK LOOP? (Y/CR,N,E)

A negative reply defines slow loop. The only difference between the slow loop mode and the normal mode, is that in the first case the same parameters are used in each iteration of the program. If a negative reply is given, the program continues in step 44. A positive reply specifies a quick loop. In this case, less checks are performed in each iteration of the program (e.g., the data is checked only in the first few words of the RX buffer, and PID levels are not checked).
After the program loads all device registers it waits a fixed delay regardless of when (or whether) the devices operation has been completed.

43. DELAY=

Here the user has to type a four digit hexadecimal number which specifies how many times the program iterates through a fixed loop after the transfers on both paths (or one) have been initiated.
Each iteration lasts about 50-100 micro seconds (depending on the number of devices). Specifying a short delay can cause the program to check the transfer results before the devices have completed. Thus, several values can be tried to find the desired delay. The delay is kept in LDELAY (currently location _^H10C) and the user can modify it while the program is running by the "O" (open) command. Typing an escape character will cause the program to use the previously selected value.

44. RE-DEFINE SYNC EVENTS? (Y/CR,N,E)

The user can specify one or more events whose occurrences cause the program to access location 2 or 3 in the local memory. Location 2 is accessed when the event occurred on path 1, and location 3 is accessed when the event occurred on path 2.
By writing this address (2/3) in the address lights, a pulse is generated each time one of these events occurs. This pulse can be used to synchronize a scope, or to trigger a logic analyzer. Before writing the address in the console lights the user has to tell the program to stop updating the console lights; this is done by pressing ATTENTION (and not RESET-ATTENTION).

45. EVENT=

Here the user has to type a two digit hexadecimal number which defines an event according to the following table. Typing an escape character will cause the program to use the previously selected value.

00- RX device error bit was set.
02- TX device error bit was set.
04- RX device quit bit was set.
06- TX device quit bit was set.
08- Quit from RX device registers occurred.
0A- Quit from TX device registers occurred.
0C- Quit from PID of RX device occurred.
0E- Quit from PID of TX device occurred.
10- Quit from memory occurred.
12- no error and last buffer bit of RX device was not set.
14- Padding word was not found.
16- Data error (in memory) was detected.
18- PID level of RX device was not written.
1A- PID level of TX device was not written.
1C- RX active bit of RX device was set (not complete).
1E- TX active bit of TX device was set (not complete).
20- Before loading device registers of a path.
22- After loading TX end address (and all device reg.) of a path.
24- Before loading device registers in phase 2 of a path.
26- End of waiting period (for transfers to complete) for both paths.
28- Start of a new cycle (1/2 paths 1/2 RX phases).

46. PRE-TRANSFER TESTS? (Y/CR,N,E)

If a negative reply is given, the program begins the DMA test by proceeding in step 56.

47. <DEV> DEVICE WATCHDOG TIMER (TX REFRESHED) IS x.yz SEC'S

The value of the watchdog timer is printed as a decimal number. The timer is measured by writing something to the TX status device register. If the value is out of the allowed range (0.8-2 sec's) an error message is printed.

48. <DEV> DEVICE WATCHDOG TIMER (RX REFRESHED) IS x.yz SEC'S

This time, the timer is measured by writing something to the RX status device register. The expected value is 0; an error message is printed otherwise.

49. <DEV> DEVICE REGISTERS TEST ...

This message precedes the beginning of the registers test for the <DEV> (REF or TST) device. During the test, the counter of bad iterations is displayed in the console address lights, and the counter of good iterations is displayed in the console data lights. In order to stop updating the console lights the user has to press ATTENTION (not RESET-ATTENTION which start the conversation).

Normally the whole registers test lasts about one second. However, if there are errors the test may last longer. In order to abort the test, the on-line command "A" (abort) can be used. After the registers test is completed (or aborted) for a device, the contents of the two counters is printed on the TTY.

Note that the loop mode selected in the conversation applies also for the registers test. Also, the user can force the program to enter (or exit from) the defined loop, by giving the on-line command "L" (loop).

If an error is detected its description is printed on the TTY; unless the program is in quick loop in which case nothing is printed.
There are several types of errors:

READ-WR: Indicates that the read data does not match the written data.

CHANGES: Indicates that changes in device registers have been detected where not expected.

OFF/ON: Indicates that while the (TX or RX) reset bit was written, certain bits which were expected to be OFF/ON were found in the wrong state.

In any case, information describing what was written (WROTE) and where was it written (IN) is printed (there are cases in which two words are written). The expected data is also printed (XPCTD). This is a masked word; bits which are not checked by the program are 0. The address where the expected data is expected (IN) is also printed; this address is not necessarily equal to the write address. A word containing the changed bits (CBITS) is also printed; each bit indicated in this word (by 1) was found to be the complement of its expected value (in XPCTD).

An example of an error description is:

CHANGES: WROTE AAAA IN E132 XPCTD 5042 IN E13C CBITS 4000

In this example, the bit corresponding to 4000 was found to be 0 instead of 1 in location E13C after writing AAAA to location E132. In order to interpret the meaning of the errors, the user has to examine the definition of the various bits in the device registers. In the above example, (assuming an HLC) the crosspatch bit in the TX status word was found to be in error.

50. TST DEVICE SWITCHES TEST? (Y/CR,N,E)

if a negative reply is given, the program continues in step 52, otherwise, it continues in the next step.

51. ADDR LIGHTS: WORD 0
DATA LIGHTS: PID LEVELS (TX,RX)
TYPE A TO ABORT

During the switches test, the program displays in the console address lights the contents of word 0 of the device registers. The left half contains the device type, and the write half contains the device number; the latter is switch selectable. The console data lights contain the TX PID level in the left half, and the RX PID level in the right half; both are switch selectable. The purpose of the test is to change the state of the relevant switches and to verify that the corresponding bits in the console lights change (thus to find broken switches). Note that the program does not check the meaning of the various bits; it merely displays them and the user has to check their correctness. The device address switches should not be changed during this test; they are tested in the following address test. The switches test continues until the "A" (abort) command is given by the user.

52. TST DEVICE ADDRESS TEST? (Y/CR,N,E)

if a negative reply is given, the program continues in step 56; otherwise, it continues in the next step.

53. ADDR LIGHTS: DEVICE ADDRESS
TYPE A TO ABORT

During the address test, the user changes the state of the device address switches, and the program traces the device address and displays the current device address in the console address lights. The data lights have no meaning during this test. The program may lose the device; in this case it prints the message in step 54 and continues trying to locate the device. If the device is located after the program has lost it, the message in step 55 is printed.

Several reasons may cause the device to disappear. First, the program tries to locate the device only in the address range: HC000-HFBF0 (multi-bus case), or HC000-HFDF0 (single-bus case). Selecting an address outside this range will cause the device to disappear

from the program point of view. Second, the program traces the device by examining the above I/O space each time the device address is changed (a quit obtained by referencing the previous device address). Each time a device is found in the I/O space, its address is compared with the addresses of all I/O devices which existed during the conversation. If a device whose address is different from all those addresses is found, it is assumed to be the TST device. Thus, if the user sets the TST device address to be identical to one of the other devices in the system, the program will lose the device.

The address test ends when the user gives the on-line command "A" (abort). The latest device address is used as the TST device address. The device details are printed before and after the test.

54. DEVICE DISAPPEARED. TRY OTHER ADDRESSES

This message is printed if during the address test, the program loses the TST device. The user should change the device address switches until the device is located.

55. DEVICE LOCATED

This message is printed when the program relocates the TST device after it has disappeared. The new address is displayed normally.

56. TESTING REF DEVICE ...

This message precedes the DMA test when two devices have been chosen in the conversation. Following this message, the REF device is tested (crosspatched) by 50 (decimal) transfers. Since at this stage only one device is tested, there is only one path- path 1. Thus, only the console address lights (corresponding to path 1) are updated. After 50 transfers are completed, the DMA test itself begins in step 57. If there is only one device, the DMA test begins immediately after the pre-transfer test are completed without printing any message.

57. STARTING TEST ...

This message is printed after the REF device has been tested as described in the previous step. The various test counters are cleared at this point, since a test with different configuration begins (the configuration selected in the conversation).

4.3 The On-Line Commands

The on-line commands are recognized by the program during the tests themselves; each consists of one or more characters (underlined in this section). After the program identifies the command it prints a short description of the command. The test is halted when the on-line command is executed. The user can normally give another on-line command before the test is resumed. Thus, in order to resume the test, the user should type a character which is not interpreted as an on-line command; e.g., a space. The commands, the messages, and their meanings are described next.

ABORT

In case of the pre-transfer tests, this command causes the current test to be aborted. In case of the DMA test, this command causes the program to stop introducing intentional errors. Intentional errors are caused via the on-line command "F" (force error). After an "A" on-line command is given, the test immediately proceeds; i.e., the program does not expect another on-line command at this point.

BEGIN CONVERSATION

This command causes restarting of the conversation. The error log is kept; it can be printed during the conversation and even be kept for the continuation of the test. If the command fails, the conversation can be restarted by using RESET-ATTENTION.

CLEAR LOG

This command causes clearing of the error log (thus allowing new entries to be pushed into the error log) and the various counters (including the time counter). The program proceeds by asking the user whether to clear the statistics too (akin to the question in step 4 of the conversation) and the user should answer by "Y" or "N".

FORCE ERROR (A/Q/R) or
FORCE ERROR (A/Q/C)

This command allows the user to cause an error. The first line is printed in case of an HLC, and the second in case of a modem. The user has to type one of the characters suggested in parentheses; their meanings are defined below.

ABORT: The transfer is aborted by writing something to the (TX or RX) begin address register of one of the devices during the transfer.

QUIT: The (TX or RX) begin address of one of the devices points to non-existent memory page; the device should get a quit.

CHECKSUM: A zero checksum is sent, thus the RX device should complain about a data error.

READY: A zero is written into the IMP ready bit (instead of one). Typing an escape character will cause the program to use the previously selected error type.

After the error type is selected, the user has to specify whether the error should apply to the TX part or the RX part of the device. Then, he has to specify whether the error should apply to the TST device or to the REF device. Finally, he has to specify whether the error should be caused once or repeatedly. Each of the above specifications is given by typing one character according to the options the program lists in parentheses. In every case, typing an escape character will cause the program to use the previously selected value.

Note, that if the user forces repeated errors, the command "A" (abort) can be used to stop causing the intentional errors.

HALT

The program stops the test and continues to interpret on-line commands. To resume the test, any character which is not interpreted as an on-line command can be typed.

KEEP EARLIEST ERRORS (ON/OFF)

This command complements the current status of the program switch described in step 40 in the conversation. The new status of the switch is printed.

LOOP (ON/OFF)

This command complements the current status of the program switch indicating whether the program is in loop mode or not. The program enters a loop mode only if a loop mode was defined in the conversation (step 41). The new status of the switch is printed.

OPEN: aaaa rrrr wwww <terminator> or
OPEN: aaaa PAGE-mmmm rrrr wwww <terminator>

The open command allows the user to access and optionally modify any location in the processor address space, in common memory, or in I/O address space. The command should be carefully used in order not to destroy the program.

aaaa is the address to be opened, or the escape character which stands for the previously opened location.

mmmm is the map address (e.g., 0000, or 0200) of the common memory page one of whose locations is to be opened, or the escape character which stands for the previously selected page.

rrrr is the contents of the opened memory location.

wwww is an optional field which specifies the new contents to be written into the opened location.

<terminator> is one of the following characters:

Carriage-return which confirms the new contents (if typed), and terminates the command.

Line-feed which confirms the new contents (if typed), and automatically opens the next location.

^ which confirms the new contents (if typed), and automatically opens the previous location.

Any other character used as a terminator, aborts the open command and does not confirm the new contents of the opened location (space is convenient to use).

If quit is detected while trying to access the opened location the word QUIT is printed by the program, and the open command is aborted.

R R(estart) command causes the program to restart the tests at step 46 of the conversation. The command is recognized only during the DMA test.

S S(ummary) command causes printing of the current status, the current error log (if it is not empty), and the accumulated statistics (if it is not empty). The format of the printed tables is given in a later section. The character "X" or the character _^0 (control 0) can be typed by the user to abort the printing.

X or _^0 (control 0)
These characters can be typed by the user while the summary is printed when the program waits for a character to be typed (after a complete screen is printed); they abort the printing.

? This command causes printing of the list of available on-line commands.

4.4 The Summary Format

The summary is printed when the on-line command "S" is typed during the test, or during the conversation (at step 5), or if the user replies positively in step 2 of the conversation. The printing can be aborted by typing the character "X" or the character ^0 (control 0) while the summary is printed, when the program waits for a character to be typed (after a complete screen is printed).

The summary consists of three parts: the current status, the error log, and the statistics. The format of the three parts of the summary is described now.

CURRENT STATUS

This message is followed by the current status of the program.

TIME nnnn

This is the current time in seconds.

POWER RESTORES nnnn

This is the current number of power restores, as detected by the program.

LOOP MODE xxxx

xxxx is the current loop mode. It can be OFF, RETRYING, SLOW ON, or QUICK ON.

In the sequel, nnn1 represents a value associated with path 1, and nnn2 a value associated with path 2.

TX DEVICE nnn1 nnn2

This is the TX device address.

RX DEVICE nnn1 nnn2

This is the RX device address.

TX GOOD BLOCKS nnn1 nnn2

This is the number of blocks successfully transmitted.

RX GOOD LAST BLOCKS nnn1 nnn2

This is the number of blocks successfully received (not including transfers to the first of two RX buffers in case of a split buffer).

RX GOOD NOT LAST BLOCKS nnn1 nnn2 This is the number of blocks successfully received in transfers to the first of two RX buffers (in case of a split buffer).

TX BAD BLOCKS nnn1 nnn2

This is the number of blocks unsuccessfully transmitted.

RX BAD LAST BLOCKS nnn1 nnn2

This is the number of blocks unsuccessfully received (not including transfers to the first of two RX buffers in case of a split buffer).

RX BAD NOT LAST BLOCKS nnn1 nnn2

This is the number of blocks unsuccessfully received in transfers to the first of two RX buffers (in case of a split buffer).

4.5 Error Log

LAST -1 -2 -3 -4

This heading precedes the second part of the summary, the error log. Below this, information about up to 5 bad transfers is printed. The latest entry in the error log is printed under the word LAST, the previous entry under -1, etc. Each entry contains information about one or two paths.

TIME nnnn

This is the time in seconds when the error occurred.

POWER RESTORES nnnn

This is the number of power restores, as detected by the program when the error occurred.

LOOP MODE xxxx

This is the loop mode when the error occurred.

LAST RX PHASE xxxx

xxxx stands for YES or NO. It indicates whether the error occurred in the last RX phase, or in the first (of two) RX phases.

TX DEVICE nnn1 nnn2

This is the TX device address when the error occurred.

RX DEVICE nnn1 nnn2

This is the RX device address when the error occurred.

BAD DATA INFO

This message precedes information describing the first received bad word. The information is printed only if data error has really occurred; otherwise, dashes (----) are printed. Question marks (????) are printed if the program has not checked the data.

TX DATA nnn1 nnn2

This is the TX buffer data corresponding to the first RX bad word.

RX DATA nnn1 nnn2 This is the data of the first RX bad word.

XOR WORD nnn1 nnn2

This is the exclusive or of the above two words; i.e., every bit in error is on in this word.

TX WORD ADDR nnn1 nnn2

This is the address of the TX word corresponding to the first RX bad word.

BAD WORD ADDR nnn1 nnn2

This is the address of the RX bad word.

BAD WORD INDEX nnn1 nnn2

This is the index of the first RX bad word in the block (0,1, ...).

DATA FOUND ON SAME PAGE nnn1 nnn2

When a data error is detected, the program tries to determine whether the device has transferred the data to another word in the same page. This is done by complementing bits 1-12 in the RX bad word address one at a time, and comparing the contents of the found words to the TX data. Every bit whose complementing results in finding the desired data is set in the words printed in this line of the summary. For a single bus version, bits 1-15 of the RX bad word address are treated as described above.

DATA FOUND ON OTHERS nnn1 nnn2

When a data error is detected, the program tries to determine whether the device has transferred the data to a word in another page whose address within the page is identical to the RX word address. This is done by complementing bits 9-15 in the current page address (the next item in this list) one at a time, and comparing the contents of the words found on the obtained pages to the TX data. Every bit whose complementing results in finding the desired data is set in the words printed in this line of the summary. For a single-bus version, this words are zero.

CURRENT PAGE nnn1 nnn2

This is the address of the page containing both TX and RX buffers.

TX BUFFER ADDR nnn1 nnn2

This is the address of the TX buffer.

RX BUFFER-1 ADDR nnn1 nnn2

This is the address of the RX buffer. In case of a split buffer (HLC only), this is the address of the first of the two RX buffers.

RX BUFFER-2 ADDR nnn1 nnn2

This is the address of the second of the two RX buffers in case of a split RX buffer (HLC only).

CURRENT LENGTH nnn1 nnn2

This is the length (in words) of the current TX buffer.

TX STATUS nnn1 nnn2

This is the contents of bits 8-15 of the TX status register after the waiting period (for the transfer to complete) is over.

RX STATUS nnn1 nnn2

This is the contents of bits 8-15 of the RX status register after the waiting period (for the transfer to complete) is over.

In the following 16 items, the program prints the state of various bits in the device registers, and the results of various checks done by the program during or after the transfer. Each item normally has the form F--L. F (the first character) stands for "Y" (yes), "N" (no), or "-" (information is unknown). L (the last character) describes whether the program thinks that the corresponding item is good or bad. L stands for "G" (good), "B" (bad), "?" (the program does not know if it is good or bad), or "-" (information is unknown but the program does not expect it). The format of an item in which L= "B" is F**B (and not F--B).

TX DEVICE COMPLETED F--L F--L

This indicates whether the TX device has completed the transfer (TX active bit of the TX device is 0).

RX DEVICE COMPLETED F--L F--L

This indicates whether the RX device has completed the transfer (RX active bit of the RX device is 0).

TX PID WRITTEN F--L F--L

This indicates whether the TX PID level of the TX device was found in the PID.

RX PID WRITTEN F--L F--L

This indicates whether the RX PID level of the RX device was found in the PID.

DATA OK F--L F--L

This indicates whether the data in the RX buffer was equal to the data in the TX buffer (or a data error has been detected).

PADDING WORD WRITTEN F--L F--L

This indicates whether the data in the last RX buffer was followed by a padding word (_^H8000) in case of an HLC.

NO ERROR _& LAST BUF=1 F--L F--L

This indicates whether the corresponding bit of the RX device was set.

MEMORY QUIT F--L F--L

This indicates whether the program got a quit while accessing one of the memory buffers.

TX PID QUIT F--L F--L

This indicates whether the program got a quit from the PID corresponding to the TX device.

RX PID QUIT F--L F--L

This indicates whether the program got a quit from the PID corresponding to the RX device.

QUIT FROM TX DEVICE F--L F--L

This indicates whether the program got a quit while accessing the TX device.

QUIT FROM RX DEVICE F--L F--L

This indicates whether the program got a quit while accessing the RX device.

TX DEVICE QUIT =1 F--L F--L

This indicates whether the TX quit bit of the TX device was set (the TX device got a quit).

RX DEVICE QUIT =1 F--L F--L

This indicates whether the RX quit bit of the RX device was set (the RX device got a quit).

TX DEVICE ERROR =1 F--L F--L

This indicates whether the error bit of the TX device was set.

RX DEVICE ERROR =1 F--L F--L

This indicates whether the error bit of the RX device was set.

GLOBAL COUNTERS

This line precedes the values of several important counters at the time of the error. The meaning of each of these counters was explained earlier in this section.

```
# TX GOOD BLOCKS                nnn1 nnn2
# RX GOOD LAST BLOCKS           nnn1 nnn2
# RX GOOD NOT LAST BLOCKS       nnn1 nnn2
# TX BAD BLOCKS                 nnn1 nnn2
# RX BAD LAST BLOCKS            nnn1 nnn2
# RX BAD NOT LAST BLOCKS        nnn1 nnn2
```

STATISTICS:

This line precedes the third part of the summary, the statistics.

BAD PAGES:

This line precedes a list of pages involved in various errors. These memory pages are not necessarily bad pages; the transfer of data from/to these pages failed.

MIN BAD WORD BLOCK INDEX nnn1 nnn2

This is the smallest index (0,1, ...) of a bad word within the block.

MAX BAD WORD BLOCK INDEX nnn1 nnn2

This is the largest index (0,1, ...) of a bad word within the block.

MIN BAD WORD PAGE OFFSET nnn1 nnn2

This is the smallest offset of a bad word within the page.

MAX BAD WORD PAGE OFFSET nnn1 nnn2

This is the largest offset of a bad word within the page.

MIN BAD BLOCK LENGTH nnn1 nnn2

This is the smallest length (in words) of a block involved in an error.

MAX BAD BLOCK LENGTH nnn1 nnn2

This is the largest length (in words) of a block involved in an error.

OR OF XOR WORDS nnn1 nnn2

This is the logical inclusive or of all XOR words; i.e., every bit found to be in error in at least one data error is set in this word.

MAJORITY BITS INFO:

This line precedes several lines, each describing whether a bit in the majority of the words of some type was 1 or 0. The information is printed in the form:

<word type> =0/1 b b b b b b b b

Each b is a group of 4 bits; each group of 4 b's describes all 16 bits on a path. If the bit in the majority of the words of the type <word type> is 0/1 the corresponding bit is set in one of the b's. The possible word types are listed below.

```
XOR WORD =1                 b b b b  b b b b
BAD WORD DATA =1            b b b b  b b b b
BAD WORD DATA =0            b b b b  b b b b
BAD WORD PAGE OFFSET =1 b b b b  b b b b
BAD WORD PAGE OFFSET =0 b b b b  b b b b
SUMMARY END
This line ends the summary.
```

4.6 The Console Lights

During the DMA test the console lights display information that shows how the test progresses. The address lights contain information about path 1, and the data lights contain information about path 2. Each such information contains 4 groups; each containing the least significant bits of some counter. The order of these counters as appears on the console lights is:

<path 1>	TX-GOOD	RX-GOOD	TX-BAD	RX-BAD
<path 2>	TX-GOOD	RX-GOOD	TX-BAD	RX-BAD

If the user wants to use the console he first has to press ATTENTION (without RESET), and this stops the displaying. An additional pressing of the ATTENTION button restores the displaying.

The user can cause the program to display other information in the console lights. ALITSP (currently at location _^H106) and DLITSP (currently at location _^H108) are pointers to the words to be displayed in the ADDRESS LIGHTS and the DATA LIGHTS respectively.

5 EXHIT

5.1 Loading and Starting the Program

Before loading the program all busses should have power turned on.
Power restart interrupts should be enabled (S71 in the console in the middle position which enables the power recovery int). Line frequency interrupts (JIFFY) should be enabled (S70 in the console in the ON position). The bus reset timers of all busses should be in the OFF (up) position.

First PSTOP must be used to halt all processors. Then EXHIT is loaded according to the normal procedure. Then HIT should be loaded, as instructed by EXHIT. HIT automatically starts EXHIT after it is loaded, and at that point the conversation between EXHIT and the user starts.

To restart the EXHIT program from the console RESET-ATTENTION can be used. If this fails, EXHIT can be manually restarted at its beginning address, ^h2000.

If from some reason EXHITL fails before it starts loading EXHIT and should be restarted, the best approach is to reload it. However, it can be restarted by RESET-ATTENTION, or manually at its starting address, ^h2100 (e.g., after the TTY address is modified).

EXHITL and EXHIT assume a TTY at ^hFA00, however, the TTY address can be modified in each of these programs. The TTY address is stored in both programs at TTYADD. The current address of TTYADD is ^h2016 in EXHIT, and ^h2104 in EXHITL.

5.2 Conversation With the User

At the first time EXHIT is loaded it prints the message:

LOAD HIT (nnnn)

and then halts. At that point HIT (version nnnn) should be loaded, and then the conversation actually starts.

During the conversation the user has to answer the program questions.
The answers to the questions are:

Y- yes, N- no, or E- exit.

"E" means exit and restart the conversation.

Each of the above characters must be followed by a carriage-return. If an illegal pair of characters is typed by the user, the program prints the string "? (Y/CR,N,E)" and waits for a legal response.

The program interprets a single carriage-return as the character "Y" followed by a carriage-return.

The messages and questions printed by the program are described next.

1. EXHIT (mmmm)

The program name and the version number (mmmm) are printed first.

2. ANY COMMAND? (Y/CR,N,E)

If the user replies positively he can then type any of the on-line commands. The main purpose of this feature is to allow the user to examine some location via the OPEN command, to print one of the error tables, or to print the current status of the system after the test was aborted and before the conversation continues. However, other on-line commands can be used here as well. The conversation continues only after a negative reply is given.

Unless the user has some specific purpose, a negative reply can be given to the above question.

3. CHANGES? (Y/CR,N,E)

This question is asked only after the conversation has been completed at least once. If a negative reply is given, no changes in the configuration tables are done and the conversation continues in step 22; otherwise, the conversation proceeds in the next step.

4. BUSSES CHANGES? (Y/CR,N,E)

A negative reply means that there is no change in the lists of I/O busses, RTC's, and memory busses. Note that it may occasionally happen that HIT deletes an I/O bus or a memory bus from its tables after a remote bus power fail and doesn't update correctly the tables when power is restored. In such cases the busses information should be entered again by replying positively at this step.

If a negative reply is given, the busses information is printed. In the first time the conversation is conducted the default configuration is assumed as explained in the following steps.

5. ALL I/O BUSSES? (Y/CR,N,E)

If a positive reply is given, the program finds the existing I/O busses (by searching for PID's) and inserts the first two busses addresses to the appropriate HIT table; this is the default answer. If a negative reply is given, the program prints the message:

SELECT (Y/CR,N,E)

followed by the list of existing I/O busses, each of which should either be selected (by typing Y/CR) or not selected. At least one I/O bus must be selected.

USE aaaa RTC? (Y/CR,N,E)

At this step the user is required to select the RTC's to participate in the test. The question is asked about existing RTC's on selected I/O busses; aaaa is the RTC address (e.g., E006). The default answer is "Y".

When this step is completed, the selected I/O busses and the selected RTC's are printed.

7. ALL MEM BUSSES? (Y/CR,N,E)

If a positive reply is given, the program finds the existing memory busses and inserts the addresses of the first two busses to the appropriate HIT table; this is the default answer. If a negative reply is given, the user is requested to type the four digit hexadecimal addresses of up to two memory busses (e.g., 0000, 4000).

8. ALL PAGES ON aaaa BUS? (Y/CR,N,E)

The user is required to specify whether all pages on memory bus aaaa are to be tested. If a positive reply is given, the program finds the number of pages on the aaaa bus and enters the information in the appropriate HIT table; this is the default answer. Otherwise, the program prints the message:

PAGES=

and the user has to type a four digit hexadecimal number specifying the number of pages to be tested on the aaaa bus (e.g., 0006).

9. CHECK aaaa BUS VIA ALL I/O BUSSES? (Y/CR,N,E)

if a positive reply is given, HIT will test the aaaa memory bus through all DMA devices on all selected I/O busses; this is the default answer. If a negative reply is given, the program prints the message:

SELECT (Y/CR,N,E) followed by the list of selected I/O busses. The user has to select (by typing Y/CR) the I/O busses whose devices will participate in testing the aaaa memory bus, or not to select them (by typing N).

When this step is completed, the gathered information about the selected memory busses is printed. It should be noted that while giving the memory busses information, the user does not have to take into account the fact that a part of EXHIT actually resides in common memory; i.e., the page in which EXHIT is stored can be included in the pages on one of the tested busses.

10. PROGRAM IS IN COMMON PAGE-pppp. CHANGES? (Y/CR,N,E)

As pointed out in the previous step, the user can specify the memory busses information without paying attention to the question "where is EXHIT stored?" However, HIT does not test that page in order not to destroy the progrm (it thinks that DDT is stored in that page).

At this step the map address of the page containing EXHIT is printed and the user has a chance to change that page address by replying positively. If a positive reply is given, the program prints:

MAP=

and the user has to type the four digit hexadecimal address of a new page to contain EXHIT. Changing EXHIT page enables the testing of all common memory pages.

11. DMA DEVICES CHANGES? (Y/CR,N,E)

A negative reply means that there is no change in the information concerning the DMA devices. Note that it may occasionally happen that HIT marks DMA devices as being on a failed I/O bus when that bus has a power fail, and does not update the information when power is restored to that I/O bus. In such cases, the DMA information should be entered again by replying positively at this step.

If a negative reply is given, the DMA devices information is printed. In the first time the conversation is conducted default answers are assumed as explained in the following steps.

12. ALL DMA DEVICES? (Y/CR,N,E)

If a positive reply is given, the program finds all DMA devices and inserts their addresses to the appropriate HIT table; this is the default answer.

If a negative reply is given, the program prints the message:

SELECT (Y/CR,N,E)

followed by the list of DMA devices (including the address, type, and name for each device), each of which should either be selected (by typing Y/CR) or not selected.

13. ALL DMA DEVICES XPATCHED? (Y/CR,N,E)

If a positive reply is given, the program skips to the next step; this is the default answer.
If a negative reply is given, the program prints the message:

SELECT (Y/CR,N,E)

followed by the list of selected DMA devices. The user has to identify the crosspatched devices (by typing Y/CR).
For modems which are not crosspatched, the program asks the question:

EXTERNAL LOOP BIT ON? (Y/CR,N,E) to which the user has to answer in the obvious way.

14. COUPLED DEVICES? (Y/CR,N,E)

This question is printed if there is a possibility that there are coupled devices in the system. Two devices are candidates to be a couple if both have the same device number and similar types associated with them, and if none of them is crosspatched.

If a negative reply is given, the program skips to the next step; this is the default answer. If a positive reply is given, the program prints the details of suspected couples and the user has to identify the real couples (by typing Y/CR).

15. ROTATE TABLE BY 2? (Y/CR,N,E)

The addresses of the selected DMA devices are stored in a linear table of HIT. Experience has shown that the location in the table in which a device address is stored may affect the results of the test (success or errors). This question allows the user to rotate the contents of the DMA devices list two locations towards its beginning (by replying positively).
The question is repeated until a negative reply is given, thus, any even number of such rotations can be achieved. The default answer is "N".

When this step is completed, the detailes of the selected DMA devices are printed.

16. MSPM DEVICES CHANGES? (Y/CR,N,E)

A negative reply means that there are no changees in the information concerning the MSPM devices. In such case the MSPM devices information is printed. In the first time the conversation is conducted default answers are assumed as explained in the following steps.

ALL MSPM DEVICES? (Y/CR,N,E)

If a positive reply is given, the program finds all BLI devices and inserts their addresses to the appropriate HIT table; this is the default answer. If a negative reply is given, the program prints the message:

SELECT (Y/CR,N,E)

followed by the list of MSPM device addresses, each of which should either be selected (by typing Y/CR) or not selected.

If the answer is Yes, the program prints the message:

LINE a

The "a" is the corresponding line number from 0 to 3. each of which should either be selected or not selected.

18. ALL MSPM DEVICES XPATCHED? (Y/CR,N,E)

If a positive reply is given, the program skips to the next step; this is the default answer.

If a negative reply is given, the program prints the message:

SELECT (Y/CR,N,E)

followed by the list of selected MSPM devices. The user has to identify the crosspatched devices (by typing Y/CR).

If the answer is NO, the program prints the message:

LINE a

The user has to reply by Yes or No. If the answer is yes, the line is crosspatched.

When this step is completed, the details of the selected MSPM devices are printed.

19. PROC CHANGES? (Y/CR,N,E)

A negative answer means that there is no change in the list of processors participating in the test. In such case the processors numbers are printed. In the first time the conversation is conducted the default answers are assumed as explained in the following step.

20. ALL PROC'S? (Y/CR,N,E)

If a positive reply is given, the program finds all existing processors, tries to halt each of them, and inserts the numbers of those which have been successfully halted to the appropriate HIT table; this is the default answer.
If a negative reply is given, the program prints the message:

SELECT (Y/CR,N,E)

followed by the list of existing processor numbers, each of which should either be selected (by typing Y/CR) or not selected. Note that the master processor is automatically selected since this is a requirement of HIT.

When this step is completed, the list of processors participating in the test is printed.

21. TST OPTIONS CHANGES? (Y/CR,N,E)

A negative reply means that there is no change in the test options (which are defined below). If a negative reply is given the test options are printed for all processors in one table. Note that before slave processors (all processors except the master) are activated, the contents of the printed table concerning them is meaningless. If the reply is positive, the program prints the message:

SELECT (Y/CR,N,E) followed by the list of possible options. For each of the options the user has to answer either positively, or negatively. The possible options and the default answers are:

```
BACKGROUND ONLY        (N)
WAIT ON PWR RESTART    (N)
IGNORE REM INTERRUPTS  (N)
JIFFY'S TST            (Y)
MEM TST                (Y)
QUITS CAUSED           (Y)
SIZE OF BUFFER FIXED   (N)
DATA FIXED             (N)
LOAD BUFFER EACH TIME  (Y)
ADDR OF BUFFER FIXED   (N)
```

If a fixed size buffer is selected, the program prints the question:

SIZE=007E

The user can select this default fixed size (by typing Y/CR), or after typing "N" specify another size of buffer by typing a four digit hexadecimal number.

If a fixed data is selected, the program prints the question:

DATA=AAAA

The user can select this default fixed data (by typing Y/CR), or after typing "N" specify another fixed data to be used, by typing a four digit hexadecimal number.

The options selected in the above process hold initially for all activated processors (since the HIT program is copied to each processor memory); however, while HIT is running these options can be modified for each processor separately, or for all processors together via the on-line commands.

22. MIN BUFLIM IS CHOSEN

After the test configuration is determined in the previous steps the program calculates a number which indicates for HIT how each of the common memory pages should be divided; this number (BUFLIM) is calculated according to the number of the DMA devices, and the number of the common memory pages to be tested.

BUFLIM indicates the boundary between the lower part of the page, which is used for memory tests; and the upper part of the page, which is used for DMA devices tests.

The program tries to keep this limit as high as possible, but if the resulted number equals to the minimum allowed (^h1800) the above message is printed. It may happen if relatively many DMA devices and few common memory pages are selected. The program continues even in this case, but there are good chances that HIT will fail (e.g., will enter an infinite loop which can be easily observed since the console lights will not blink). If HIT fails, the conversation can be restarted (by an on-line command, or if this fails, by RESET-ATTENTION) and the tested configuration can be changed (e.g., less DMA devices to be tested).

23. START ALL PROC'S? (Y/CR,N,E)

If a negative reply is given, the conversation continues in the next step.
If a positive reply is given, EXHIT terminates the conversation and activates HIT on the master processor only. However, while EXHIT is later polled by HIT, it starts all selected processors (after a delay of 2-3 seconds). This starting procedure includes halting each processor via its registers (or killing the processor), copying the program to each of the selected processors, and then activating the processors one at a time.

The program prints the numbers of the processors during the activating step; each number is followed by one of the words: DONE, or CAN'T. The later word is printed if EXHIT can't activate the indicated processor (it gets too many quits).

It should be noted that due to limitations of the PLURIBUS, EXHIT cannot check whether the selected processors are really running. Thus, the user has to check it by watching the console lights for a few seconds. If some of the processors are not running, they can be restarted by the on-line commands. It may be helpfull to switch the I/O bus used for BBC references by the on-line commands if normal trials to restart the processors fail.

Another note is in place here. It is recommended to have a copy of the HIT program in each of the processors in the system; in particular in the local memories of both processors at least one of which is selected to run HIT. This decreases the vulnerability of HIT. It can be easily achieved by selecting all processors (in steps 19-20 above) and replying positively in this step. The PROC RESTART on-line command can also be used for achieving the same effect. When done, the conversation can be restarted and the configuration can be changed in the conversation.

24. START MASTER? (Y/CR,N,E)

If a negative reply is given, the conversation continues in the previous step. If a positive reply is given, EXHIT terminates the conversation and activates HIT on the master processor only. The on-line commands can be used to restart the other processors after this point.

5.3 The On-Line Commands

While HIT is running it polls EXHIT (thinking that it polls DDT). This enables EXHIT to recognize on-line commands and to print information on the TTY. The on-line commands can also be given at the beginning of the conversation (see step 2 of the conversation). The commands are described below. In all cases, the user has to type a character and the program comlements the command (and possibly waits for additional character or some data). In the following descripiton, the information typed by the user is underlined; upper case letters should be typed as in the examples, lower case

letters stand for data that the user has to provide. A summary of EXHIT on-line commands appears at the end of this document.

1. CLEAR ERRORS

This command should clear the error tables of all processors currently running HIT, and clear the time counter. Processors which are currently not running HIT, or have some troubles ignore the command.

2. DISPLAY IN (D/A) DATA LITES: ADDR=aaaa or
DISPLAY IN (D/A) ADDRESS LITES: ADDR=aaaa

The Display command allows the user to select the locations to be displayed in the console lights. aaaa is the address of the location to be displayed in the data/ address lights. Information from HIT variables page can be displayed by using addresses in the range ^h4000-^h5FFE.

The default HIT locations displayed in the console lights are: RUNL- in the data lights, and CERRL- in the address lights.

3. ERROR TABLE (GLOBAL)

This command is used to print a global error table. This table contains condensed information about each error in the error table of each of the processors selected to run HIT.

The errors found by each processor are printed in one column, three lines per error. The first line indicates the error name; the second line contains some address associated with the error; and the third line indicates the global time (in seconds) at which the error occurred. All printed numbers are hexadecimal ones.

An example of such an error entry is:

```
NAME:   -DVTO-DATA-
ADDR     E150 0400
TIME     0015 0012
```

We now give the names of the errors and the meaning of the addresses for each of the various error types discovered by HIT.
The hexadecimal numbers in parentheses indicate the HIT error types.

(1) DATA
A data error during memory test. The address specifies the contents of the map register.

(2) LOCK
A lock error. The address specifies the contents of the map register.

(3) PIDR
A PID read error. The address specifies the PID address (e.g., E000).

(4) CQIT
A quit while accessing a location in common memory address space (h4000-hBFFE). The address specifies the contents of the map register.

(4) IQIT
A quit while accessing a location in the common I/O address space (hC000-hFBFE). The address specifies the quit address.

(4) LQIT

space (0000-h3FFE, hFC00-hFFFE). The address specifies the quit address.

(5) DVTO
Device timeout. The address specifies the device address.

(6) PIDB
Device PID / busy. The address specifies the device address.

(7) DVTX
Device transmit. The address specifies the device address.

(8) DDAT

Device data. The address specifies the device address.

(9) MSab

MSPM error. "a" indicates the specific error as can be found in the description of the MSPM error in step 24 (9) of this section. "b" indicates the error channel number. If the error is an error of a line, the channel number represents any channel on the line. If the error concerns the whole device, the number is always zero and has no meaning. If the channel number is eight or nine, it represents the Pluribus to Micro queue and the Micro to Pluribus queue respectively. The address specifies the device address.

(A) BLTO

Buffer lock timeout. The address specifies the number of the processor who had the lock when timeout occurred.

(B) MPID

Missing PID. The address specifies the PID address (e.g., E000).

(C) NJFY

No jiffies obtained. The address has no meaning.

4. HALT ALL CONFIRM (Y/CR,N)

This command is used to cleanly halt all processors currently running HIT. The master sets a flag in common memory, and each processor halts when observing that the flag is set. The master restarts the conversation.

There is a similar command (R) which halts each processor via its registers and then restarts the conversation. The HALT command is preferable; however, if some processor does not respond to the HALT command, the RESTART command can be used.

5. I/O BUS FOR BBC =iiii

EXHIT uses one of the selected I/O busses for BBC references. The user can switch to the other I/O bus by using this command. iiii (printed by the program) is the I/O bus that will be used from now until the next I command, or until the conversation is restarted and I/O busses information is modified.

6. OPEN: aaaa P- pp rrrr wwww <terminator> or

OPEN: aaaa P- pp VIA I/O BUS-iiii rrrr wwww <terminator or

OPEN: aaaa PAGE-mmmm rrrr wwww <terminator>

The open command allows the user to access and optionally modify any location in the whole system address space. The command should be carefully used in order not to destroy the programs.

aaaa is the address to be opened, or the character ESC which stands for the previously opened location.

pp is the processor number (e.g., 12) in whose address space the location should be opened, or the character ESC which stands for the previously selected processor.

mmmm is the map address (e.g., 0000, or 0200) of the common memory page one of whose locations is to be opened, or the character ESC which stands for the previously selected page.

iiii is the current I/O bus used for BBC.

rrrr is the contents of the opened memory location.

wwww is an optional field which specifies the new contents to be written into the opened location.

<terminator> is one of the following characters:

Carriage-return which confirms the new contents (if typed), and terminates the command.

Line-feed which confirms the new contents (if typed), and automatically opens the next location.

^ (up arrow) which confirms the new contents (if typed), and automatically opens the previous location.

Any other character used as a terminator, aborts the OPEN command and does not confirm the new contents of the opened location (space is convinient to use).

If quit is detected while trying to access the opened location the word QUIT is printed by the program, and the OPEN command is aborted.

7. RESTART CONFIRM (Y/CY,N)

This command is used to halt (kill) all selected processors via their registers and then to restart the conversation. The list of processor numbers is printed; each processor number is followed by one of the words: DONE or CAN'T. See also the HALT command described earlier.

8. STATUS:

This command is used to get the current status of the system. It prints the current global time, followed by several error lists (if any) which are self explanatory. Then, all the current test parameters are printed.

9. PROC: <sub-command> or

PROC -ALL: <sub-command>

The sub-command indicated by <sub-command> is executed for processor number pp or for all selected processors depending on the typed information. pp must denote a selected processor, otherwise the program prints the message: "? (UNKNOWN)" and aborts the command.
Upon completion of executing the command for the selected processor(s) the program prints the processor number followed by one of the words DONE or CAN'T.

The descriptions of the available PROC sub-commands appear in the following steps.

10. PROC -pp: ADDR OF BUFFER FIXED (Y/CR,N) or

PROC -ALL: ADDR OF BUFFER FIXED (Y/CR,N)

The A sub-command allows the user to specify whether the buffer address is fixed; the default is N.

11. PROC -pp: BACKGROUND ONLY (Y/CR,N) or
    - -- -

    PROC -ALL: BACKGROUND ONLY (Y/CR,N)
    - - -

The B sub-command allows the user to specify whether the selected processor runs in background only; the default is N.

12. PROC -pp: DATA FIXED (Y/CR,N) or
    - -- -

    PROC -ALL: DATA FIXED (Y/CR,N)
    - - -

The D sub-command allows the user to specify whether to use fixed data; the default is N. If a positive reply is given, HIT uses the fixed data selected in the conversation (step 21); the default fixed data is ^hAAAA.

13. PROC -pp: GO or
    - -- -

    PROC -ALL: GO
    - - -

The G sub-command allows the user to reactivate a halted processor which already has a copy of the HIT program, without copying the program again (in contrast to the R sub-command).

14. PROC -pp: HALT or

PROC -ALL: HALT

The H sub-command is used to cleanly halt a specific processor, or all selected processors except the master. EXHIT sets a flag in the local memory of the selected processors and they halt when detecting that the flag is set.

A processor halted by the H command can be later restarted by one of the sub-commands G (GO) or R (RESTART).

If a processor does not respond to the H sub-command, the K (KILL) sub-command can be used to halt that processor via its registers. The H sub-command is preferable since the K (KILL) sub-command may introduce errors.

15. PROC -pp: IGNORE REM INTERRUPTS (Y/CR,N) or

PROC -ALL: IGNORE REM INTERRUPTS (Y/CR,N)

The I sub-command allows the user to specify whether to ignore remote bus interrupts; the default is N.

16. PROC -pp: JIFFY'S TST (Y/CR,N) or

PROC -ALL: JIFFY'S TST (Y/CR,N)

The J sub-command allows the user to specify whether to perform JIFFY test; the default is Y.

17. PROC -pp: KILL or

PROC -ALL: KILL

The K sub-command allows the user to halt a processor via its registers. A similar sub-command is H. EXHIT prints whether the processor was running or halted when the kill sub-command was executed. Additional information is printed in some cases:

If the killed processor had executed an intentional illegal instruction to denote a catastrophic hardware trouble, a message describing the trouble is printed (see a later section).

If the killed processor is halted in a special address denoting a deliberate halt (such as following the H sub-command), the message "DELIBERATE HALT" is printed.

One of the uses of the K command is to try to find out why a processor is not running; if it detected some catastrophic trouble a message is printed.

18. PROC -pp: LOAD BUFFER EACH TIME (Y/CR,N) or

PROC -ALL: LOAD BUFFER EACH TIME (Y/CR,N)

The L sub-command is used to specify whether to load the buffer each time; the default is Y.

19. PROC -pp: MEM TST (Y/CR,N) or

PROC -ALL: MEM TST (Y/CR,N)

The M sub-command allows the user to specify whether to perform common memory tests; the default is Y.

20. PROC -pp: QUITS CAUSED (Y/CR,N) or

PROC -ALL: QUITS CAUSED (Y/CR,N)

The Q sub-command allows the user to select whether to cause deliberate quits; the default is Y.

21. PROC -pp: RESTART (B/S) BOTH CONFIRM (Y/CR,N) or

PROC -pp: RESTART (B/S) SINGLE CONFIRM (Y/CR,N) or

PROC -ALL: RESTART CONFIRM (Y/CR,N)

The R sub-command allows the user to restart both processors on a bus, a single processor on a bus, or all selected processors. In the case of the S option, both processors are halted, however, only one of them is activated. The master cannot be restarted by this sub-command; the only way to restart the master is by restarting the conversation (or via the console).

When the command is confirmed, EXHIT executes a sequence of operations. First, all processors on the relevant processor busses are cleanly halted by EXHIT. Then, after a delay of 2-3 seconds, all those processors are killed (halted via their registers). In the A (ALL) case, the C (CLEAR ERRORS) command is executed next. Then, the program is copied to all processors that have to be restarted. Finally, all the relevant processors are activated.

From time to time, execution of this sub-command causes errors to be detected by HIT; normally these errors can be cleared (by the C command), and no new errors are detected. If the command does not work properly, it may be useful to switch to the other I/O bus used for BBC (via the I command).

22. PROC -pp: SIZE OF BUFFER FIXED (Y/CR,N) or

PROC -ALL: SIZE OF BUFFER FIXED (Y/CR,N)

The S sub-command allows the user to specify whether the buffer size is fixed; the default is N. If a positive reply is given, HIT uses the fixed buffer size selected during the conversation (step 21); the default fixed size is ^h007E.

23. PROC -pp: WAIT ON PWR RESTART (Y/CR,N) or

PROC -ALL: WAIT ON PWR RESTART (Y/CR,N)

The W sub-command allows the user to specify whether a processor has to halt on power restart or to resume execution of HIT; the default is N.

24. PROC -pp: ERROR TABLE or

PROC -ALL: ERROR TABLE

The E sub-command allows the user to examine the detailed error table associated with the specified processor(s). Note that one global error table can be obtained via the E command.

We now describe each of the possible errors. The hexadecimal numbers in parentheses indicate the HIT error types.

(1) DATA

MAP=mm ADDR=aaaa WROTE=wwww READ=rrrr RTRY=yy TIME=tttt

A data error during memory test.

mmmm- the contents of the map register.
aaaa- the data address.
wwww- the written data.
rrrr- the read data.
yy- can be either OK, or BAD.

(2) LOCK

MAP=mmmm ADDR=aaaa WROTE=wwww READ=rrrr RTRY=yy TIME=tttt

A lock error. The meanings of the various fields are identical to case (1) above.

(3) PIDR

ADDR=aaaa BADLVL=ll TIME=tttt

A PID read error.

aaaa- the PID address.
ll- the bad pid level read by HIT.

(4) QUIT LOCAL or,
QUIT CMEM or,
QUIT CI/O followed by:

MAP=mmmm ADDR=aaaa PC=cccc RTRY=y TIME=tttt

A quit while accessing a location in the local address space, in the common memory address space, or in the common I/O address space respectively.

mmmm- The contents of the map register (only in case of CMEM).
aaaa- the quit address.
cccc- the program counter at the instruction that caused the quit.
y- the number of retries before success. 9 indicates a solid quit.

(5) DEVICE T.O.

ADDR=aaaa	IMAP=iiii	OMAP=oooo		TIME=tttt
REND=re	RSTA=rs	RCOM=r	RQIT=x	RERR=y
TEND=te	TSTA=ts	TCOM=t	TQIT=z	TERR=w

A device timeout error.

aaaa- the device address.
iiii- the eight most significant bits of input buffer pointer + eight 0 bits (similar to a page map address).
oooo- the eight most significant bits of output buffer pointer + eight 0 bits (similar to a page map address).
re- the eight least significant bits of the receive end pointer.
te- the eight least significant bits of the transmit end pointer.
rs- the receive status.
ts- the transmit status.
r- receive complete (Y/N); obtained from the receive status.
t- transmit complete (Y/N); obtained from the transmit status.
x- device receive quit (Y/N); obtained from the receive status.
z- device transmit quit (Y/N); obtained from the

transmit status.
y- device receive error (Y/N); obtained from the receive status.
w- device transmit error (Y/N); obtained from the transmit status.

(6) DEVICE PID/BUSY

```
ADDR=aaaa  IMAP=iiii  OMAP=oooo              TIME=tttt
REND=re    RSTA=rs    RCOM=r     RQIT=x     RERR=y
TEND=te    TSTA=ts    TCOM=t     TQIT=z     TERR=w
```

A device PID / busy error. The meanings of the various fields are identical to case (5) above.

(7) DEVICE TX

```
ADDR=aaaa  IMAP=iiii  OMAP=oooo              TIME=tttt
REND=re    RSTA=rs    RCOM=r     RQIT=x     RERR=y
TEND=te    TSTA=ts    TCOM=t     TQIT=z     TERR=w
```

A device transmit error. The meaning of the various fields are identical to case (5) above.

(8) DEVICE DATA

```
ADDR=aaaa  IMAP=iiii  OMAP=oooo    TIME=tttt
XPCTD=xxxx RCVD=rrrr  DISP=d
```

A device data error. The meanings of the fields in the first line are identical to case (5) above. The other fields are:

xxxx- the expected data.
rrrr- the received data.
d- the bad data displacement in words modulo 16.

(9) MSPM

QUIT MSPM PAGE= nnnn

The page containing codes of MSPM TEST and its variables is not good or the map variable of that page is bad. "nnnn" is the map of the page. The error code is zero.

MSPM

ADDR=aaaa CODE=cc TIME=tttt

An MSPM error.

aaaa-the device address.
cc- the error code.

1- the micro gets quits from common memory
2- a dma chip gets quit from common memory
3- bso never starts.

MSPM

ADDR=aaaa CODE=cc CHANNEL=x TIME=tttt

4- bso channel timeout.
5- bso channel error. micro returns the buffer.
6- bso data transmission error.
7- bso line timeout. data is not transmitted nor received.
8- no buffer in the channel queue when needed.

The channel number 0 to 7 represents corresponding channel. If the error is a kind of line error, the channel number may be any channel associated with the line. When the channel number is 8, it is the Pluribus to Micro channel and '9' is the Micro to Pluribus channel.

(A) BUFFER LOCK T.O.

PROC=Pnn TIME=tttt

A buffer lock timeout. nn is the number of the processor who had the lock when timed out.

(B) MISSING PID

ADDR=aaaa XPCTD-LVL=ll TIME=tttt

A missing PID error.

aaaa- the pid address.
ll- the PID level which should have been read.

(C) NO JIFFIES

TIME=tttt

No JIFFY interrupts obtained.

5.4 Catastrophic Hardware Troubles

While HIT is running, or during the conversation, several severe troubles may occur that cause some error message to be printed (if possible) by EXHIT, and then the conversation is automatically restarted if possible or the master halts. If the trouble occurs while other processors are running HIT, they may continue and will probably complain about errors (i.e., will enter error information into their local error tables that can be examined later).

We first describe the catastrophic error messages associated with troubles detected by EXHIT (during the conversation, or while polled by HIT).

1. EXHIT COMN PAGE GONE

Each time the conversation is started, or EXHIT is polled, a test is made to see whether the common memory page containing a part of the EXHIT program exists. The message is printed if the page is not found.

If the trouble occurs at the beginning of the conversation, EXHIT halts; It can be restarted after the page is operational. If the trouble occurs while HIT is running, EXHIT enters a special mode in which it does not recognize any command; It returns control to HIT immidiately after it is polled. When the missing page re-appears, EXHIT starts recognizing the on-line commands again.

2. HIT COMN PAGE GONE

HIT keeps global information on the test in some common memory page. This message is printed when EXHIT tries to access this page for printing status information. EXHIT continues normally after this message is printed; however, some catastrophic behaviour can be expected (or it may have already occurred).

3. UNEXPECTED QUIT @aaaa

This message is printed whenevr EXHIT gets an unexpected quit (unless the quit occurs while trying to access the TTY). After the message is printed, the master halts. The user can examine the registers, and restart the program. If the message is consistent, then the program listing must be used to analyze the cause of the quit.

4. UNEXPECTED INTERRUPT

This message is printed whenevr a level 2 or level 3 interrupt occurs while EXHIT or HIT are running. After the message is printed, the conversation is restarted.

5. ILLOP @aaaa

The message is printed whenever an illop occurs while EXHIT or HIT are running. An additional information may be printed following this message, as explained next; after this the conversation is restarted.

As mentioned earlier, some catastrophic troubles are detected by HIT. In such cases HIT executes special illegal operations which cause an ILLOP message to be printed by EXHIT. After printing the ILLOP message, EXHIT checks whether the ILLOP is one of HIT special ILLOP's. If this is the case, an error message identifying the severe error is printed (whenever possible); then the conversation is restarted.

Note that if a slave processor detects a catastrophic trouble, it executes a special illegal instruction and then halts without printing anything (the master processor is the only one that accesses the TTY). In order to find out the source of the trouble, the PROC KILL command (step 17 earlier) can be used.

The special error messages are described below.

1. NO PROC TABLE

This error message indicates that HIT found an empty processor table. When activating HIT via EXHIT, it cannot happen unless some trouble caused undesired side effects.

2. COMN PWR FAIL

This error message indicates that HIT got a level 1 interrupt indicating that some memory bus has a power failure, and HIT couldn't recover from that power fail interrupt due to one of the following reasons:

a. All memory busses have a power failure.

b. The bus containing HIT common information has a power fail and HIT does not find a free page on the other memory bus (if there are 2 busses).

c. The bus containing HIT common information has a power fail and HIT didn't succeed to copy this information to a page on the other memory bus befor the first memory bus totally died.

3. SAME PID TWICE

This error message indicates that HIT found two devices with the same PID level.

4. TOO LOW DEVICE PID LVL

This error message indicates that HIT found a device with too low PID level.

5. QUIT WITHIN QUIT

This error message indicates that HIT got a quit within a quit; additional information about this quit can be obtained by examining the special low core addresses _^h0028-_^h002E for an even processor number, or locations _^h0038-_^h003E for an odd processor number.

6. NO FAIL ON RMOT PWR FAIL INT.

This error message indicates that HIT got a level 1 interrupt and didn't find any bus who could cause the interrupt. This can happen as a result of some hardware problem, or if the following events occur:

a. Some remote bus (memory or I/O) has a power failure that causes a level 1 interrupt.

b. HIT recognizes the interrupt and marks that bus as having a power failure.

c. The bus has its power restarted but HIT does not recognize it (probabaly due to some timing problem in HIT).

d. The same bus has another power failure, and another level 1 interrupt is caused.

e. HIT does not check if this bus caused the level 1 interrupt since it thinks that this bus still has a power failure from the previous time; HIT doesn't find any bus that could cause the level 1 interrupt.

f. HIT executes the illegal operation, which in turn causes EXHIT to print the message.

Note that during the conversation which follows this message the information related to that bus (and to its DMA devices, if it is an I/O bus) should be re-entered, otherwise HIT will still think that the bus (and the devices) does not exist.

7. UNKNOWN INT. @ LVL 1

This error message indicates that an unknown levle 1 interrupt occurred.

8. UNKNOWN INT. @ LVL 4

This error message indicates that an unknown levle 4 interrupt occurred.

9. QUIT IN INIT.

This error message indicates that HIT got a quit during the initialization stage. The address that caused the quit is displayed by HIT in the address lights, and the instruction address is displayed by HIT in the data lights. Additional information can be obtained by examining the special low core locations ^h0028-^h002E for an even processor number, or locations ^h0038-^h003E for an odd processor number.

10. NO COMN MEM

This error message indicates that during the initialization stage HIT didn't find even a single page for its use on the selected memory busses (HIT ignores the EXHIT page).

11. BAD PROC. #

This error message indicates that during the initialization stage, the processor didn't find its number in the processor table.

The On-Line Commands Summary

CLEAR ERRORS
-

DISPLAY IN (D/A) DATA LITES: ADDR=aaaa
- ----

DISPLAY IN (D/A) ADDRESS LITES: ADDR=aaaa
- ----

```
ERROR TABLE (GLOBAL)
-

HALT ALL   CONFIRM (Y/CR,N)
-

I/O BUS FOR BBC =iiii
-


OPEN: aaaa P- pp rrrr wwww <terminator>
-     ----    --     ----

OPEN: aaaa P- pp VIA I/O BUS-iiii rrrr wwww <terminator>
      ----    --                    ----

OPEN: aaaa PAGE-mmmm rrrr wwww <terminator>
      ----     ----     ----
      <terminator> is: CR / LF / ^


RESTART   CONFIRM (Y/CY,N)
-

STATUS:
-


PROC -pp: ADDR OF BUFFER FIXED   (Y/CR,N)
-     --  -

PROC -ALL: ADDR OF BUFFER FIXED   (Y/CR,N)
-     -    -

PROC -pp: BACKGROUND ONLY        (Y/CR,N)
-     --  -

PROC -ALL: BACKGROUND ONLY        (Y/CR,N)
-     -    -

PROC -pp: DATA FIXED             (Y/CR,N)
-     --  -

PROC -ALL: DATA FIXED             (Y/CR,N)
-     ---  -

PROC -pp: ERROR TABLE
-     --  -
```

```
PROC -ALL: ERROR TABLE
-     ---  -

PROC -pp: GO
-     --  -

PROC -ALL: GO
-     -   -

PROC -pp: HALT
-     --  -

PROC -ALL: HALT
-     -   -

PROC -pp: IGNORE REM INTERRUPTS  (Y/CR,N)
-     --  -

PROC -ALL: IGNORE REM INTERRUPTS  (Y/CR,N)
-     -   -

PROC -pp: JIFFY'S TST            (Y/CR,N)
-     --  -

PROC -ALL: JIFFY'S TST            (Y/CR,N)
-     -   -

PROC -pp: KILL
-     --  -

PROC -ALL: KILL
-     -   -

PROC -pp: LOAD BUFFER EACH TIME  (Y/CR,N)
-     --  -

PROC -ALL: LOAD BUFFER EACH TIME  (Y/CR,N)
-     -   -

PROC -pp: MEM TST                (Y/CR,N)
-     --  -

PROC -ALL: MEM TST                (Y/CR,N)
-     -   -

PROC -pp: QUITS CAUSED           (Y/CR,N)
-     --  -
```

```
PROC -ALL: QUITS CAUSED                (Y/CR,N)
-    -   -

PROC -pp: RESTART (B/S) BOTH   CONFIRM (Y/CR,N)
-    --  -

PROC -pp: RESTART (B/S) SINGLE   CONFIRM (Y/CR,N)
-    --  -

PROC -ALL: RESTART   CONFIRM (Y/CR,N)
-    -   -

PROC -pp: SIZE OF BUFFER FIXED   (Y/CR,N)
-    --  -

PROC -ALL: SIZE OF BUFFER FIXED   (Y/CR,N)
-    -   -

PROC -pp: WAIT ON PWR RESTART    (Y/CR,N)
-    --  -

PROC -ALL: WAIT ON PWR RESTART    (Y/CR,N)
-    -   -
```

6 INVENTORY

6.1 Loading and Starting the Program

Before loading the program all busses should have power turned on. Power restart interrupts should be enabled (S71 in the console in the middle position which enables the power recovery int). Line frequency interrupts (JIFFY) should be enabled (S70 in the console in the ON position). The bus reset timers of all busses should be in the OFF (up) position.

First PSTOP must be used to halt all processors. Then DDT may be loaded if the operator wants to use it later (the INVENTORY program does not need DDT for its operation).

Then the program is loaded and starts automatically. To restart the program from DDT or from the console: INVENTORY starts at ^h100 and the printing program starts at ^h104.

RESET-ATTENTION can be used to restart INVENTORY until the master processor starts activating slave processors. After the printing program is started, it can be restarted by pushing "RESET" then "ATTENTION" on the contro panel.

A. Load correct DEVTST tape or cassette according to system configuration.

B. Press "Reset" on control panel.

C. Set start address in data lights on control panel if loading from cassette.

D. Press "Load" on control panel. Check the console to see if all processors and common memory are listed as per the SRN. If not, you are not running the correct version. If no display on console, check to see if your console is at the address the diagnostic expects to see it.

E. Verify next on the screen is start of questions.

6.2 Operation Of The Program

The program expects co-operation from the operator.
The amount of action required from the operator depends on the amount of information needed. For maximum information busses should be powered off when the program suggests it. The processor in whose local memory the program was loaded is called the master. Normally the master is the even processor on the bus containing the console and the auto-load, in other cases RESET-ATTENTION cannot be used. The master's buddy is called the buddy, and a remote processor is called remote. Any processor which is not the master is called a slave (i.e. a slave can be either the master's buddy or a remote processor).

First the master finds the configuration, while interacting with the operator. Then the program is copied to the buddy. The buddy finds the configuration while the master is halted. Then the master is reactivated by its buddy and starts activating remote processors one at a time. Each remote processor reactivates the master after

terminating.

If a slave fails after it was activated by the master, operation of the master can be resumed by RESET ATTENTION.

After all processors having at least 4 KW of local memory have been activated, the operator has the options of restarting the search program or printing the results.

After printing the results, the operator can reprint the results or restart the search program (except for 4 KW multi-bus systems).

6.3 The Search Program Messages

The messages printed on the TTY are preceded by the processor identification and have the form:

<PRO>.XXXX: <MESSAGE>

where <PRO> can be: MAST (master), BUDY (buddy), or RMOT (remote).

For an even processor, XXXX is the address of the bcm control register that corresponds to the processor bus. For an odd processor, XXXX is the above address + 1. A remote processor is activated by bbc using that address. The answers to the questions are:

Y- yes, N- no, or E- exit (to the end of the program).

Each of the above characters must be followed by a carriage return. If an illegal pair of characters is typed by the operator the program prints the string "? (Y/CR,N,E)" and waits for a legal response.

The program interprets a single carriage-return as the character "Y" followed by a carriage-return. The messages printed on the TTY and their meaning are explained below.

1. INV-MB or INV-SB

The version of the program is printed only by the master.

2. <PRO>.XXXX: ACTIVATED

The message is printed by each processor shortly after it is activated. Several checks are performed by the processor before printing the message:

a. Is it an odd or an even processor as expected by the master?
b. Is it connected to the same bcm control register the master used?
c. Is the program checksum correct? The message is not printed if the answer to any of these questions is negative, or in case of a hardware failure.

3. <PRO>.XXXX: SINGLE BUS? (Y/CR,N,E)

The message is printed by a single bus version or by a multi-bus version that didn't find maps or had troubles with the maps.

4. <PRO>.XXXX: FIX MAPS AND RESTART

The message is printed in a multi-bus version by a processor that received a negative answer to question 3.

5. <PRO>.XXXX: PWR OFF ALL BUSSES? (Y/CR,N,E)

The operator should reply positively if he wants a check that no devices exist on the processor bus in the address space allocated for common memory or common I/O.

6. <PRO>.XXXX: PWR OFF ALL BUSSES; PWR ME ON (2 SEC)

The message is printed by a processor that received a positive reply to question 5. The operator must power off all the busses in any order, then power on the processor bus containing that processor. After at least 2 seconds all other busses must be powered on.

7. <PRO>.XXXX: ALL BUSSES ON? (Y/CR,N,E)

After everything was checked on the processor bus, and the processor is ready for checking common memory, the message is printed. If the operator types the character "N", then the message is reprinted.

8. <PRO>.XXXX: PWR OFF MEMORY BUSSES? (Y/CR,N,E)

The message is printed after common memory was checked by the processor. If the operator wants information about common memory busses, he should reply positively. In this case he should power off memory busses, one at a time in any order, as explained next.

9. <PRO>.XXXX: PWR ON PREVIOUS;BUS, ANOTHER ONE? (Y/CR,N,E)

The operator should power on the previous memory bus which was powered off, and then reply whether there is another memory bus to power off, or not.

10. <PRO>.XXXX: PWR OFF NEXT; TYPE ANY CHAR; WAIT 2 SEC AND PWR ON

The message is printed after a positive reply to question 8 or to question 9. The operator has to power off the next memory bus, to type any character, and to wait for message 9. The memory bus must be powered on after message 9 is printed or after at least 2 seconds (if the TTY is on the powered off bus).

11. <PRO>.XXXX: PWR OFF I/O BUSSES? (Y/CR,N,E)

The message is printed after common memory busses were powered off and after I/O devices and busses were found. If the operator wants information whether devices and bcms are plugged in the correct bus, he should reply positively. In this case he should power off I/O busses, one at a time in the order dictated by the program. Combined M/I busses are discovered at this stage.

12. <PRO>.XXXX: PWR ON PREVIOUS; PWR OFF BUS: YYYY; TYPE ANY CHAR; WAIT 2 SEC AND PWR ON

The operator must power on the previous I/O bus, then power off the indicated I/O bus, then type any character (if TTY is not on the powered off bus). The operator has to power on the I/O bus after message 12 or 13 is printed or after at least 2 seconds (if TTY is on the powered off I/O bus).

13. <PRO>.XXXX: ALL BUSSES ON? (Y/CR,N,E)

The message is printed after I/O busses were powered off, and the processor is ready to check connections between memory and I/O busses. The period of this stage depends on the number of bad devices found on each I/O bus.

14. AST.XXXX: ACTIVATING PROCESSOR: YYYY; IF SLAVE STUCK, PUSH
ATTN

The message is printed by the master before activating a slave processor. If the slave does not immediately print message 2, or is stuck some time later, the master operation can be resumed by pushing RESET and ATTENTION.

15. <PRO>.XXXX: REACTIVATING MASTER

The message is printed by a slave before reactivating the master.

16. <PRO>.XXXX: CAN'T REACTIVATE MASTER The message is printed by a slave if it does not succeed in reactivating the master.

17. MAST.XXXX: REACTIVATED BY SLAVE

The message is printed by the master after it is reactivated by a slave processor.

18. MAST.XXXX: WHAT NEXT? (Y=PRINT, N=RESTART, E=DDT)

The message is printed when the end of the search program is reached. The operator can restart the search program (N), print the results (Y), or exit to DDT (E). In the case of 4 KW multi-bus version, replying with Y causes automatic loading of the printing program. 8 KW version does not allow transfer to DDT.

19. <PRO>.XXXX: UNEXPECTED QUIT AT: YYYY

The mssage is printed in the implied case. The program should be restarted and if the message is consistent the program listing must be used to analyze the cause.

20. MAST.XXXX: BAD RESULTS-PAGE= YYYY OR ZZZZ

The message is printed by the master if one of the 2 pages of common memory selected for storing the tables causes a quit.

21. MAST.XXXX: NO RESULTS

The message is printed by the master when the end of the search program was reached before results of all processors were transferred to the tables area.

6.4 Printing the Results

The printing program expects some information from the user, and prints the tables produced by the search program.

1. HARD COPY? (Y/CR,N,E)

The message is printed when the program is started.
If the operator replies negatively, implying a CRT device, then he should type any character to cause the printing of each table. If the typed character is E (exit), message 4 is printed.

If the operator replies positively, all tables are printed without pauses between tables. RESET-ATTENTION can be used to abort printing in this case and to restart the printing program. If the operator replies with E, then message 4 is printed.

2. BEGIN WITH TABLE 01? (Y/CR,N,E)

If a positive reply is given, the tables are printed starting with table 1. If a negative reply is given, message 3 is printed.

3. TABLE (XX)=

The user has to type a 2 digit decimal number specifying the first table to be printed.

4. WHAT NEXT? (Y=PRINT, N=RESTART, E=DDT)

This message is printed at the end of the tables' printing.

Y- causes reprinting the results.
N- causes restarting of the search program (if loaded).
E- causes transfer of control to DDT (if not 8 KW multi-bus version).

5. UNEXPECTED QUIT AT: XXXX

The message is printed in the implied case. The program should be restarted and if the message is consistent the program listing must be used to analyze the cause.

6. NO RESULTS

The message is printed if the printing program does not find results in the expected tables area.

6.5 The Printed Tables

Each table is preceded by the table number and an ordinal number associated with each table type. The table name is printed followed by a header which contains the processors' numbers (e.g. P10 P11 P32 P33).

Two fields are allocated for every processor bus even if there are not 2 processors on that bus. The width of each processor field is 4 characters. Every information line starts with a prefix of 0-15 characters, followed by 16 processor fields.

The last character may be the character "#", in this case the operator should pay attention to this line which usually contains some error. The character "#" is printed in tables associated with common resources if not all existing processors have seen the same objects. For local-view tables, the character "#" is printed if both processors on the local bus have not seen the same objects.

In most of the tables the information printed in each processor field is the character "1" if the object was found and nothing otherwise. Additional characters may be printed in a field and usually this indicates some trouble. The different tables are

described now as printed for a 2 processor bus system. In a single bus system, only tables 1, 3-5, and 19-21 are printed.

TABLE 1

PROCESSOR ACTIVATION

P10	P11	P32	P33
M	AS	AS	AN

The meaning of the characters is:

M- this is the master processor, all others are slaves.

A- this slave processor was activated by the master.

S- this slave processor successfully reactivated the master.

N- this slave processor did not reactivate the master by itself, and ATTENTION was pushed.

R- master got quit from processor registers.

H- master did not succeed in halting the slave.

F- master got a quit from the bbc window while transferring data from a remote slave.

T- master got a quit from the bbc window while transferring program to a remote slave, or quit in transferring to/from the master's buddy.

C- master got a quit from the bcm control register associated with this processor bus.

B- master got a quit from the bbc map register.

TABLE 2

	PROCESSOR BUS: XXXX			
	P10	P11	P32	P33
EVEN PROC.	1	1	1	1
ODD PROC.	1	1	1	1
PAGE-0000 KY-0	1	1	1	1
PAGE-2000 KY-0	1	1	1	1
PAGE-0000 KY-1	1	1	1	1
PAGE-2000 KY-1	1	1	1	1

The table contains the main objects on processor bus XXXX as seen by each processor in the system. Additional characters may be printed in the first 2 lines:

Q- quit from the processor registers.

H- the processor couldn't be halted.

B- bad processor, its buddy got quits from its registers or couldn't halt it.

TABLE 3

PROCESSORS, ROM AND CONSOLE (LOCAL VIEW)				
	P10	P11	P32	P33
EVEN PROC.	1	1	1	1
ODD PROC.	1	1	1	1
ROM	1	1		
CONSOLE	1	1		
QUIT TIME	TTT	TTT		

The table reflects what each processor saw on its bus. TTT is the (decimal) quit time, in microseconds, on the processor bus, as measured by the even processor on that bus. If TTT is out of the range 300..700, then the character "#" is printed at the end of the line.

An additional character may be printed in the first 2 lines:

B- bad processor, its buddy got quits from its registers or couldn't halt it.

TABLE 4

LOCAL MEMORIES (LOCAL VIEW)

ADDR	KEY	P10	P11	P32	P33
XXXX	0	1	1	1	1
YYYY	1	1	1	1	1
PWRED OFF?		T	T	T	T

The table contains the memory pages found by each processor on its processor bus. If all busses were powered off then 6 pages were searched (4 of them overlapping common memory), else only 2 pages were searched.

If all busses were powered off for a period of time which is too short (i.e. less than 2 seconds), local pages overlapping common memory pages may be printed. Therefore it is recommended to restart the program before trying to debug the hardware, while waiting longer period of times when all busses are off.

Additional characters may be printed:

M- multiply addressed page.

B- bad page, quits or write-read do not match.

K- key problem, page recognizes both keys. The last line in the printed table indicates if all busses were powered off (T=Y), or not (T=N). If at least one of the printed entries is the character "N" then the line is marked by the character "#" at its end.

TABLE 5

LOCAL I/O DEVICES (LOCAL VIEW)

ADDR NUM	TYPE TPID	NAME RPID	P10	P11	P32	P33
XXXX	YY	NNN	1	1	1	1
ZZ	UU	VV				
PWRED	OFF?		T	T	T	T

The table contains the I/O devices found by each processor on its bus. Two lines are printed for each device. ADDR is the device address. TYPE is the device type. If the master can't read the device type during the printing stage the character "?" is printed. NAME is a 3 characters mnemonic name.

If the device name is not known to the printing program the word NEW is printed in the NAME field. If the type is 0, the character "-" is printed in the NAME field. The second line printed for each device contains information for DMA devices (and for devices of type SLI), or the character strings "--" for other devices.
NUM denotes the contents of byte 2 in the device registers block which is normally the device number, or the code in case of a device of type SLI.

For a DMA device, TPID and RPID denote the transmit PID and the receive PID respectively. For a device of type SLI, TPID denotes the type of the second device and RPID denotes its code (there are two devices per each 16 bytes block in case of type SLI).
The last line in the printed table indicates if all busses were powered off (T=Y), or not (T=N). If at least one of the printed entries is the character "N" then the line is marked by the character "#" at its end.

TABLE 6

NOT CLEAR 100 BYTES REGIONS (LOCAL VIEW)

ADDR	P10	P11	P32	P33
XXXX	1	1		
YYYY			1	1
PWRED OFF?	T	T	T	T

The table contains a list of regions containing devices or memories in illegal addresses as seen by each processor on its bus.

Each region is ^h100 bytes long and the search is done in increments of ^h10 bytes. The information is available only if all busses were powered off.

If all busses were powered off for a period of time which is too short (i.e. less than 2 seconds), clear regions may be indicated as "not clear" ones. Therefore it is recommended to restart the program before trying to debug the hardware, while waiting longer period of times when all busses are off.

The last line in the printed table indicates if all busses were powered off (T=Y), or not (T=N). If at least one of the printed entries is the character "N" then the line is marked by the character "#" at its end.

TABLE 7

PROCESSOR I/O BUS BCMS (LOCAL VIEW)

I/O BUS	P10	P11	P32	P33
XXXX	10	10	32	32
YYYY	10	16	32	32
BAD ADDRESSES		#		

The table contains the bcms' addresses associated with each processor bus as found by each processor on that bus. The address printed is relative to the starting address of the I/O bus. Only the 2 least significant hex digits are printed (the others are 0).

The last line is printed if the information in a processor field is not identical on all I/O busses.

TABLE 8

	COMMON MEMORY PAGES			
MAP	P10	P11	P32	P33
XXXX	1	1	1	1
YYYY	1	1	1	1

The table lists the common memory pages as found by each processor. Additional characters may be printed:

B- bad page. Quit or write-read do not match.

M- multiply addressed page.

r- read-only page.

TABLE 9

	COMMON MEMORY PAGES WITH OVERLAPPING BCMS			
MAP	P10	P11	P32	P33
XXXX	1	1	1	1
YYYY	1	1	1	1

The table lists the common memory pages with at least one overlapping bcm as found by each processor. These pages are called bcm pages. An additional character may be printed:

B- bad bcm addresses. A bcm with address bits 11/12 not both 0 was found.

TABLE 10

BCMS OVERLAPPING PAGE: XXXX

ADDR	P10	P11	P32	P33
YYYY	1	1	1	1
ZZZZ	1	1	1	1

Relative addresses of bcms overlapping a memory page (relative to the page starting address) as found by each processor are printed.

MEMORY BUS: XXXX .

QUIT TIME TTT

This message precedes several tables associated with memory bus XXXX. TTT is the (decimal) quit time, in micro-seconds, on the memory bus, as measured by the master. If TTT is out of the range 3..7, then the character "#" is printed at the end of the line.

TABLE 11

MEMORY BUS: XXXX

PAGE TABLE

AP	P10	P11	P32	P33
YYYY	1	1	1	1
ZZZZ	1	1	1	1
PWRED OFF?	T	T	T	T

The table contains the common memory pages belonging to memory bus XXXX, as found by each processor. The information is available only if memory busses were powered off or I/O busses were powered off in case of a combined M/I bus.

The last line in the printed table indicates if memory busses were powered off (T=Y), or not (T=N). If at least one of the printed entries is the character "N" then the line is marked by the

character "#" at its end.

TABLE 12

	MEMORY BUS: XXXX			
	NUMBER OF BCM PAGES			
	P10	P11	P32	P33
	Y	Y	Y	Y
PWRED OFF?	T	T	T	T

The table contains the number of memory pages with overlapping bcms, on memory bus XXXX, as seen by each processor. Y can be:

0- no bcm pages.

1- one bcm page.

- more than one bcm pages were found.

Additional characters may be printed:

A- bcms are not on correct bus, something wrong with the bcms' addresses.

B- bad bcms' addresses. When the memory bus has been powered off no bcm page disappeared but changes in bcms were detected. In this case the changed bcms are printed in table 13.

S- strange behavior, probably memory busses overflow (more than 4 memory busses).

The information in this table is available only if memory busses were powered off, or I/O busses were powered off in case of a combined M/I bus.

The last line in the printed table indicates if memory busses were powered off (T=Y), or not (T=N). If at least one of the printed entries is the

character "N" then the line is marked by the character "#" at its end.

TABLE 13

MEMORY BUS: XXXX

BCM ADDRESSES

ADDR	P10	P11	P32	P33
YYYY	1	1	1	1
ZZZZ	1	1	1	1
PWRED OFF?	T	T	T	T

The table contains relative addresses of bcms overlapping one of the bcm pages of memory bus XXXX, as found by each processor. The addresses are relative to the page starting address.

information is available only if memory busses were powered off, or I/O busses were powered off in case of a combined M/I bus.

last line in the printed table indicates if memory busses were powered off (T=Y), or not (T=N). If at least one of the printed entries is the character "N" then the line is marked by the character "#" at its end.

TABLE 14

MEMORY BUS: XXXX

COMBINED WITH I/O BUSSES

I/O BUS	P10	P11	P32	P33
YYYY	1	1	1	1
PWRED OFF?	T	T	T	T

The table contains the list of I/O busses combined with memory bus XXXX, as found by each processor.
The information is available only if memory busses and I/O busses were powered off, or only I/O busses were powered off in case of combined M/I bus.

The last line in the printed table indicates if I/O busses were powered off (T=Y), or not (T=N). If at least one of the printed entries is the character "N" then the line is marked by the character "#" at its end.

TABLE 15

MEMORY BUS: XXXX

CONNECTED TO I/O BUSSES (I/O TO MEM TRANSFER)

BUS G/P/B	P10	P11	P32	P33
YYYY GOOD	ADD	ADD	ADD	ADD
YYYY POOR	ADD	ADD	ADD	ADD
YYYY BAD	Z	Z	Z	Z
PWRED OFF?	T	T	T	T

The table contains a list of I/O buses connected to memory bus XXXX as seen by each processor. Up to three lines may be printed for each I/O bus, as printed above. The information is obtained by trying to transfer data from the memory bus to itself by using devices on each I/O bus. For each of the used I/O devices, data is transferred from each found page on the memory bus to itself, and the results obtained for each page are compared (for each device separately).

For each possible connection the process is terminated if a good connection is found (the device finished the transfer, all its error bits were off, and the correct data was found in memory), or there are no more devices to try.

Poor connection describes the case in which the device finished the transfer, all its error bits were off but the expected data was not found in memory. The address of the first good device, and the last poor device found in this process are printed in the processors fields. Only the 3 least significant hex digits are printed. The most significant digit is identical to that of the I/O bus (YYYY).

In the BAD entry, Z can be either "M", or "D":

M- means that the processor (and not the device) got a quit from memory (from the destination area in the transfer), while trying to refer to the destination area.

D- means that while transferring data by one of the devices on the indicated I/O bus, not the same results were obtained for all pages on this memory bus (e.g. for one page the transfer was GOOD, and for another page the transfer was POOR, or didn't terminate at all, etc.) .

The last line in the printed table indicates if memory busses were powered off (T=Y), or not (T=N). If at least one of the printed entries is the character "N" then the line is marked by the character "#" at its end.

TABLE 16

	I/O BUSSES			
ADDR	P10	P11	P32	P33
XXXX	1	1	1	1
QUIT TIME	TTT			
YYYY	1	1	1	1
QUIT TIME	TTT			

The table contains a list of the I/O busses found by each processor. TTT is the (decimal) quit time, in micro-seconds, on the above I/O bus, as measured by the master. If TTT is out of the range 30..70, then the character "#" is printed at the end of the line.

Additional characters may be printed:

L- devices in low addresses (bcms' area).

N- devices were found in some quarter of the primary I/O space, but no bcm was found in the bcms' area of that quarter.

TABLE 17

	I/O BUSSES BCMS			
ADDR	P10	P11	P32	P33
XXXX	1	1	1	1
YYYY	1	1	1	1
PWRED OFF?	T	T	T	T

The table contains a list of I/O busses bcms as found by each processor. An additional character may be printed:

A- bad address. When I/O bus in which the bcm should have been plugged according to the bcm address was powered off, the bcm didn't disappear, or the bcm disappeared when another I/O bus was powered off. The last line in the printed table indicates if I/O busses were powered off (T=Y), or not (T=N). If at least one of the printed entries is the character "N" then the line is marked by the character "#" at its end.

TABLE 18

		I/O DEVICES				
ADDR NUM	TYPE TPID	NAME RPID	P10	P11	P32	P33
XXXX ZZ	YY UU	NNN VV	1	1	1	1
PWRED OFF?			T	T	T	T
SECONDARY DEV.			S	S	S	S

The table contains the common I/O devices found by each processor. Two lines are printed for each device. ADDR is the device address. TYPE is the device type. If the master can't read the device type during the printing stage the character "?" is printed. NAME is a 3 characters mnemonic name.

If the device name is not known to the printing program the word NEW is printed in the NAME field. If the type is 0, the character "-" is printed in the NAME field. The second line printed for each device contains information for DMA devices (and for devices of type SLI), or the character strings "--" for other devices. NUM denotes the contents of byte 2 in the device registers block which is normally the device number, or the code in case of a device of type SLI.

For a DMA device, TPID and RPID denote the transmit PID and the receive PID respectively. For a device of type SLI, TPID denotes the type of the second device and RPID denotes its code (there are two devices per each 16 bytes block in case of type SLI).

The last (or one before the last) line in the printed table indicates if I/O busses were powered off (T=Y), or not (T=N). If at least one of the printed entries is the character "N" then the line is marked by the character "#" at its end.

The last line in the table (SECONDARY DEV. ...) is printed only if devices were found on secondary I/O busses (^hc000-^hdfff).

Additional characters may be printed:

A- bad address. when I/O busses were powered off the device didn't behave properly (concerning its disappearance).

M- one of the PID levels appears in the status word of more than one device on the same I/O bus.

Q- the processor got a quit from one of the device registers while trying to transfer data using this device.

S- the processor found devices on secondary I/O busses.

TABLE 19

	PIDS			
ADDR	P10	P11	P32	P33
XXXX	1	1	1	1
YYYY	1	1	1	1
PWRED OFF?	T	T	T	T

The table contains the list of PIDS found by each processor. An additional character may be printed:

A- bad address. When I/O busses were powered off the PID didn't behave properly (concerning its disappearance).

The last line in the printed table indicates if I/O busses were powered off (T=Y), or not (T=N). If at least one of the printed entries is the character "N" then the line is marked by the character "#" at its end.

TABLE 20

	RTCS			
ADDR	P10	P11	P32	P33
XXXX	1	1	1	1
YYYY	1	1	1	1
PWRED OFF?	T	T	T	T

The table contains the list of RTCS found by each processor. An additional character may be printed:

A- bad address. when I/O busses were powered off the RTC didn't behave properly (concerning its disappearance).

The last line in the printed table indicates if I/O busses were powered off (T=Y), or not (T=N). If at least one of the printed entries is the character "N" then the line is marked by the character "#" at its end.

TABLE 21

RTCS' REGISTERS

ADDR	SPID	FPID	REGA	REGC	REGE
XXXX	SS	FF	AAAA	CCCC	EEEE

The table contains the contents of RTCS' registers as found by the master while printing the table. ADDR- is the RTC's address (the rightmost digit is 6) SPID and FPID- are the slow PID level and the fast PID level respectively.

REGA, REGC, and REGE- are the switch registers whose addresses are ^ha, ^hc, and ^he relative to the beginning of the 16 bytes block associated with the RTC and the corresponding PID. The character "?" is printed in columns for which the master cannot read the information while printing.

7 KGBTST

7.1 Loading and Starting the Program

A. Load a DDT starting at 2000 (hex) (e.g., DDT-12). Make the following patches if you are using DDT-12.328 (all patches are in hex): change the contents of location 209A to FA00, 209C to FA06, and 209E to FA08.

B. Press RUN to start DDT. DDT should print a message on the terminal. If you get an ILOPR, press RESET, then ATTN. DDT will print the message again, this time without the ILOPR. It is helpful to type ^H so DDT will print the locations in hex.

C. Load KGBTST and make the following patches: change the contents of location 0172 to FA00, 0174 to FA06, and 0176 to FA08.

D. Use DDT to start the program at 100 hex.

Pressing RESET, then ATTN restarts DDT, which may be used to change program parameters and start the various subprograms. Data errors cause the printout of the error history table. Other errors result in ILOPRs which are fielded by DDT, and the KGBTST listing should be consulted for comments explaining the errors.

If in the course of checkout you change a location in the "pure code" portion of the program, the program, upon restarting, reminds you of that fact by halting with all "ones" in both the ADDR and DATA lights. Press RUN to continue.

7.2 Parameters

Several operating mode parameters are available. To prevent the program from using the console ADDR lights (so you may use them for Address-Halt or other manual functions) change NOLITE to non-zero.

To tally errors but continue rather than halting, make CONTIN non-zero. To change the length of the data table, change TABSIZ (800 hex) to the number of bytes to be used. Hence the table may be extended to the top of memory, to include the DDT code and any random data lying about. Several tapes (<PLUDIAG>P63B and P511B) are available to provide the repeating pseudo-random patterns used by the modem industry.

•

7.3 Indications and Errors

The word read from the DATA register is displayed in the DATA lights. Bits 0-7 contain the data byte, and will normally flash rapidly. Bits 14 and 15 will glow dimly, as explained in the Functional Spec.

The rightmost byte of ADDR counts the number of times the data table has been transmitted. The leftmost byte contains the error tally.

7.4 Algorithm

Initialization sumchecks the code, finds the card address, zeros the error history table, and sets up the various interrupt vectors. Status/control bits are tested, and the durations of several oneshots are checked. The table-driven loopback test begins by transmitting a header that tests the character-sync logic and then repeatedly transmits the data table, checking each received character against the same table. Errors cause a table starting at 140 hex to be filled, ordinarily providing enough pre- and post-error information to allow reconstruction of events.

Independent subprograms test the address decoding logic thoroughly, provide a crude test of the Red-to-Black path (PLIOPT is much better), exercise the status bits, and provide a brute force loop test

that should allow some transfer of data even if the card is very sick.

7.5 Checkout of an Untested KGB

7.5.1 Pre Power

A. Visual Inspection

(1) Where sockets are used, check that bug fingers are properly inserted.
(2) Check that correct bugs are inserted by using KGBPX printout or KGB-00.
(3) Check wiring of the .1 uf redcaps to the left of the 40 pin USRT. USRT pin 1 gets its +5V from one of them, but the cap frequently fails to be wired to the +5V bus.
(4) Check for crud on the infibus connector and scrub with a pink-pearl eraser when in doubt. Use compressed air to remove the crumbs - they are as bad as the crud.
(5) Check values of all discrete components against KGB-10, particularly zener polarities.
(6) Typical places for extra wires are oneshot capacitors that have been wired as if they were bypass caps, and screwed up ground wires on the switch bugs.

7.5.2 Power Shorts

(1) Ohmeter power leads (+5, +15, -15) to ground and to each other. High resistance should be observed with one meter polarity, and no less than about 25 ohms with the other polarity. 6-ohms and 30-ohms are typical from +5 to ground.

7.5.3 Power Test

Use a bus with an F-stuck PID whose bottom switch is in the left (off) position, (or substitute the prescribed 1K jumper bug in A2 (see KGB-10).

A. Power on and check USRT Pin 1 = +5V, pin 6 = -12V, and pin 4 = gnd.

•

7.5.4 Standalone Program Tests

All addresses below are in Hex. Read in a DDT that starts at 2000 (e.g., DDT-12xxx or -15xxx,) and KGBTST.BIN. Get listing for the latter. Jumps to starting points, and locations to be changed during testing, are found on the first page of the listing. Most errors found result in an ILOPR, with DDT fielding them and pointing to a comment in the listing that suggests what is wrong. When required, DDT may be restarted with RESET + ATTEN.

When the program is first read in it calculates a sumcheck for the "pure code" portion of the program. Thereafter each time it is run it checks the code against this checksum and, if something has been changed, HALTs with all console lights on as a reminder to the operator. Push RUN to continue. In this way you will probably be prevented from testing a new card with a program you may have modified to diagnose troubles found with a previous card. Read in the tape again or unmodify whatever you changed.

A. Address Decoding

Start subprogram TFIND (104 presently). Change each of the address switches while watching the ADDR lights for confirmation that the switches work. Avoid E000, FA00, FC30, and other addresses of cards already present on the bus.

Change the leftmost four switches on A4. DATA lights 0-3 should follow, but be inverted in sense.

B. Status and Control Bits

(1) Start subprogram T3A (presently 110). The program allows you to change bits in the console ADDR register and stores them in the KGB status/control register. The KGB s/c register is read every 1/3 sec and its contents stored in the DATA lights. Every other pass, just before reading s/c, the ADDR register is read and stored in s/c. This allows you to observe the effect of those bits that time out and reset themselves. The Black-to-Red word is also written to test its Busy bit (ST-5).

In general, the DATA lights will follow the ADDR buttons, except as follows: Bits 4 and 15 should remain ON. Bits 0, 3, 9, and 11-13 should remain OFF. Bits 6 and 7 will be OFF until you force high pins 8 or 7 respectively of the bottom (26-pin) edge-card connector with a cliplead. "VOLT3+" is a suggested source of "high" for the cliplead. Bit 1 will flash at about once per second if ADDR-1 is on, and ADDR-5 affects Bit-5 similarly. Bit 14 may be on or off. Bit 8 should be turned ON by ADDR-8 and off by ADDR-9 except that if both are ON it is complemented "at half rate". ADDR-10 and -11 affect DATA-10 the same way. See the Functional Specs and/or logic drawings.

C. Red to Black Path (Optical Isolators), quick test

PLIOPT should be used to test the optical isolator logic. Only if trouble develops, revert to this part of KGBTST. Start subprogram T38 (presently 108). At 1/4 sec intervals it alternately writes ones and zeros into all B-R bits but allows you to mask off bits individually with the console DATA lights. The outputs of the optical isolator drivers may be observed with a scope. Alternately, a second bus containing a KGR, and running the T3B portion of KGRTST may be used (with a set of optical isolators to connect KGB to KGR) to display the

data in the Red Data lights. This also allows you to observe signals through the whole path, using controlled data patterns. Turn Black DATA bit 10 OFF so the program will stop reseting Red's bus. Restart T3B in Red. Turning off bits 0-9 of Black's DATA lights should then have an identical effect on Red.

Black Data bit 11 is ignored - the corresponding optical isolator signal is complemented (this is Status/Control word bit TGCK) each pass through. The program will ILOPR if the 3 mS "buffer busy" oneshot is out of tolerance by more than 25%.

D. Simple Data Check, looped.

T3C, T3C7, and T3C7B are three startpoints for a simple test that transmits data extracted from the console ADDR lights, loops it, and displays the word received - in the DATA lights. Various malfunctions can prevent ANY data from being received, and the three startpoints provide ways of making successively fewer demands on working logic. The test need not be run at all if the continuous go/nogo 100 hex) test works.

(1) Read the comments that precede the three startpoints in the listing. Try each until one is found which gets through to T3D. This is signaled by the right byte of the DATA lights going on.

(2) Change bits 0-7 of the ADDR lights. The DATA lights should follow, with bit 15 also set. The program is continually reading the ADDR lights, transmitting the low order byte while looped, and storing the character read back into the DATA lights. It is useful for tracking down gross failures in the data paths.

E. Complete Data Check, looped.

(1) Start KGBTST at 100, which is also the power-on and paper tape jump startpoint. This part of the program is described in detail on pages 1 and 2. If the flashing of the lights doesn't look random enough, read in P511B.BIN which is a pseudo random modem-industry test

pattern. Failures save the evidence before halting at an ILOPR. The listing tells what to print out and how to interpret it.

F. External Loop Test.

See comments that precede T3E in the listing. Phasing of input and output clocks is unrelated but has a very low probability of being so far out that this test will not work. Input and output waveforms, however, will not be realistic because the load terminations provided in a KG installation will not be present.

7.5.5 Tests with KGR card

A full check can only be made with a second independent bus containing a KGR, and a third independent bus containing a KG-Simmulator (SLI #9, modified). The tests used are only listed here but are written up in detail in their own -.DOC files.

A. Black-to-Red Path Test with PLIOPT.

B. Half-Duplex with KG Simulator Test with PLITST. Run KGB clock at 50 and 80 KB.

C. Operational PLI program with simulator Test with <PLI>RPLI. Run KGB clock at 50 and 80 KB. See details in PLI Maintenance Guide.

D. Half-Duplex with Real KG ... as in B. Run KGB clock at 50 and at least 133 KB, depending on the individual site.

E. Operational PLI program with Real KG... as in C. Run KGB clock at 50 and at least 133 KB, depending on the individual site. Check waveforms listed in part 4.10 of PLI Maintenance Guide.

KGB INITIAL TEST CHECKLIST
KGB Ser. No._____

80

F

Init's

hotrm

and Duratn	if used	Para.	Test	Date	Done	Sgstd	Test	KGR Duratn
I,A			Visual			x	x	x
____	x							
,B			Ohmeter			x	x	x
____	x	II,A	Voltage				x	x
x	____	x						
III,A			Address Decoding			twice	x	x
____	x							
,B			Stat/Cntrl word bits			3 mins	____	x
____	x							
,C			Simple B/R Functions			twice	x	x
____	x							
,D			Simple SYN Test			3 mins	____	x
____	x							
,E			Final Standalone Test			8 hrs	____	x
____	____							
IV,A			Bl/Rd Path (PLIOPT)			8 hrs		____
____	____	____						
,B			HalfDup w/ KG Sim (PLITST)			* 1 hr		____
____	____	____						
,C			Operational Pgm w/ Sim.			* ovrnite		____
____	____	____						
,D			HalfDup w/ KG (PLITST)			* 1 hr		____
____	____	____						
,E			Operational Pgm w/ KG			* ovrnite		____
____	____	____						

* NOTE: Test at both high and low extremes of the KG clock rate for the system being used.

Comments, Troubles found:

8 KGRTST

8.1 Loading and Starting The Program

A. Load a DDT starting at 2000 (hex) (e.g., DDT-12). Make the following patches if you are using DDT-12.328 (all patches are in hex): change the contents of location 209A to FA00, 209C to FA06, and 209E to FA08.

B. Press RUN to start DDT. DDT should print a message on the terminal. If you get an ILOPR, press RESET, then ATTN. DDT will print the message again, this time without the ILOPR. It is helpful to type ^H so DDT will print the locations in hex.

C. Load KGBTST and make the following patches: change the contents of location 0170 to FA00, 0172 to FA06, and 0174 to FA08.

D. Use DDT to start the program at 100 hex.

Pressing RESET, then ATTN restarts DDT, which may be used to change program parameters and start the various subprograms. Data errors cause the printout of the error history table. Other errors result in ILOPRs which are fielded by DDT, and the KGBTST listing should be consulted for comments explaining the errors.

If in the course of checkout you change a location in the "pure code" portion of the program, the program, upon restarting, reminds you of that fact by halting with all "ones" in both the ADDR and DATA lights. Press RUN to continue.

8.2 Parameters

Several operating mode parameters are available. To prevent the program from using the console ADDR lights (so you may use them for Address-Halt or other manual functions) change NOLITE to non-zero. To tally errors but continue rather than halting, make

CONTIN non-zero. To change the length of the data table, change TABSIZ (800 hex) to the number of bytes to be used. Hence the table may be extended to the top of memory, to include the DDT code and any random data lying about. Several tapes (<PLUDIAG>P63B and P511B) are available to provide the repeating pseudo-random patterns used by the modem industry.

8.3 Indications and Errors

The word read from the DATA register is displayed in the DATA lights. Bits 0-7 contain the data byte, and will normally flash rapidly. Bits 14 and 15 will glow dimly, as explained in the Functional Spec.

The rightmost byte of ADDR counts the number of times the data table has been transmitted. The leftmost byte contains the error tally.

8.4 Algorithm

Initialization sumchecks the code, finds the card address, zeros the error history table, and sets up the various interrupt vectors. Status/control bits are tested, and the durations of several oneshots are checked. The table-driven loopback test begins by transmitting a header that tests the character-sync logic and then repeatedly transmits the data table, checking each received character against the same table. Errors cause a table starting at 140 hex to be filled, ordinarily providing enough pre- and post-error information to allow reconstruction of events.

Independent subprograms test the address decoding logic thoroughly, provide a crude test of the Red-to-Black path (PLIOPT is much better), exercise the status bits, and provide a brute force loop test that should allow some transfer of data even if the card is very sick.

8.5 Checkout Of An Untested KGR

8.5.1 Pre Power

A. Visual Inspection
   (1) Where sockets are used, check that bug fingers are properly inserted.
   (2) Check that correct bugs are inserted by using KGRPX printout or KGR-00.
   (3) Check wiring of the .1 uf redcaps to the left of the 40 pin USRT. USRT pin 1 gets its +5V from one of them, but the cap frequently fails to be wired to the +5V bus.
   (4) Check for crud on the infibus connector and scrub with a pink-pearl eraser when in doubt. Use compressed air to remove the crumbs - they are as bad as the crud.
   (5) Check values of all discrete components against KGR-10, particularly zener polarities.
   (6) Typical places for extra wires are oneshot capacitors that have been wired as if they were bypass caps, and screwed up ground wires on the switch bugs.

B. Power Shorts

Ohmeter power leads (+5, +15, -15) to ground and to each other. High resistance should be observed with one meter polarity, and no less than about 25 ohms with the other polarity. 6-ohms and 30-ohms are typical from +5 to ground.

8.5.2 Power Test

Use a bus with an F-stuck PID whose bottom switch is in the left (off) position, (or substitute the prescribed 1K jumper bug in A2 (see KGR-10).

A. Power on and check USRT Pin 1 = +5V, pin 6 = -12V, and pin 4 = gnd.

8.5.3 Standalone Program Tests

All addresses below are in Hex. Read in a DDT that starts at 2000 (e.g., DDT-12xxx or -15xxx,) and KGRTST.BIN. Get listing for the latter. Jumps to starting points, and locations to be changed during testing, are found on the first page of the listing. Most errors found result in an ILOPR, with DDT fielding them and pointing to a comment in the listing that suggests what is wrong. When required, DDT may be restarted with RESET + ATTEN.

When the program is first read in it calculates a sumcheck for the "pure code" portion of the program. Thereafter each time it is run it checks the code against this checksum and, if something has been changed, HALTs with all console lights on as a reminder to the operator. Push RUN to continue. In this way you will probably be prevented from testing a new card with a program you may have modified to diagnose troubles found with a previous card. Read in the tape again or unmodify whatever you changed.

A. Address Decoding

Start subprogram TFIND (10C presently). Change each of the address switches while watching the ADDR lights for confirmation that the switches work. Avoid E000, FA00, FC30, and other addresses of cards already present on the bus.

Change the leftmost four switches on A4. DATA lights 0-3 should follow, but be inverted in sense.

B. Status and Control Bits

(1) Start subprogram T3A (presently 104). The program periodically changes the state of bits 2 and 3 and reads them back to the ADDR lights. Bit 2 should wink at half the rate of Bit 3. Bits 4-11 should be off, bits 12 and 13 should be on, and bits 0, 1, 4, and 15 may be on or off. Disregard the data lights.

(2) With DDT change DLY1 to 500 decimal ("500."). Observe TPREP- with a scope (KGR-23, pin 24 of upper edge connector): it should have a 6ms pulse falling from +3V to -1V. Change DLY1 back.

C. Static Test, Black-to-Red Path (Optical

Isolators)

Run subprogram t3B (presently 104), starting at 104, and use a groundlead to touch the specified pins of the upper edge connectors, watching which ADDR lights are turned off. Grounding pin 14 should turn off DATA light 13. Grounding pin 13 should turn off light 12. The remaining bits are better tested with PLIOPT (see section IV,A). (KGRTST can also be useful when PLIOPT won't run in the Red side. It doesn't check the data; it only displays it, so stuck bits or a failure to read the BK2RD word will be fairly obvious).

D. Simple Data Check, looped.

(1) Start subprogram T3C (at 108). The first part of the test checks the character-sync logic of the card. Completion of this is signalled by the right byte of the DATA lights going on.

(2) Change bits 0-7 of the ADDR lights. The DATA lights should follow, with bit 15 also set. The program is continually reading the ADDR lights, transmitting the low order byte while looped, and storing the character read back into the DATA lights. It is useful for tracking down gross failures in the data paths.

E. Complete Data Check, looped.

Start KGBTST at 100, which is also the power-on and paper tape jump startpoint. This part of the program is described in detail on pages 1 and 2. If the flashing of the lights doesn't look random enough, read in P511B.BIN which is a pseudo random modem-industry test pattern. Failures save the evidence before halting at an ILOPR. The listing tells what to print out and how to interpret it.

8.5.4 Tests with KGB card

A full check can only be made with a second independent bus containing a KGB, and a third independent bus containing a KG-Simmulator (SLI #9, modified). The tests used are only listed here but are written up in detail in their own -.DOC files.

A. Black-to-Red Path. Test with PLIOPT.

B. Half-Duplex with KG Simulator. Test with PLITST. Run KGB clock at 50 and 80 KB.

C. Operational PLI program with simulator. Test with <PLI>RPLI. Run KGB clock at 50 and 80 KB. See details in PLI Maintenance Guide.

D. Half-Duplex with Real KG As in B. Run KGB clock at 50 and at least 133 KB, depending on the individual site.

E. Operational PLI program with Real KG As in C. Run KGB clock at 50 and at least 133 KB, depending on the individual site. Check waveforms listed in part 4.10 of PLI Maintenance Guide.

8.6 KGR Initial Test Checklist

KGB

Ser. No._____

80

F

Init's

hotrm

KGB Duratn	and Duratn	if used	Date	Done	Sgstd Para.	Test Test
I,A	Visual				x	x
x	____	x				
,B	Ohmeter				x	x
x	____	x				
II,A	Voltage				x	x
x	____	x				
III,A	Address Decoding				twice	x
x	____	x				
,B	Stat/Cntrl word bits				3 mins	____
x	____	x				
,C	Simple B/R Functions				twice	x
x	____	x				
,D	Simple SYN Test				3 mins	____
x	____	x				
,E	Final Standalone Test				8 hrs	____
x	____	____				
IV,A	Bl/Rd Path (PLIOPT)				8 hrs	____
____	____	____				
,B	HalfDup w/ KG Sim (PLITST) *				1 hr	____
____	____	____				
,C	Operational Pgm w/ Sim. *				ovrnite	____
____	____	____				
,D	HalfDup w/ KG (PLITST) *				1 hr	____
____	____	____				
,E	Operational Pgm w/ KG *				ovrnite	____
____	____	____				

* NOTE: Test at both high and low extremes of the

KG clock rate for the system being used.

Comments, Troubles found:

9 MEMCHK

Before loading the program all busses should have power turned on. Power restart interrupts should be enabled (S71 in the console in the middle position which enables the power recovery int). The bus reset timers of all busses should be in the OFF (up) position.

9.1 Parameters

Location(Hex)	Default Value(Hex)	Function
104	FA00	TTY
100		RESTART PROGRAM

First PSTOP must be used to halt all processors. Then DDT may be loaded to common memory if the operator wants to use it later (the MEMCHK program doesn't need DDT for its operation).

Then the program is loaded and starts automatically. To restart the program from DDT or from the console, the address ^h100 should be used as per the the Parameters. RESET-ATTENTION can be used to restart the program only during the initial interaction phase, later it has another meaning. One can also restart the conversation by replying "E" to one of the program questions during the conversation. The special character "R" can be used for achieving the same effect during the test itself.

A. Load correct DEVTST tape or cassette according to system configuration.

B. Press "Reset" on control panel.

C. Set start address in data lights on control panel if loading from cassette.

D. Press "Load" on control panel. Check the console to see if all processors and common memory are listed as per the SRN. If not, you are not running the correct version. If no display on console, check to see if your console is at the address the diagnostic expects to see it (Refer to Parameters section).

E. Verify next on the screen is start of questions.

9.2 The Initial Interaction With the User

The program starts by interacting with the user, which has to define the program's mode of operation. The answers to the questions are:

Y- yes, N- no, or E- exit.

"E" means exit and restart the program at the beginning of the conversation.

Each of the above characters must be followed by a carriage-return. If an illegal pair of characters is typed by the user, the program prints the string "? (Y/CR,N,E)" and waits for a legal response.

The program interprets a single carriage-return as the character "Y" followed by a carriage-return.

The messages and questions printed by the program are described next.

1. MEMCHK-MB or MEMCHK-SB

The program name is the first message printed by the program.

2. THE FOLLOWING QUESTION WILL BE ASKED ONLY ONCE. BE CAREFUL !!!!

This message is printed before question 3 is printed.

3. DO BATTERY TESTS? (Y/CR,N,E)

Answer yes to detour through the battery check code, which is documented starting with step number 27 below, say no to continue from here with memcheck.

4. ALL I/O BUSSES? (Y/CR,N,E)

The user has to specify whether he wants remote memories to be tested using BBC through all existing I/O busses or through some partial list of the existing I/O busses.

If the answer is "Y" then all I/O busses are found by the program. I/O busses are found by looking for devices at the following locations: E000, E800, F000, F800. The program then continues in step 6.

If the answer is "N", then the user has to specify up to four I/O busses for use in testing remote pages in the following step.

5. ADDRESS=

The user has to specify an I/O bus starting address (4 hex digits e.g., E000, E800, etc.). Up to four I/O busses can be specified.

As in all the cases in which the user has to specify some numbers to the program, the above message is preceded by the question:

MORE? (Y/CR,N,E)

to which the user should answer positively if more entries should be entered to the currently discussed table, or negatively if no more entries are needed.

Each number specified by the user is followed by the message:

CONFIRM? (Y/CR,N,E)

to which the user has to respond in the implied way.

6. I/O BUSSES:

At this step the existing selected I/O busses for use in testing remote pages are printed by the program.

7. RUN ALL PROC? (Y/CR,N,E)

The user has to answer whether he wants all processors to run the tests, or only selected ones.

If the answer is "Y", then all existing processors are found by the program, and the program continues in step 9.

If the answer is "N", then the user has to specify up to 16 processors to run the tests in the following step.

8. PROC=

Following this message the user has to type a 2 digit processor number (e.g. 12 or 13). This number is the hexadecimal offset of the processor BCM address on one of the I/O busses. Up to 16 processors can be specified.

For a single bus system, the numbers 00, and 01 are associated with the main processor (the MASTER), and its buddy (the BUDDY), respectively. In this case up to two processors can be specified in this step.

As in all the cases in which the user has to specify some numbers to the program, the above message is preceded by the question:

MORE? (Y/CR,N,E)

to which the user should answer positively if more entries should be entered to the currently discussed table, or negatively if no more entries are needed. Each number specified by the user is followed by the message:

CONFIRM? (Y/CR,N,E)

to which the user has to respond in the implied way.

9. PROCESSORS:

At this step the program prints a list of all selected processors that will participate in the tests. A processor that the program can't halt at this step is removed from the list, and the word BAD is printed after its 2 digit number.

10. CHK ALL PROC MEM? (Y/CR,N,E)

The user has to specify whether he wants all processor memories to be tested, or only some of them.

If the answer is "Y", then all processor memories are tested, and the program continues in step 12.

If the answer is "N", then the user has to specify the numbers of all processors whose local memories should be tested. Up to 16 processors can be selected. Note, that this list need not be identical to the list of processors running the tests. The selection of the processor memories is done in the next step.

11. PROC=

The user has to react exactly as in step 8 above. Note that all pages of a selected processor are tested. For a multi-bus system the program searches up to two pages, and for a single-bus system up tp 6 pages per processor.

12. PROC MEM'S:

At this step, the list of processor memories is printed. The program tries to halt each processor whose local memory should be tested. If this fails, the word BAD is printed after the processor number, otherwise, all found pages of this processor are printed after the processor number.

13. USE ALL MAPS? (Y/CR,N,E)

The user has to specify whether he wants common memory pages to be tested using all map registers, or only some of them.

If the answer is "Y", then all maps are used and the program skips the following step.

If the answer is "N", then the user has to specify up tp four map registers in the next step.

14. MAPR= (0-3)

Following this message the user has to specify a map register by typing a single digit (0-3). 0- stands for map 0 (FC00), 1- for map 1 (FC02), etc.

As in all the cases in which the user has to specify some numbers to the program, the above message is preceded by the question:

MORE? (Y/CR,N,E)

to which the user should answer positively if more entries should be entered to the currently discussed table, or negatively if no more entries are needed.

Each number specified by the user is followed by the message:
CONFIRM? (Y/CR,N,E) to which the user has to respond in the implied way.

15. MAPS:

At this step, the program prints the addresses, of the selected map registers.

16. CHK ALL COMN MEM? (Y/CR,N,E)

The user has to specify whether he wants all pages of common memory to be tested, or only some of them.

If the answer is "Y", then all common memory pages except those specified in step 19, are tested, and the program continues in step 18.

If the answer is "N", then the user has to specify the pages to be tested in the following step.

17. MAP=

Following this message the user has to type a four hexadecimal digits number, specifying the map address of a page in common memory (e.g. 0000, 0200, 4000, etc.)

For convenience to the user, the escape character can be used in this step for generating successive page map addresses. The escape character should be typed once, instead of a four digits number, and it increments a built-in program counter. This counter is initialized to 0000, and is set to a new value whenever the user types an explicit four digits number.

As in all the cases in which the user has to specify some numbers to the program, the above message is preceded by the question:

MORE? (Y/CR,N,E)

to which the user should answer positively if more entries should be entered to the currently discussed table, or negatively if no more entries are needed.

Each number specified by the user is followed by the message:

CONFIRM? (Y/CR,N,E)

to which the user has to respond in the implied way. When the escape character is used, it is not followed by the "confirm" message.

18. IGNORE COMN MEM PAGES? (Y/CR,N,E)

The user has to specify whether he wants some of the common memory pages specified in the previous steps to be ignored (i.e., not to be tested), or not.

If the answer is "N", then all selected pages will be tested, and the program continues in step 20.

If the answer is "Y", the user has to specify which common memory pages are to be ignored, in the following step.

19. MAP=

The user has to react to this message exactly as described in step 17. The escape character can be used in this step also.

20. COMN MEM PAGES:

In this step the program prints all ^&existing selected common memory pages which are not included in the list of pages to be ignored. Selected pages which are not found at this step are ignored for the rest of the test.

21. RUN ALL TESTS? (Y/CR,N,E)

The user has to answer whether he wants to run all tests, or only some of them.

If the answer is "Y", then all tests are selected and the program continues in step 23.

If the answer is "N", then the user has to select tests from the available list of 11 tests in the following step.

22. SELECT- (Y/CR,N,E)

At this step the user has to select tests by typing "Y" (or equivalently carriage-return), or "N" after each test name (printed by the program). The tests names and meanings are described in a later section.

23. SELECTED TESTS:

At this step, the list of selected tests is printed.

24. FIND MULTIPLY ADDRESSED PAGES? (Y/CR,N,E)

The question is asked only in case of a multi-bus version. If a positive reply is given, then the first test each processor performs is to find out whether multiply addressed pages exist. Information about such pages is accumulated in the TEST LOG.

Due to memory limitations, if a positive reply is given, the master pages are not tested; the user is notified about this by a short message. In order to also test the master pages, the user has to give a negative reply in this step (however, see next step).

25. ABOVE QUESTION WILL NEVER BE ASKED AGAIN
IF CONFIRMED!! CONFIRM? (Y/CR,N,E)

This question is asked only if a negative reply is given to the question in the previous step. The program requires a confirmation since if a test for finding multiply addressed common memory pages is not needed, the program modifies itself in a special way and the user will never be asked again the question in step 24 (unless the program is reloaded). It is recommended that after MEMCHK tested all pages except those of the master, the master pages will also be tested (by replying positively to this question).

26. OVER-NIGHT? (Y/CR,N,E)

The user has to reply whether the tests are to be performed over a long period of time or not. The answer has an effect only if there is no hard copy device in the system.

If there is no hard copy device, and the answer to the above question is "N", then during the tests, whenever the screen has beened (with error messages), the program waits for a character to be typed by the user (which implies that he must check the display from time to time).

If there is no hard copy device and the answer to the above question is "Y", then the program continues printing even when the screen has been filled, thus error messages may disappear. In this case however, when the TEST LOG is printed as a request of the user, the program will wait for user characters whenever the screen has beened.

After the last step, the tests are started. The MASTER activates the selected processors one at a time, each of them tests all selected pages except those of his buddy, and then reactivates the MASTER.

27. CHK ALL COM MEM? (Y/CR,N,E)

The folowing steps are the battery check interactions. Control arrives here after a yes answer at step 3.

The battery check code follows according to steps 16 thru 20 in order to establish which pages of common memory contain battery backed up memory to test. After step 20 return to the next step (28).

USE DEFAULT PATTERN (AAAA)? (Y/CR,N,E)

All selected memory pages of common memory are filled with the selected pattern for later comparison.

COMPARE CYCLE ONLY? (Y/CR,N,E)

This question allows the user to do only the compare of previously initialized memory. A no answer here will continue with a normal load, wait and compare cylce for the battery check. A yes will skip the load and jump to the "COMPARE CYCLE" of this step.

WRITE CYCLE DONE- POWER DOWN NOW!

The memory is initialized and power may now be turned off. Power up interrupts and reset-attention both will begin the program at the compare cycle to complete the test.

COMPARE CYCLE

A word by work comparison of memory with "pattern" will be done with all of memcheck's error detection code enabled. If no errors are reported the test is complete and the board(s) are good.

BATTERY TEST DONE

This message signals the end of the battery check code. Control returns to step 3 to allow repeating the test or continuing on with memcheck.

9.3 The Tests

In this section we describe the available tests. Each test will be characterized by several parameters:

NAME	The test name.
INFO	The written information.
MODE-W_(W/B)	The information is written word by word (W), or byte by byte (B).
MODE-R_(W/B)	The page is read word by word (W), or byte by byte (B).
DELAY-W_(Y/N)	Y if delays are introduced between write operations, else N.
DELAY-R_(Y/N)	Y if delays are introduced between read operations, else N.

The meaning of each test is defined in the following table:

NAME	INFO	MODE-W	MODE-R	DELAY-W	DELAY-R
WORST	note 1	W	W	Y	N
WORST CON.	note 2	W	W	N	Y
ADDRESS	note 3	B	W	N	N
ADDRESS CON.	note 4	W	W	Y	Y
RANDOM	note 5	B	B	N	N
ALTER AAAA	AAAA	W	B	N	N
ALTER 5555	5555	B	W	Y	Y
CONST 0000	0000	B	W	N	Y
CONST FFFF	FFFF	W	W	N	N
LOCKS WORDS -1	FFFF	W	W	N	N
LOCKS WORDS -2	note 6	-	W	-	Y
LOCKS BYTES -1	FFFF	W	B	N	N
LOCKS BYTES -2	note 7	-	W	-	Y

Notes:
(1) The5pattern in this test consists of 8 words 0000, 8 words FFFF, etc. In the middle of the page the order is reversed.
(2) The5complement of the pattern in (1).
(3) The5pattern is the rightmost 13 bits of the word address (0000, 0002, ... ,1FFE).
(4) The5complement of the pattern in (3).
(5) A 5pseudo-random pattern (see listing for additional information).
(6) Thi5 is the second part of the LOCKS WORDS test. Nothing is written at this test, and zeroes (0000) are expected at each location. This test is automatically selected when LOCKS WORDS -1 is selected.
(7) Thi5 is the second part of the LOCKS BYTES test. Nothing is written at this test, and zeroes (0000) are expected at each location. This test is automatically selected when LOCKS BYTES -1 is selected.

9.4 The Test Messages

During the testing period, the program prints messages on the TTY. Some of the messages are "life signs", and some are errors. The life signs are printed one over the other (in the same line) so as not to causing of the screen. The error messages are printed in separate lines. Error messages are ended by a long sequence of bell characters, to catch the attention of the user.

Each message is preceded by the processor identification as in the following line:

Pnn.<proc>:

nn- is the processor number. <proc> is the processor type, it can be one of the following: MAST (master), BUDY (the master's buddy), or RMOT (remote).

nn
ex P20 mast

The messages at this stage are:

1. Pnn.<proc>: ACTIVATING Pmm

This message is printed by the master before activating another processor, or by a slave processor before reactivating the master processor. The number of the activated processor is mm. The message is over-printed by other messages.

2. Pnn.<proc>: CAN'T ACTIVATE Pmm

The message is printed by processor nn who can't activate processor mm. The message is not over-printed by other messages.

3. Pnn.<proc>: ACTIVATED

The message is printed by processor nn upon activation by the master, and before starting the tests. The message is followed by a short sequence of bell characters on the TTY to indicate a processor switching.

This is an over-printed message in case of a non hard copy device. In case of a hard copy device, the message is not over-printed.

4. Pnn.<proc>: T=tttt <test name> aaaa/bbbb LOOPS=llll

The message is printed by processor nn before each test (approximately every 1-2 seconds). This message is overprinted by other messages, and is intended to serve as a detailed "life sign". In case of a hard copy device it is replaced by a simple carriage-return.

The meanings of the fields are:

tttt- The current test (hexadecimal) number.
llll- The (hexadecimal) number of times the program has looped through the whole system.
aaaa- The meaning depends on the mode as follows:

A- If processor nn checks one of its pages, then aaaa=00nn (e.g., 0012).
B- If processor nn checks a remote processor, then aaaa=ddmm, where dd are the most significant digits of the I/O bus starting address through its BBC the test is done, and mm is the remote processor number (here remote is relative to the testing processor- nn) (e.g. E032).

If processor nn checks a common memory page, then aaaa equals to the page map address (e.g. 0200).

bbbb- This is the address of the first word to be tested on the current page. For processor bus memories it is: 0000, or 2000 (for a single-bus system it continues up to A000), and for a common memory page it is one of: 4000, 6000, 8000, A000 depending on the map register used in this test.

5. Pnn.<proc>: T=tttt <test name> aaaa/bbbb W=wwww R=rrrr Q=q

FIRST or LAST

This is an error message printed while reading the information from the tested page. The meaning of nn, tttt, and aaaa is as in message 4.

bbbb- The address in which an error was found (same method as in message 4).
wwww- The written information.
rrrr- The read information. If 2 quits had occurred while trying to read from the above address, then rrrr=DEAD.
q- The number of quits during this read operation. It can be 0/1/2, or "B". The program does not retry reading the information after getting two quits. The character "B" stands for BBC and means that some trouble has occurred while referring to one of the BBC registers.

The line ends with one of the words: FIRST or LAST. Only the first error in a sequence of errors, and the last one (if different from the first error) are printed, and the additional word indicates if the error is the first or last in a sequence.

6. Pnn.<proc>: T=tttt <test name> aaaa/bbbb W=wwww DISAPPEARED

This error message is printed by processor nn when 2 quits had occurred while trying to ^&write information to the tested page at address bbbb. The program marks this page as if it has disappeared. It continues with the next page, but it retries testing this page when its turn comes again. However, it prints the message about its disappearance only once.

A list of pages that have disappeared is kept in the TEST LOG.

7. Pnn.<proc>: T=tttt MULTIPLY ADDRESSED PAGES FOUND

This error message is printed by processor nn when it found multiply addressed common memory pages. A list of multiply addressed common memory pages is kept in the TEST LOG.

8. Pnn.<proc>: T=tttt <test name> aaaa/bbbb W=wwww QUITS=qqqq

This error message is printed if in qqqq (where qqqq>0) words of the tested page one quit has occurred while trying to write information to the page.

9. Pnn.<proc>: BAD BBC: xxxx

This error message is printed by processor nn who failed to activate the BBC for testing a remote page. xxxx is the bad BCM address (e.g. E032). This error is ^&not marked in the TEST LOG.

10. UNEXPECTED QUIT @xxxx

The message is printed for the obvious reason. The program should be restarted and if the message is consistent then the program listing must be used to analyze the cause of the quit.

9.5 The Interaction During The Tests

During the testing period, the user may interact with the program by typing ^&special characters on the TTY.
The characters are: H, N, R, S (other characters are ignored) and their meanings are explained below.

H- Halt- The program stops the tests and waits for additional character to be typed on the TTY. If the additional character is a special character, it is interpreted appropriately, otherwise it

causes resumption of the tests. A possible use of the "H" character is: first type "H", then power OFF a memory bus, exchange a memory card, power ON the memory bus, and let the program proceed (e.g., by typing a space character).

N- Next- This character instructs the program to abort the current group of tests being performed on the current page, and to start the next group of tests. This may cause skipping several tests on the current page and starting testing it through another path (the next map register or the next I/O bus), or starting with the next page.

R- Restart- The program returns from the current processor to the master, and restart the initial interaction. The TEST LOG is cleared.

S- Summary (or Status)- This character causes printing of the TEST LOG (as described in a later section). After the printing, the program waits for a character to be typed on the TTY. If this character is a special character, it is interpreted appropriately, otherwise it causes resumption of the tests.

If an activated processor does not respond within 5 seconds from the moment it was activated by the MASTER (as can be seen on the display in case when a non hard copy device is used), or if a processor "dies" in the middle of its operation (no output on the TTY), then the user has to perform the following steps:

1. First the user has to type some character (maybe the screen has beened).

2. If it does not help, the special characters should be tried.

3. If all these fails, there is a chance that the program still exists at the master processor and RESET-ATTENTION can be used at this stage for resumming the tests. The processor which caused the troubles in this case is marked as ^&"bad" in the TEST LOG.

4. If RESET-ATTENTION does not help, then probably the program is somewhere in the system. Powering

OFF/ON a processor bus, in one of whose processors the program resides, may cause the processor to reactivate the master. The user can try this as the last resort. If this does not help the program should be reloaded and restarted.

9.6 The Test Log

The TEST LOG consists of several tables. The first table is printed in any case, while the other ones are printed only if they are not empty (i.e., if there are errors).

The tables are printed below. The ordinal numbers are not printed on the TTY.

1. LOOPS=llll TESTS=tttt
LOCAL ERRORS=iiii REMOTE ERRORS=jjjj COMN ERRORS=kkkk

The table consists of the contents of several counters. The numbers are printed in hex.

LOOPS- The total number of times the program has cycled through the whole system.

TESTS- The total number of tests performed so far.

LOCAL_ERRORS- The total number of errors found by processors in one of their pages.

REMOTE_ERRORS- The total number of errors found on processor bus memories when using BBC (i.e. from remote processors).

COMN_ERRORS- The total number of errors found in common memory pages.

2. BAD PROC'S:
Pn1 Pn2 This table lists all bad processors, i.e., processors the master processor couldn't activate, or processors that didn't reactivate the master after their activation by the master. n1, n2 are the processor numbers.

3. DISAPPEARED PROC MEM'S:
   Pn1 xxxx yyyy
   Pn2 xxxx yyyy

This table lists the processor busses's memories that have disappeared during the test (i.e., at least one word of each such page couldn't be written- caused two quits).

Pn1, Pn2, etc. are the processor's numbers. xxxx yyyy are the page numbers. For a multi-bus system they can be 0000 or 2000. For a single-bus system, they range over: 0000-A000.

4. DISAPPEARED COMN PAGES:
   xxxx yyyy

This table lists all common memory pages that have disappeared (in the sense of table 3 above). xxxx, yyyy, etc. are the pages map addresses.

5. BAD COMN PAGES:
   xxxx yyyy

This table lists all bad common memory pages, i.e. all pages that caused at list one error at some stage of the tests. xxxx, yyyy, etc. are the pages map addresses.

6. MULTIPLY ADDR. COMN PAGES:
   xxxx yyyy

This table lists all multiply addressed common memory pages. xxxx, yyyy, etc. are the pages map addresses.

7. BAD PROC MEM'S:

TEST	Pn1	Pn2	Pn3	Pn4
<test 1>	lr	lr	lr	lr
<test n>	lr	lr	lr	lr
DISAPPEARED	lr	lr	lr	lr

This table lists all processors one of whose pages caused at least one error. The processor numbers are: n1, n2, etc. The table also indicates in which tests the errors have been detected. The tests names are printed at the left side of the table. Note that only the selected tests are printed. The last line indicates whether pages of these processors have disappeared.

The character "l" can be either "L" indicating that the error has been detected locally, i.e., by the indicated processor, or "-" indicating that the error has not been detected locally.

The character "r" can be either "R" indicating that the error has been detected by some remote processor (i.e., via BBC), or "-" indicating that the error has not been detected by any remote processor.

8. COMPLAINTS ON COMN:

TEST	Pn1	Pn2	Pn3	Pn4
<test 1>	c	c	c	c
<test n>	c	c	c	c
DISAPPEARED	c	c	c	c
MULTIPLY ADDR.	c	c	c	c

This table lists all processors that have detected at least one error in one of the common memory pages. The processor numbers are: n1, n2, etc. The table also indicates in which tests the errors have been detected. The tests names are printed at the left side of the table. Note that only the selected tests are printed. The line before the last indicates whether the processors are complaining about disappearance of common memory pages. The last line indicates whether the processors are complaining about multiply addressed

common memory pages.

The character "c" can be either "C" indicating that the processor is complaining on common memory, or "-" indicating that the processor has no complaints.

Note that this table lists complaining processors, while the previous table indicates processors whose pages have caused errors to some processors.

10 MLCTST

MLCTST requires changing parameters such as lines to be tested, baud rate, etc. The parameters may be set either by the control panel or by using DDT. Refer to the section on DDT for explanation of the commands and/or the section on how to run the control panel. DDT is the best way to control/monitor processes (programs) by inspecting/changing registers of the machine in running MLCTST.

MLCTST is actually a collection of several separate programs which use a common collection of subroutines. The choice of a particular test depends on how dead the machine is and what you wish to debug. Below is a list of the diagnostics and a brief explanation of what they do:

As a rule of thumb, running Start 3 first sets up a table(starting at 1A00) that reflects the active lines on the MLC under test. After that, running Start 2 will use the entries(lines) in the table to test. It is much easier to let the software find out who is out there and build the table for you. If you want to go from there in which tests to run you can. Zeroing out lines effectively takes that line off the list to be tested.

If errors do occur they are entered in the Error table(1C00) to be analysed. Refer to the Error Type Section for the error type breakdown found in the Error table.

10.1 Operating Instructions

1- Load DDT followed by MLCTST from the operator console.
2- Enter ^H which changes the values to hex representation.
3- Enter ^K display value as constant and not as symbolic.
4- Enter: 100/ (read version no., should be 10.)
Results: 100/Pxx yy

5- If the MLC under test is not an MBB, change location 11C to 0.

6- Proceed to desired test procedure.

The following locations are the important locations that will be of interest to you:

Location	Default	Description
011C	78	MBBFLG - nz=MBB attached, z=mlc
0102	F210	DEVADR - AHL Address under test
0104	600	LINNO - Ln. No. for single line tests
0106	30	SIZE - Char. Size (see below chart)
010C	02	RATE - Baud Rate (see below chart)
0146	0	HITIME
0148	0	LOTIME
014E	0	SEC.
0150	0	MIN.
0152	0	HRS.
0B72-0B84	MLC TEST	MESAGE - Test message
1A00		RUNTAB - Start addr. of device table
1C00		ERRTAB - Start addr. of Error Table

Character Size	
0	5 bits
1	6 bits
2	7 bits
3	8 bits

Baud Rates	
0	illegal
1	75
2	110
3	134
4	150
5	300
6	600
7	1200
8	1800
9	2400

Following is a list of tests depending on what is needed to be tested. The start location is given in the heading.

10.2 START1 (200)

Its purpose is to find gross failures in the control paths of the AML and MLC. It does not do any data checking, but is mostly used as simple scope loop for getting hardware off the ground.

When the program is started it sets up line parameters for a single line to run cross-patched. Thus the operator must set the correct values for DEVADR,LINNO, and RATE. The rest of the parameters default to sensible values. After the line is set up the program forces characters over that line as fast as it can. No error checking is done. Note that the program can send data much faster than the MLC can shift it out, for low baud rates, so the MLC will flush many of the output characters. The actual data for output is held in OUTDAT which is incremented after each use.

In normal correct operation, the program should cause a characteristic behavior of the hardware. In the console address display (LOC FF80) the selected line number should be in the left byte and the received data character in the right byte. This is a display of the contents of INDAT, the input data buffer, and gives an early indication of trouble. For example, the sign bit represents the BREAK bit, which should be zero. An all zero display usually means a gross failure in the data path, such as missing or dead LIU or LIM.

The lights on the MLC display panel will normally be as follows: XPCH on, INRATE and OUTRATE as set, SIZE 8bits (3). OUTDAT and INDAT shifting, BINDAT and BOUTDAT flickering. OUTBUF should have bits 1 and 2 on ,the correct line number in the LINE NUMBER field, and the data field flickering. INBUF should have bit 1 (BREAK bit) off, the next bit (which is used as the MLC

power indicator)on, the correct line number in bits 3 to 8, and the flickering input character in bits 9 to 16. OIBUF should contain the correct line number. OTABUF should contain a 1 in bit 2, the correct line number, and a 1 in bit 16. INABUF is irrelevant, but may contain the correct line number, among other things.

10.3 START2 (204)

This program allows for much more thorough testing than the preceeding. It's purpose is to test multiple lines at user selectable rates. A fixed data message is sent out each port under test, at the specified rate, to an attached terminal, if any. The input stream is crosspatched to the output for each port, so the return data can be checked for errors. This program is useful in finding a wide range of problems, once START1 demonstrates that the basic data paths are intact. This program helps find: bit errors (pick and drops) in the data and line number field, interaction among multiple lines, particular data rate sensitivity and a variety of problems between the MLC and a terminal.

Before the program is started, the operator must set the correct AML device address at DEVADR and initialize RUNTAB. RUNTAB is a table with one entry per MLC port, which determines which MLC ports should be run and at what rate. The format of an entry in RUNTAB is: 8000 bit on, and the desired data rate in the low 4 bits. For example, to cause line 6 to be tested at 2400 baud, set 8009 into RUNTAB+C. It is most convienient to think about the RUNTAB entries in hexadecimal, and to remember that the addressing of the PLURIBUS is byte oriented. Thus the entry in RUNTAB is computed as twice the line number, converted to hex plus 1A00, which is the table start address. (Or you can count up on your fingers.)

When the program is started it has an initial phase which looks at RUNTAB and sets the appropriate hardware values for rate etc on each MLC line. After all the lines are set up, the program watches the AML

input FIFO for OI's from the lines it has set up. Unexpected OI's are recorded as errors. Expected OI's cause the next character to a line. The program maintains a separate pointer for input and for output for each line. These pointers are kept in tables which are indexed the same way as RUNTAB. RPTAB and SPTAB are the respective start address of the receive and send pointer table. Normally these are pointers into the typeout message which begins at MESAGE. ("SIMPLE MLC TEST") The message string may be modified if desired, by inserting any new string starting at MESAGE (for example, all "aa" or some non-printable characters) but be sure not to overflow into the code which follows the string storage. The string terminator is default set to zero (0000 hex). In addition to sending data characters out each line, the program periodically re-sets the output data rate to the value specified in RUNTAB. This is to imitate the behavior of the PTIP program, which has both data and control information flowing in the AML at all times. A warning! The basic structure of this test is built around the notion of using OI's to request the next output character. The PTIP does not use OI's, but rather makes some timing estimates about the various line rates, and produces output on that basis.

In operation, this program will produce a confusing set of values in all the MLC light registers, since multiple lines are running and sharing the MLC lights. However, INBUF, OUTBUF, OIBUF,OTABUF and INABUF should show signs of activity, and have plausible values in their line number fields, as well as no BREAK indications anywhere. By selecting a particular line with the LINE SELECT switches of the MLC front panel, the display for that particular line will appear in the lower two rows of lights on the MLC front panel. These should have the expected values for in and out rates, size etc. and should have characters flowing in the output and input buffers and shift registers. Note: The switches on the front panel of the MLC are in read in octal and not hexidecimal from the lowest order bit to the right. This sub-program does extensive error checking which is explained in the error description portion of this writeup. Errors are signalled by a non-zero value in the lower control panel light register (DATA). This light register will display the error code for the most recent error. The program builds a list of errors as they occur, starting

at ERRTAB which is at location 1C00. A summary of the error codes is included on the first page of the program listing.

10.4 START3 (208)

This program is an elaborate version of START2. The basic tests are the same, as are the visible indications of its operation.
The main difference is in its startup phase. Whereas,in START2 the user is required to specify which lines are to be tested, this program does that automatically. The user must set the correct value at DEVADR, to denote the AML to be used. The program, upon being started, resets the AML and MLC. It then attempts to force a single character out each LIU port, which it has cross-patched. For each line that responds by sending back the test character, the program selectes a value at random, from the table of asynchronous line rates (2400 baud and below) RATAB. This data rate is used to create an entry in RUNTAB for the line. After this test and setup phase the program begins execution of START2.

The use of this program is primarily for long term burn in and verification of the entire set of line interfaces. It should be used as a final acceptance test rather than as a detailed debugging tool.

10.5 START4 (20C)

Where the previous programs have been oriented toward diagnosing troubles within the AML and the MLC, this program is oriented outward. Much of the complexity of the terminal handling problem has to do with problems between the terminals and the scanner hardware. This program helps pin down that class of problem by allowing the attachment of external terminals or dial in modems.
The objective is to provide some simple programs to answer a dataphone and then transmit back to the caller (or local terminal) each character as it is received without requireing that the very complex PTIP or TENEX terminal support software be workable. The program answers incoming calls in the same way as the PTIP (by watching for RING DETECT) rather than like the the 316 (which watches for CARRIER DETECT). Other than that, no claim is made that this program will behave like the PTIP.

The user must set the correct value for DEVADR. He must also set RATE to the rate which will be used for all lines. When the program is started, it does an LIU search like START3 (using the same subroutines) and builds a RUNTAB. Then, rather than cross-patching all lines, it leaves them un-cross-patched and awaits any input character. In this wait loop, it also polls the LIU status bits for each line. If the ring detect comes true the program launches into a complicated proceedure for maintaining a dialed-up connection on that port. That is best understood by reading the code at ANSWER. Very briefly, operation is as follows. Upon receipt of RING DETECT, the program asserts DATA TERMINAL READY and REQUEST TO SEND. If the modem responds with CARRIER DETECT and CLEAR TO SEND (which means that the caller is a dataset and not a normal telephone) the entry in TOTAB (timeout table) for this line is set non-zero. Incoming data is then echoed. If the caller hangs up, a value is counted down (because CARRIER DETECT went away) and eventually the connection is broken and the state reinitialized for the next call.

The visible indications from the program are few if there is no terminal traffic. The OTABUF lights will flicker as the various LIU's are polled, and nothing else will happen. As characters are typed on the attached terminal, they will be seen to shift into INDAT and move to INBUF. They should also be seen moving out through OUTBUF and OUTDAT. A good character to try if you can't think of anything else is ASCII "A" which appears as either Hex "C1" or Hex "41" dependig on the terminal's parity mode (which doesn't matter here, since nothing in the hardware or program looks at this bit).

Some errors are recorded, such as input or an OI from an unexpected line.

10.6 START5 (210)

START5 is a simple operator aid which does no testing. Its purpose is to clear memory selectively before a test is started. It clears: RUNTAB, all pointer tables, the timeout table and the error table. It does not clear the program stack, and of course, does not touch any part of the code or DDT.
It transfers control to DDT when it completes. It produces no display on the MLC.

10.7 START6 (214)

This collection of one-time tests tries to excercise every path in the AML-MLC combination. The operator must specify the AML at DEVADR. When started, the program hunts for a working line using the standard crosspatch and force subroutine. The first test on a working line turns off XPCH and forces data. It expects only OI's back. Both no OI and input data are errors here, and are logged.

OI's are then turned off, and more output is forced. Any input is an error here. To ensure that the input and output FIFO's are of the correct length, the next test sends a block of 64 characters out, with XPCH off. These should be held in the out FIFO. When XPCH is then turned on, this block should return in order. Errors are noted. The selected line is then made to handle data at every asynchronous rate from 75 to 2400 baud. If a line passes all these tests, it is considered perfect, and the next working line is discovered and tested in a similar fashion.

10.8 Error Indications

Errors are entered in an error buffer beginning at 1C00. An error entry consists of 5 consecutive memory locations which begin with the code which is unique for that error. The next two locations are values which depend on the error type and the last two locations give the time as copies of HITIME and LOTIME which is real time since the beginning of execution.
A few errors are fatal, but most are not and attempt to restart in some way. The various timeout errors are sometimes confusing, since in general they will exit to a part of the program which tries to restart whatever timed out. This occasionally will cause additional data errors.

Visible indication that an error has occurred is given in the control panel data lights, which will contain the error code for the most recent error. This is cleared at initialization so any non-zero value denotes an error.

10.9 Error Types

These are the error types generated by the program. The description here will omit mention of the last two entries since they are the same for each error type; the time of the error.

•

FFF0 LVL1CD -
A level 1 interrupt occurred and the device code was not FF80 (attn interrupt)
word 2 - the bad device code
word 3 - program counter at interrupt time

FFEF LVL2CD - Any level 2 interrupt is an error
word 2 - The bad code
word 3 - Program counter

FFEE LVL3CD - Any level 3 interrupt is an error.
word 2 - Device interrupt code
word 3 - Program counter

FFED LVL4CD - An unexpected level 4 interrupt occurred.
word 2 - The unexpected code
word 3 - Program counter

FFEC QUITCD - An unexpected QUIT occurred
word 2 - Program counter
word 3 - The address which failed

FFEB ILLOCD - An unexpected illop occurred.
word 2 - Address of the ILLOP
word 3 - The illop

FFEA PFALCD - A power fail interrupt occurred.
word 2 - Processor status
word 3 - Program counter

FFE9 PRSTCD - A power restore interrupt occurred.
word 2 - Address the program is restarting from
word 3 - ZERO

FFE8 INERCD - Input error, data did not match expected.
word 2 - Received data and line number
word 3 - Expected data

FFE7 OIERCD - Received an OI from a line we aren't using
word 2 - The OI we got
word 3 - ZERO

FFE6 OITOCD - OI time out
word 2 - Line which timed out
word 3 - ZERO

FFE5 INTOCD - Input time out
word 2 - Line which timed out
word 3 - ZERO

FFE4 FLOTCD - Output fifo was full
word 2 - AML device address (default R2)
word 3 - AML status register (XXX2)

The next set of errors occur in the one time tests

FFE3 OTFLCD - Output fifo was full
word 2 - Line number we are currently testing
word 3 - ZERO

FFE2 OTNOCD - Got no OI in OI test
word 2 - AML status
word 3 - Line we are currently testing

FFE1 OTCHCD - Got data with XPCH turned off
word 2 - AML status
word 3 - Contents of INDAT (what we got)

FFE0 OTOICD - Got the wrong OI in the OI test
word 2 - Line we are testing
word 3 - Contents of the OIDAT we received

FFFDF OTOJCD - Got an OI with the OI's turned off
word 2 - Line we are testing
word 3 - OI that we got

FFDE OTINCD - Got an input character,in OI test,XPCH and OI off
word 2 - Line we are testing
word 3 - Contents of INDAT that we got

FFDD OTSECD - Characters out of sequence in the fifo test. This
probably means the input or output fifo is dropping characters
when it fills up

word 2 - Line we are testing
word 3 - Character we got from INDAT

FFDC OTNICD - No input in fifo test
word 2 - Line we are testing
word 3 - Last thing we sent from OUTDAT

FFDB OTOSCD - Got an OI during the fifo test with OI's off
word 2 - Line we are testing
word 3 - What we last sent out

FFDA OTSOCD - Character does not match expected input
in fifo test.
word 2 - Line we are testing
word 3 - Data we got in INDAT

FFD9 OTNMCD - Bad input character during data rate test
word 2 - Line we are testing
word 3 - Data we got in INDAT

11 MSPMTST

MSPMTST along with DDT-41 forms an exerciser for the BSO card. This test is not a comprehensive diagnostic, but a good exercise program and a go/nogo test. If a card does not work, then one can with a logic analyzer trace where the micro processor is and determine which circuts may be at fault and trace back to the problem from there.

11.1 Micro DDT

Micro DDT is an extension of normal PLURIBUS DDT that allows one to look into to the BSO micro memory to examine and change locations, as well as to reset, start, and stop the BSO micro processor. This is done by having DDT have two modes. One mode is normal DDT mode and the other is micro DDT mode. Only one mode can be active at any given time. In the rest of this document micro DDT mode will be referred to as MDDT and normal PLURIBUS DDT mode will be called DDT.

First switching between DDT and MDDT modes is dicussed. Next the MDDT commands are discussed in logical order. MDDT error messages are discussed last.

11.2 Switching between Micro DDT and Normal DDT.

Entering MDDT for the first time is done my typing the BSO address, usally F100, followed by a "^D". It is not necessary to enter the BSO address when entering MDDT again after the first time, unless the BSO address changes. When you enter MDDT, it will make sure that a BSO exist at the given address. If there is no device present then a "Quit" message is printed and you are left in DDT mode. If the device at the given address is not a BSO, then "???" is printed and you are left in DDT mode. If a BSO is present, then a herald ,"(M 204 mode)", is printed and you are now in MDDT. The number printed in the MDDT herald is the version number of the BSO micro program that is supported by this MDDT. Type ^D to exit back to DDT mode, and DDT will print "(DDT mode)" to let you know that you are back in DDT mode. Restarting DDT by any means does not change the mode you are in.

11.3 MDDT Commands

MDDT is very simular to DDT in command syntax. All numbers are typed in and out in hexadecimal. Following is a list of MDDT commands along with an explanation of their use and impact. Items within square brackets are opitional. The square brackets are not part of the syntax.

^R: Reset the BSO hardware and reintialize the software. The address of the MCB is passed to the BSO and the commands in the MCB command byte are executed. MDDT will respond by typing out the version number of the BSO program.

^U: Switch between byte and word access mode for accessing BSO address space. Intially MDDT is in byte access mode and

^W toggles between the two access modes. The BSO can not do word accesses, but instead MDDT fakes it by doing two successive byte accesses. Byte order within a word is the reverse order then that of the SUE proceesor. In the SUE the low order byte is the high byte of a word, and the high order byte is the low byte of word. MDDT does the reverse order, because the 6502 processor does some instructions by accessing two bytes to form an address, and this order is convient for looking at those bytes with MDDT.

[nn]/: Open the byte addressed by nn in the current access mode. After the slash is typed MDDT will print out the address of the BSO being accessed, some spaces, and the contents of that location. If nn is omitted, then the last location openned is reopenned.

[nn]<cr>: If a location is open, then write nn into that location in the current access mode. If the current access mode is byte mode and nn takes up more then 8 bits, then the high order bits are lost. If no location is open, then nn is ignored. If nn is omitted, then the current location is closed, and MDDT waits for the next command.

[nn]^: Same as nn<cr> except that previous location is openned.

[nn]<lf>: Same as nn<cr> except that next location is openned.

^C: Halt the BSO and open the location at which the BSO halts. The halt that the BSO does is a pseudo-halt. The 6502 processor has no halt instruction, and therfore the halt must be faked. This is done by saving the state of the processor in reserved locations in BSO RAM memory and going into an idle loop in ROM memory. A second ^C would halt the 6502 in the idle loop and destroy the state of the first ^C. The locations for saving the processor state in response to a ^C are known to MDDT and be

accessed by the ^A,^X,^Y,^S,^Q, and ^T commands which are explained below.

^A: Open the location of the saved A register from the last ^C.

^X: Open the location of the saved X register from the last ^C.

^Y: Open the location of the saved Y register from the last ^C.

^S: Open the location of the saved processor status register from the last ^C.

^Q: Open the location of the saved stack pointer register from the last ^C.

^T: Open the location of the saved PC register from the last ^C. The PC is a 16 bit register, and therfore MDDT temporarily changes to word access mode, if MDDT is in byte access mode.

^P: Proceed 6502 from the last ^C. This command starts the 6502 with the saved state in RAM memory, and therefore one can change the state that 6502 will get with the next ^P by changing those locations (reference ^A,^X,^Y,^S,^Q, and ^T commands) in RAM memory.

[nn]^G: Start 6502 at address nn. If nn is omitted, then use the address from previous ^G. This command writes the new 6502 PC into the save area in RAM memory and then does a proceed. This command is equivalant to doing a ^T to open the PC save word, then depositing a new PC, and doing a ^P.

nn^B: Write a breakpoint instruction at address nn. The address argument is required. One should open the location first and make a note of the contents so that the breakpoint can be removed later. The break point instruction causes the 6502 to perform a maskable interrupt, which in the current BSO is vectored to loaction

210. Because this is a RAM memory location, the BSO operational program must be loaded so as to be able to field the the maskable interrupt. Furthermore the operational BSO program does not normally check for break point interrupts in the maskable interrupt service routine, because this would in an operational enviornement waste valuable processor bandwidth. The maskable interrupt service routine can be made to check for break points by changing the instruction at BSO address 0215 from a CLC (opcode 18) to a SEC (opcode 38).

^D: Return back to normal DDT. A herald "(DDT mode)" will be printed to inform you that you are back to DDT.

11.4 Micro DDT Error Messages

MDDT can respond to anyone of the above commands with one of several error messages as follows:

???: This is MDDT's response to an illegal command or to a syntax error.

QUIT: This message is printed if MDDT gets a quit while accessing one of the BSO registers. If a quit message comes from a ^R, then the address causing the quit is base or base+2. All other access to the BSO registers are at base+4. All MDDT access to the BSO registers are writes, except that the MDDT entry routine reads the base register to validate the device type code.

?II-QUIT: The BSO returned a quit reply code in response to a CNMI command issued by MDDT writing to BSO address base+4. This message is only relevant if the BSO was doing an Infibus access, because there is no such thing as a quit on any internal BSO

busses.

?M-Dead: The BSO did not respond in time to a CNMI command issued by MDDT writing to BSO address base+4.

?M-Bad_reply: The BSO did not return a legal reply to a CNMI command issued by MDDT writing to BSO address base+4.

11.5 MSPMTST program outline

The test program performs the test by continually performing a loop which goes throught the following steps:

1. Intialize MSPMTST data structures and transmit buffers.

2. Start of loop: Display pass (loop) number and error count in console lights. The error count is the count of passes with errors wrather then the count of total errors.

3. Intialize data structures for this pass.

4. Start the BSO.

5. Wait for the BSO to complete the data transfer or to timeout. If the BSO times out, then an error message is printed saying which channels timed out and control is trnasfered back to DDT.

6. Halt the BSO.

7. Compare the transmit buffer to the receive buffer for a line. If there is an error, then action is taken based on the current value of ERRHND which is explainned below.

8. Update the pass number and the error count and then go to step 2.

11.6 Data Structures.

In this section the various MSPMTST data structures are explained.

Settable test parameters

Following is a list of settable BSO parameters and how they effect the test. Each location comes intialized with a default value, as mentioned below. The address of the parameter is given in paranthesis after the name of the parameter. Use PLURIBUS DDT to change the values of the parameters.

ERRHND(108): The value in this location determines how the program will deal with data errors. There are three different values for ERRHND as follows:

-1: This is the default value fór ERRHND. An error message is printed telling which line had the error. Clearing R4 will cause the compare to continue with the next byte.

0: This causes the terminal bell to ring when the first error on a pass is detected. The pass will be aborted and the next pass started. This is a developement tool to be used when testing the BSO to find the maximum throughput rate that it can maintain without data errors.

1: The error will be tabulated and the compare will continue. This error handling mode is useful for overnight runs.

LINES(108): This is a bit mask to tell MSPMTST which BSO lines to test. Bit 0 is for line 0, ... ,Bit 3 is for line 3. A 1 means to test the line and a 0 means don't test the line. The MSPMTST default value is 0F which test all four lines.

BUFPAG(10E): BUFPAG is the PLURIBUS common memory page to be used for transmit and receive

buffers. This normally set to the 400 page, but may be set to any other existing page except the 0 page and the 200 page, which are used to hold the BSO RAM program and the MCB.

XLBITS(118): This contains the value to be loaded into the BSO loop and crosspatch register by the BSO at intialization time. Bits 0-3 are used to control internal cross-patching of a line, and bits 4-7 are used to control external looping a modem. A zero is true and a one is false in this register. When there is no modem connected to a BSO line, then the loop bits are irrelevant. The default value for this location is F0, which cross-patches and unloops all four lines. How to loop test a line will be explained later.

11.7 Status Variables

List here are the MSPMTST status words that may be examined using DDT. The address of a status word is given in paranthesis after the name of the name of the given status word.

PASNUM(11E&120): A double precission count of the number of test passes.

ERRCNT(122&124): A double precission count of the number of test passes with erors.

LINERR(126): This is a four word table that is indexed by BSO line number. LINERR+0 is for BSO line 0, ..., LINERR+6 is for BSO line 3. Each entry in the table counts passes that the given BSO line has had data TX errors.

BADPID(12E): The last illegal PID value read

from the PID. This status word is not cleared between passes. A value of zero means that no illegal PID values have been detected scince the test was last started.

11.8 Buffer Structure

Currently the buffer structure for MSPMTST is very simple. A single PLURIBUS common memory page as specified by the BUFPAG parameter (see section 4.1) is divided up into 8 equal size buffers. The first four buffers are for the four transmit(TX) channels (0,2,4, &6), and the last four buffers are for the four receive(RX) channels (1,3,5, &7). The four TX buffers are initialized once when MSPMTST started. The data stored in the TX buffers is the line number in the low byte of every word in the TX buffer, and a number from the sequence 0 - FF (hex) in the odd byte of a word. The number sequence is repeated sequentially in succesive words of the buffer.

11.9 Run Procedure

The procedure for running the diagnostic is as follows:

1. Load paper tape [BBND]PS:<X25MIC>ddt-41.bin.516.

2. Load paper tape [BBND]PS:<X25MIC>bsotst.bin.1. This tape contains the BSO diagnostic and the operational micro program that is loaded into the BSO RAM memory.

3. Enter into MDDT (see section 2.1) to see if the BSO will respond to MDDT commands.

4. Type "^R" to reset the BSO. This reintializes the software as well as the hardware. If the BSO is sucessfully reset, then MDDT will print the

version number of the micro program currently loaded in the BSO. This number should be the same as that typed in the MDDT herald. If the BSO does not reset, then a "?M-Dead" message will be printed. This message is printed anytime that the micro doesn't complete a command in time. There are several possibilities for the BSO not to respond to a command. For the BSO to complete a reset requires all Infibus, micro processor, and interrupt circuts to be functional as well as the Micro Control Block (MCB) for the BSO in PLURIBUS common memory to be valid.

5. Type ^D to return to DDT mode.

6. Set the BSO parameters (see section 4.1, if any need changing, in PLURIBUS local memory using DDT.

7. Type "^G" to start the BSO test program. It will print "Auto test MSPM". As the test runs it will count every pass in the console address lights and every pass with errors in the console data lights. Both DDT and MDDT will still run while the test is in progress, but certain commands could cause the test to fail. The BSO parameters mentionned above may be changed at anytime without causing a failure.

The above steps outline a normal list of sequences for running the test.

12 PIDTST

12.1 Operating Instructions

Load the tape and the program should start. If it halts instead of starting, there is no TTY where it expected one; see Parameters. Pressing RESET then ATTN on the operator panel will cause a restart. The processor may also be started at BEGIN.

12.2 Parameters

Location(Hex)	Default Value(Hex)	Function
1700	E000	BEGIN
1ACC	E000	Pid Addr.
1AD8	FC10	TTY
1ADA	FC10	TTY
1ADC	FC10	TTY
1ADE	FF80	CONSOLE

If the configuration under test is not according to the above default values, change the above noted locations to the appropriate values.

If the PID is on the same bus as the processor, which is not totally (4-bits) F-stuck, then patch 1700 to a "KEY3" (0813!) and set the switch on the PID that sticks the other 2 bits. Note!! The program memory must then be unkeyed.

It should be noted that the default values presently in this version of PIDTST are NOT what is normally found as of the date this document is published. Therefore, the appropriate registers will have to be set.

12.3 Indicators and Errors

All errors are indicated by messages on the teletype. After printout, the program will loop on the error condition unless it is a drastic error (QUIT or ILLOPR) whereupon the program will just halt. Errors will count in the data (lower) lights on the operator panel. Successful retries will count in the address (middle) lights.

12.4 Algorithm

PIDTST consists of a series of tests. The first checks for proper address recognition of the various registers. Next the program empties the PID and makes sure it stays empty. Next the program sets each level in turn, reads it to make sure it was set, and reads again to make sure it cleared. Next all levels are set and then read back to make sure they come back in order. Next, for each level, all levels are set and the identified level cleared. Then all levels are read back to make sure the correct level was cleared. The main loop of the program consists of random writes, reads, and clears. The status of what should be in the PID is maintained in memory and each read is checked for consistency.

13 PLI-KG-TST

13.1 Introduction

A copy of PLI-KG-TST has to be loaded to each of the two processors. The two processors exchange control messages via the direct paths connecting them (the optical isolators path, and the capacitors path). Thus, these paths (called in short the direct path) should be tested before the PLI-KG-TST program is used (e.g., by the PLI-OPTCAP-TST program).

The user interacts with the program mainly via the TTY. The program needs only one TTY (connected to RED), and the direct path is used for exchanging information with BLACK. (This is another reason why the direct path has to be tested before the program is used.)

The user has complete control on both processors via two means: first, through the initial conversation with the program which has the flavor of answering the program questions; and second, after the test starts, by means of on-line commands which are recognized by the program. The program automatically finds whether the processor is the RED processor or the BLACK processor.

The test performed by the program is quite simple. The RED processor sends a block of data whose format is user selectable (either "random" or "alternate"), to the BLACK processor through the KG and monitors the transmission. The BLACK processor receives the encrypted block and keeps it in its memory. It then sends the encrypted block to the RED processor (through the KG) four times. The RED processor compares the received blocks with the original block; a data error occurs each time a mismatch is detected. The RED processor counts the number of good blocks received and the number of bad blocks received (those containing at least one data error). The test continues repeatedly and important information is displayed on the console lights.

The operation is slightly more complicated since some means of synchronization of the processors is needed. In fact the BLACK processor

is the one which controls the test. It instructs the RED processor via the direct path what to do next (e.g., send a block to BLACK, start receiving a block, etc.).

The program is designed to survive power failures (and later power restores) in the KG, and in one or two of the processors; therefore, various timeout counters are maintained by each processor. When some timeout occurs (e.g., TX path timeout), the processor resets the other processor and the test proceeds.

Each time an error is detected by the program an appropriate message is printed on the TTY, and some error counter is incremented. Some information is stored about the nature of the error. In simple cases (such as some timeout) the time of the event is recorded. In cases where more information is needed to describe the error, i.e., when a data error occurs, the state of the RED processor when the error occurred is recorded.

The information describing the errors is in fact pushed onto one of several stacks, which are together called the error log. The stack containing the data error information is called the data error log. Each stack can contain the history of up to five events (e.g., a TX timeout, a data error, etc.). The error log can be printed or cleared upon a user request. In some cases, after an error is detected (e.g., a timeout) the detecting processor resets the other processor and the test proceeds; in some cases, the test proceeds without any reset. In both cases, the error log is not cleared.

During the test, each time the BLACK processor detects a power restart interrupt it resets the RED processor and restarts the test. Each time the RED processor detects a power restart interrupt, it restarts the test (assuming that it was caused by the BLACK processor which reset the RED bus). The program maintains various counters, some of them are displayed on the console lights, and the important ones can also be printed on the TTY by an on-line command.

The program structure is flexible and allows the user to select a wide variety of options; thus, creating different tests. Some of the selections can be made during the test by means of the on-line commands, and some of them can only be made during the conversation which precedes the test; the conversation can always be restarted via an on-line command or by using RESET-ATTENTION.

Each processor should have at least 8 K words of memory and a console at ^HFF80. The RED processor needs a TTY interface at ^HFA00; another TTY address can be used by storing the correct address at location TTYADD which is currently at ^H104. The optical isolators, the feed through capacitors, and the KG should all be appropriately connected.

For a successful operation each processor bus must also contain at least one interface card from the correct type (KGB or KGR). The program looks for devices in the range ^HC000 to ^HFE00. The program allows the user to select one device if several are found. However, in case of BLACK the program may fail in trying to communicate with RED if before the user makes his selection, the program chooses the wrong device (the one not connected to RED through the direct path). In such case, the user has to enter the desired KGB address to location CRDADR which is currently at ^H106, and restart the conversation with BLACK.

13.2 Loading And Starting The Program

Before loading the program each processor bus should have power turned on. Power restart interrupts should be enabled (S71 in the console in the middle position which enables the power recovery int). Line frequency interrupts (JIFFY) should be enabled (S70 in the console in the ON position). The bus reset timer of each processor bus should be in the OFF (up) position.

The program is first loaded into both processors; the recommended order is RED, then BLACK. RED automatically starts the conversation. At the end

of the conversation with RED, the conversation with BLACK starts automatically. It may happen that the first messages of BLACK are lost (e.g., when the programs are not loaded in the recommended order); in such case, the conversation with BLACK can be restarted by one of the ways described in the next paragraph. When the conversation with BLACK is completed, the test itself starts.

The conversation can always be restarted by one of the following ways: using RESET-ATTENTION, restarting the program at ^H100, replying "E" to one of the program questions during the conversation, or via the on-line command "B" (begin conversation) during the test itself. However, precautions must be taken to make sure that the other processor will not reset this processor. Such precautions can be: halting the second processor via the console (only BLACK can be so halted ,since BLACK needs RED running during its conversation), or causing the other processor to restart the conversation.

Note that the BLACK processor will always wait for the answer to its first question in the conversation, and RED will start sending to BLACK the characters typed by the user only after the conversation with RED itself has been completed. When RED finished its conversation but the test itself has not yet started, RED is said to be in copy mode, since it simply transfers characters between the TTY and the BLACK processor.

Causing a power restart interrupt to one of the two processors (by turning its power OFF and ON) can be used to restart the test; it is recommended to turn OFF/ON BLACK since it will anyhow reset RED. The same effect can also be achieved by the on-line command "R" which can be given to either the RED processor or the BLACK processor. Note that turning off the power on one of the processors may cause some errors to be detected by the other (timeouts, or data errors); these errors can be deleted from the error log by clearing the error log via the on-line command "C".

13.3 The Initial Interaction With The User

The program starts by interacting with the user, which has to define the program's mode of operation. The conversation is first conducted with RED, and then with BLACK. The answers to the questions are:

Y- yes, N- no, or E- exit.

"E" means exit and restart the program at the beginning of the conversation with the current processor. (There is also a way to restart the conversation with RED at the end off the conversation with BLACK.)

Each of the above characters must be followed by a carriage-return. If an illegal pair of characters is typed by the user, the program prints the string "? (Y/CR,N,E)" and waits for a legal response.

The program interprets a single carriage-return as the character "Y" followed by a carriage-return.

Most of the questions and messages printed by the program are preceded by the processor identification and have the form: <COL>* where <COL> (color) can be either BLK or RED. The messages and questions printed by the program are described next.

1. PLI-KG-TST

The program name is the first message printed by the program.

2. <COL>* HARD COPY? (Y/CR,N,E)

If the user replies negatively, implying a CRT device, then the program waits, whenever the screen is filled while printing the current summary, for the user to type any character; otherwise, the program does not wait for such characters. In case of a hard copy device,

the messages are printed more slowly than in the other case (by intentional delays within the program).

3. <COL>* SUMMARY? (Y/CR,N,E)

This question is asked by the program only if the conversation is restarted after the test has begun. If the user replies positively, the current status and the error log are printed (see a later description). Typing the character "X" or the character ^O (control O) during the summary printing aborts the printing.

4. TYPE=tt ADDRESS=aaaa CODE=c RATE=rr KB

This message describes the interface card found by the program (the KGB or the KGR). CODE is the contents of bits 0-3 of the DT (device type) word in the device register block. Rate is printed only by the BLACK processor. If several interfaces with the same type are found, the program prints their details and the user has to select one of them by typing one digit (as explained by the program in this case).

5. <COL>* CHANGES? (Y/CR,N,E)

If the user does not want to make any change in the program's mode of operation, then the character "N" should be typed. If this is the first iteration of the program, the answer "N" denotes the default answer to each of the following questions. If there are no changes the program continues on step 14. If there are changes, the conversation continues on the next step.

6. RED* TRANSMIT "RANDOM" PATTERN? (Y/CR,N,E)

This question is asked only by the RED processor. If a positive answer is given, the RED processor will use a pseudo-random table as the source of data to be transmitted and received. If a negative answer is given, the RED processor will use a table consisting of 8 words: ^HFFFF, 0000, ^HFFFF, 0000, ... , as the source of the data to be

transmitted and received. The default answer is "Y".

7. BLK# FIXED LENGTH? (Y/CR,N,E)

This question is asked only by the BLACK processor. If a negative answer is given, then the length of each transmitted block is randomly selected (in the range ^H0010, ... , ^H0500 bytes), and the conversation continues in step 9. If the answer is positive the conversation continues in step 8. The default answer is "N".

If the current length is n, then in fact the number of bytes passed on each path is slightly greater than n due to a header for synchronizing the KGR, and due to the MI pattern. In addition to these bytes, RED transmits additional 3 bytes, and BLACK receives and transmits additional 10 bytes per block. These (magic) numbers have been found experimentally, and can be explained only partially.

8. BLK# LENGTH=

Following this message, the user has to type a four digit hexadecimal number in the range ^H0010, ... , ^H0500. This number is the intended length of all blocks to be transmitted by RED. The program converts the number typed by the user to an appropriate number and prints the selected length.

9. <COL># KG NOISE? (Y/CR,N,E)

A positive answer will cause the program to send bytes through the KG to the other processor when that path is not used, thus creating noise for the other path. The answer is stored in a program switch whose status can be complemented during the test via the on-line command "N". The default answer is "Y".

10. <COL># DIRECT PATH NOISE? (Y/CR,N,E)

A positive answer will cause the program to send bytes through the direct path to the other processor, thus creating some noise over the paths involving the KG. The answer is stored in a program switch whose status can be complemented during the test via the on-line command "N". The default answer is "Y".

11. <COL># TX PATH HICCUP? (Y/CR,N,E)

If a positive answer is given, this processor ignores the TX path for one second every 10 seconds. The answer is stored in a program switch whose status can be complemented during the test via the on-line command "H". The default answer is currently "Y" for BLACK, and "N" for RED. (In the future, when flow control over the TX path is installed in RED, the program will be modified such that the default is "Y" in both processors.)

12. <COL># RX PATH HICCUP? (Y/CR,N,E)

If a positive answer is given, this processor ignores the RX path for one second every 10 seconds. The answer is stored in a program switch whose status can be complemented during the test via the on-line command "H". The default answer is "Y" for both processors.

13. <COL># KEEP EARLIEST ERRORS (INSTEAD OF LATEST)? (Y/CR,N,E)

In case of a positive answer the program does not push more entries onto a stack of the error log after it is filled with 5 entries (thus the first 5 entries can be saved). In case of a negative reply, the latest 5 entries are kept in each stack of the error log, and previous entries are forgotten. In any case, the error log can be cleared during the test via the on-line command "C", or in the conversation as described in the next step. The answer is stored in a program switch whose status can be complemented during the test via the on-line command "K". The default answer is "Y".

14. <COL># CLEAR LOG? (Y/CR,N,E)

The error log from previous tests is cleared when a positive reply is given. The various counters are also cleared in this case. A positive reply causes an automatic "C" (clear) on-line command to be executed when the test begins in order to synchronize the clocks of both processors. A negative reply causes keeping of the error log and the counters from previous tests. The error log and the various counters can be cleared during the test via the on-line command "C".

15. <COL># RESTART CONVERSATION? (Y/CR,N,E)

A positive reply causes restarting the conversation with RED. "E" causes restarting the conversation with the current processor. After a negative reply, if the current processor is RED, the conversation continues in the next step; if the current processor is BLACK, the conversation proceeds in step 17.

16. RED# NOW TALK TO BLK

This message is printed by RED after a negative reply was given in step 15. Following this message, BLACK should immediately start its conversation in step 1. If no message is printed at that point by BLACK, it may be that BLACK's first messages have been somehow lost. In this case BLACK is probably waiting for an answer to the question in step 2. The easiest way to proceed is by restarting the conversation with BLACK (by typing "E" and a carriage-return, or by pushing RESET ATTENTION on BLACK). If BLACK does not start its conversation, it may be that the direct path is broken.

17. BLK# STARTING TEST ...

This message is printed by BLACK after a negative reply was given in step 15. After some delay, BLACK resets RED and then the test itself starts. From this point the on-line commands are recognized by the program. Initially the commands are

directed to RED.

13.4 The On-Line Commands

The on-line commands are recognized by the program during the test itself; each consists of one or more characters (underlined in this section). After the program identifies the command it prints a short description of the command. The test continues during the on-line interaction without any delay (except of course in the cases of commands which explicitly halt the test or some part of it).

Since there is only one TTY, connected to RED, RED maintains an internal switch indicating which processor is currently interpreting the user's commands (or to which processor the user currently talks). At the beginning of the test and after every power restart the user talks to RED; the on-line command "T" (talk to) can be used to talk to the other processor. The messages printed in response to an on-line command are preceded by the processor color; thus the user can identify the printing processor at any stage. The commands, the messages, and their meanings are described next.

B <COL>* BEGIN CONVERSATION

This command causes restarting of the conversation. It can be given to any of the processors; the other processor is automatically notified and restarts its conversation. The error log and the counters are kept; they can be printed during the conversation and even be kept for the continuation of the test. If the command fails, the conversation can be restarted by using RESET-ATTENTION on both processors (the recommended order is RED then BLACK).

C <COL>* CLEAR LOG

-

This command can be given to any of the two processors, and is automatically sent to the other processor. It causes clearing of the error log (thus allowing new entries to be pushed onto the various stacks) and the various counters (including the time counter).

-

This command causes printing the details of the selected device (KGR or KGB) in the same format as in step 4 of the conversation.

F <COL>* FORCE ERROR
-

One of the bytes sent by this processor to the other one through the KG is intentionally complemented to force a bad block. The error should be detected by RED, which should print "data error", as explained later.

H <COL>* HICCUP (T/R) TX PATH (ON/OFF) or
- <COL>* HICCUP (T/R) RX PATH (ON/OFF)

This command complements the current status of the program switch described in step 11 or 12 of the conversation, depending on whether the user has typed "T" or "R" respectively. The new status of the switch is printed.

H <COL>* KEEP ERRORS (ON/OFF)
-

This command complements the current status of the program switch described in step 13 in the conversation. The new status of the switch is printed.

N <COL>* NOISE (K/D) KG (ON/OFF) or
- <COL>* NOISE (K/D) DIRECT PATH (ON/OFF)

This command complements the current status of the program switch described in step 9 or 10 of the conversation, depending on whether the user has typed "K" or "D" respectively. The new status of the switch is printed.

```
0    <COL>* OPEN: aaaa rrrr wwww <terminator>
-
```

The open command allows the user to access and optionally modify any location in the addressed processor. The command should be carefully used in order not to destroy the programs. Note that reading a register of the KGB or the KGR may change its contents and therefore cause problems to the program.

aaaa (typed by the user) is a four digit hexadecimal address
of the location to be opened, or the character ESC
which stands for the previously opened location.

rrrr is the contents of the opened memory location.

wwww (typed by the user) is an optional field which specifies the new contents to be written into the opened location.

<terminator>(typed by the user) is one of the following characters:

Carriage-return which confirms the new contents (if typed), and terminates the command.

Line-feed: which confirms the new contents (if typed), and automatically opens the next location.

^ which confirms the new contents (if typed), and automatically opens the previous location.

Any other character used as a terminator, aborts the OPEN command and does not confirm the new contents of the opened location (space is convenient to use).

If quit is detected while trying to access the opened location the word QUIT is printed by the program, and the OPEN command is aborted.

R <COL># RESET

This command instructs the processor to reset the other processor in order to restart the test. The test restarts with the same mode of operation; the error log and the various counters are not modified. The command can be used when some processor seems to get stuck. A similar effect can be achieved by turning OFF/ON the BLACK processor.

S <COL># SUMMARY:

This command causes printing of the current status and the current error log (if it is not empty). The format of the printed tables is given in a later section. The character "X" or the character ^O (control O) can be typed by the user to abort the printing.

T RED# TALK TO BLK or
BLK# TALK TO RED

This command allows the user to start talking to the other processor. The command is recognized by the currently selected processor; therefore, if this processor is BLACK and it is not working properly from some reason the command cannot be used in order to restart talking to RED. The solution in such cases is to turn OFF/ON one of the processors; since whenever RED identifies a power restore interrupt it marks itself as the processor currently talking to the user.

X or ^O (control O)

These characters can be typed by the user during the short interruptions between printed lines of the summary or the list of on-line commands; they abort the printing.

? <COL>* ON-LINE COMMANDS:
-

This command causes printing of the list of available on-line commands. The character "X" or the character ^O (control O) can be typed by the user to abort the printing.

13.5 The Summary Format

The summary is printed when the on-line command "S" is typed during the test, or during the conversation (at step 3). The printing can be aborted by typing the character "X" or the character ^O (control O) in the short interruptions between printed lines.

The summary consists of two parts: the current status, and the error log. In case of RED, the error log contains the data error log which is not printed unless some data errors are recorded in it. An error counter whose contents is zero is not printed. Some of the printed items are marked by the character ' (a single quote); these items indicate some event which could result from the normal (but strange) operation of the KG. Some of the printed items are marked by the character " (a double quote); these items indicate serious errors associated with the KG or the interfacing cards. The program keeps together with some of the error counters information about the times of up to five corresponding events. These times are printed immediately after the counters and they will not be shown here. The format of the two parts of the summary is described now.

13.5.1 <COL># CURRENT STATUS:

This is followed by the current status of this processor. Some of the entries in the status table are printed only by one processor and this will be indicated below.

<COL># TIME nnnn

This is the current time in seconds.

<COL># # TX GOOD BLOCKS nnnn

This is the number of blocks successfully passed on the TX path.

<COL># # RX GOOD BLOCKS nnnn

This is the number of blocks successfully passed on the RX path.

<COL># CURRENT LENGTH nnnn

This is the length of the current block.

BLK# # ALARMS nnnn

This is the number of alarms detected by BLACK.

<COL># # B2R TIMEOUTS nnnn

This counter counts timeouts on the direct path. In case of BLACK it counts cases where RED did not read on time a command sent by BLACK. In case of RED, it indicates the lack of commands from BLACK.

BLK# # SPURIOUS PREPS nnnn

This is the number of spontaneous PREP's; i.e., PREP's generated by the KG, not in response to a PREP request signal from the RED processor.

RED* # LOST PREPS ' nnnn

This is the number of PREP's issued by RED which didn't cause BLACK to identify the complete MI (message indicator) pattern; probably PREP's dropped by the KG.

BLK* # BAD MI'S " nnnn

This is the number of MI's which were only partially correct.

BLK* # UNEXPECTED ACKS ' nnnn

Each time RED receives a complete block it sends an acknowledgement to BLACK. This counter indicates the number of ACK's which arrived when BLACK did not expect an ACK. This may happen for example, if RED does not interpret correctly the command from BLACK specifying the length of the block (e.g., an error on the direct path), and selects a shorter block. RED will acknowledge the block before BLACK completes the sending.

RED* # RX BAD BLOCKS " nnnn

This is the number of blocks received from RED which contained at least one error. Information about the first error in such blocks is normally found in the data error log (unless it is full).

RED* # TX UNDERRUN " nnnn

This is the number of blocks sent by RED where SYN characters were (automatically) inserted by the KGR into the transmitted block due to lack of bytes in the KGR TX fifo.

BLK* # TX TIMEOUTS ' nnnn

This is the number of TX path timeouts while BLACK
still hasn't received the whole MI.

<COL># # TX TIMEOUTS " nnnn

This is the number of TX path timeouts; note that for BLACK this has been split to two phases.

<COL># # RX TIMEOUTS " nnnn

This is the number of timeouts on the RX path.

<COL># # RESTARTS nnnn

This is the number of times this processor detected a power restore interrupt from the latest "C" (clear) command. Note that after a real power fail, RED normally detects two power restore interrupts: the first when the line power is restored, and the second after BLACK resets it in order to restart the test.

<COL># # KG ON-OFF'S nnnn

This is the number of times this processor detected a transition in the KG power on bit from ON to OFF. Errors detected by the program when the KG power is OFF can be ignored.

<COL># # KG OFF-ON'S nnnn

This is the number of times this processor detected a transition in the KG power on bit from OFF to ON.

The format of the data error log printed by RED is defined below.

RED# ERROR LOG LAST -1 -2 -3 -4

Below this line, information associated with up to five data errors is printed in two separate groups. The latest entry in the data error log is printed under the word LAST, the previous entry under -1, etc. The first group describes the

bad (received) block:

13.5.2 RED# BAD BLOCK:

This title precedes the information about the bad block. In fact, this information summarizes the way the four blocks associated with the same transmitted block have been received; complete information is kept only about the first data error in the first bad block (of the four blocks).

RED# READ DATA " nnnn

The first bad byte read from the input fifo of the KGR.

RED# XPCTD DATA nnnn

The expected data (read from the program table).

RED# TABLE ADDR nnnn

The address of the expected byte in the program data table.

RED# ERROR INDEX nnnn

The index of the bad byte in the bad block (0,1,2,...).

RED# BLOCK LENGTH nnnn

The length of the current block.

RED# # BAD BLOCKS " nnnn

The number of bad blocks received in response to the previously transmitted block (1-4).

RED* # RX BLOCKS nnnn

The number of blocks received in response to the previously transmitted block; this number should be four unless operation of some processor or device is somehow aborted.

RED* # 1'ST BAD BLOCK nnnn

The number of the first bad block out of those received in response to the previously transmitted block (1-4).

RED* DIFF ERRORS YES/NO

This indicates whether there were different errors in the bad received blocks. In fact, only the first error in a block, and the error index are compared for the bad received blocks.

13.5.3 RED* COUNTERS:

This title precedes the second group of the data error log. This group contains the values of several important global counters at the time of the error. The meaning of these counters was explained earlier in this section.

```
RED* TIME                    nnnn
RED* # TX GOOD BLOCKS        nnnn
RED* # RX GOOD BLOCKS        nnnn
RED* # TX UNDERRUNS      "   nnnn
RED* # RX BAD BLOCKS     "   nnnn
RED* # RESTARTS              nnnn
```

13.6 The Console Lights

The console lights display information that shows how the test progresses.

The contents of the ADDRESS LIGHTS from right to left is:

BITS_0-7: Current state of this processor (described later in this section under Current State).

BITS_8-11: Bits 0-3 of the sum of the counters indicating serious errors (marked by " in the summary) on the TX path.

BITS_12-15: Bits 0-3 of the sum of the counters indicating serious errors (marked by " in the summary) on the RX path.

The contents of the DATA LIGHTS from right to left is:

BIT_0: This bit is ON when the KG power on bit is ON.

BIT_1: This bit is ON when the KG alarm bit is ON.

BITS_8-11: Bits 0-3 of the counter of successful block transfers on the TX path (good TX blocks).

BITS_12-15: Bits 0-3 of the counter of successful block transfers on the RX path (good RX blocks).

If the user wants to use the console he first has to press ATTENTION (without RESET), and this stops the displaying. An additional pressing of the ATTENTION button restores the displaying.

The user can cause the program to display other information in the console lights. ALITSP (currently at location ^H10E) and DLITSP (currently at location ^H110) are pointers to the words to be displayed in the ADDRESS LIGHTS and the DATA LIGHTS respectively.

The current state of each processor can be determined by the contents of bits 0-7 of the ADDRESS LIGHTS. In general, bit 7 OFF indicates that the TX path is active, and bit 7 ON indicates that the RX path is active. Bit 6 ON indicates that there is currently a hiccup on the active path (determined from bit 7). The other bits indicate a more refined state. The state information is different for the two processors, and is given below. The contents of bits 0-7 is written as a hexadecimal number.

BLACK TX active:

^H1 BLACK is trying to send to RED the length of the next block through the direct path.
^H2 BLACK is trying to send to RED a command instructing RED to PREP the KG.
^H4 BLACK is waiting for the alternate bits of the MI pattern.
^H8 BLACK is waiting for the non-pattern byte following the alternate bits which should be the inverse of the pattern byte.
^H10 BLACK is waiting for its fifo to fill (with 64 pattern or inverse pattern bytes).
^H20 BLACK is reading information bytes from the TX path.
^H40 BLACK is in TX hiccup mode.

BLACK RX active:

^H81 BLACK is trying to send to RED a command instructing it to start receiving a block.
^H82 BLACK is trying to write into its fifo a group of bytes consisting of 20 alternate bytes, 8 pattern bytes, and one inverse pattern byte.
^H84 BLACK is sending the encrypted information through the RX path.
^HC0 BLACK is in RX hiccup mode.
^H88 BLACK is waiting for an ACK from RED for the last block sent to RED on the RX path.
^H98 BLACK is waiting for an ACK from RED for the last block sent to RED on the RX path; an ALARM was detected during this waiting period.

RED TX active:

^H1 RED is waiting for a command from BLACK on

the direct path.

^H2 RED received the PREP command from BLACK, and is waiting a little (for an internal timer) before prepping.

^H4 RED is waiting for the PREP signal to be OFF before performing the actual PREP.

^H8 RED is sending the information through the TX path.

^H40 RED is in TX hiccup mode.

D RX active:

^H81 RED is waiting for a command from BLACK.

^H82 RED received a command to start receiving a block, and is waiting until a gap is detected in the RX clock (SC bit 1 is OFF).

^H84 RED detected a gap in the RX clock (SC bit 1), and is waiting for the RX clock to become ON again.

^H88 RED is waiting for the first non-sync byte from the RX path.

^H98 RED received at least one non-sync byte from the RX path, and is waiting to the rest of the block. ^.uw HC0 4 RED is waiting for the first non-sync byte from the RX path; it is currently in RX hiccup mode.

^HD0 RED received at least one non-sync byte from the RX path, and is waiting to the rest of the block. It is currently in RX hiccup mode.

13.7 The Error Messages

After the occurrence of an error (of most of the error types) the processor prints some message; in some cases, the other processor is reset, and the test continues. In some cases, the test continues without a reset. The error messages are:

RED# UNEXPECTED CODE IN COPY MODE ccc

This message is printed by RED if during copy mode (conversation with BLACK) it receives an unknown code from BLACK through the direct path; ccc

is the accepted code.

RED* UNKNOWN BLK COMMAND ccc

This message is printed by RED during the test if it receives an unknown command from BLACK through the direct path; ccc is the command code.

RED* TX UNDERRUN

This message is printed by RED when a TX underrun is detected (SC bit 14).

RED* DATA ERROR

This message is printed by RED when a data error is detected. The message is printed only once for the whole group of 4 blocks received through the RX path even if there are several data errors.

RED* STACK FULL

This message is printed by RED when a data error occurs, the data error log is full, and the keep error switch (defined in the conversation at step 13, or via the "K" on-line command) is ON. The information about the last error is lost and the data error log is not modified.

RED* POWER RESTART nnnn

This message is printed by RED when a power restore interrupt occurs. nnnn is the new value of the power restart counter.

BLK* BAD PREP

This message is printed by BLACK if the first non-pattern byte following the alternate bytes in the MI is not the inverse pattern.

BLK* BAD MI PATTERN

This message is printed by BLACK if one of the first 64 (decimal) bytes following the inverse pattern above is neither pattern nor inverse pattern.

BLK* UNEXPECTED BLOCK ACKNOWLEDGE

This message is printed by BLACK if an unexpected ACK is accepted from RED. This may occur for example if the length information was not transferred correctly through the direct path to RED.

<COL>* TX PATH TIMEOUT

This message is self explanatory.

<COL>* RX PATH TIMEOUT

This message is self explanatory.

<COL>* B2R TIMEOUT

This message is printed by RED when commands are not accepted from BLACK through the direct path; or by BLACK if it detects that RED does not read its commands (SC bit 5 is OFF).

<COL>* KG POWER OFF

This message is printed when the KG power is first found to be OFF and every 10 minutes afterwards.

<COL>* DEVICE GONE

This message is printed when the device has disappeared (causes quits). Note that when the device disappears, the program may print this message or the "UNEXPECTED QUIT" message (since not all accesses to the device are protected by the QUIT pattern).

The remaining error messages are self explanatory:

```
<COL>* UNDEFINED LEVEL 4 INTERRUPT
<COL>* UNEXPECTED QUIT @aaaa
<COL>* ILLOP @aaaa
```

13.8 Assembly Of The Program

Since the PLURIBUS assembler can handle names of up to six characters the following procedure can be applied. First rename PLI-KG-TST.PLR to P.PLR. Then assemble the new file as explained next.

When P.PLR is being assembled, the user has to supply the assembler with several parameters, each of which consists of one character (followed by a carriage-return). The parameters are self explanatory except for the following one:

If the program should be debugged and traps (illegal operations) are used as a debugging aid, then Y should be answered to the question:

DEBUG MODE?

In this case, a pointer to DDT is stored at the illop. vector by the program. The appropriate version of DDT to be used is: DDT-35.

After the assembly is complete the source file and the files produced by the assembler have to be renamed appropriately.

14 PLI-OPTCAP-TST

PLI-OPTCAP-TST consists conceptually of two parts: a transmitter and a receiver. The user has complete control on both parts via two means: first, through the initial conversation with the program which has the flavor of answering the program questions, and second, after the test starts, by means of on-line commands which are recognized by the program.

The program automatically finds whether the processor is the red processor, or the black processor, and accordingly activates the corresponding transmitter and receiver. The transmitter part sends a block of data whose format is user selectable (either "random" or "alternate"), to the path interface (the KGR or the KGB) and monitors the transmission. The receiver part receives a block of data, compares its format with a user selectable format (not necessarily identical to that of the transmitter) and monitors the reception. Since the interfaces are not DMA interfaces, the sending and receiving are done on a word by word basis. The test continues repeatedly and important information is displayed on the console lights.

Each time an error of any kind is detected by the program an appropriate message is printed on the TTY, some error counter is incremented, and in important cases (receive data error) the state of the receiver when the error occurred is pushed onto a stack called the error log. This stack can contain the history of up to five such errors and can be printed or cleared upon a user request. After the error message is printed, the program waits several minutes (to let both processors print their error logs if the user desires), and then resets the other processor in order to resume the test.

The transmit part of the program sends a word and waits for the completion of the transmission or to a transmit timeout (whichever comes first), then sends the next word after a delay of 50 ns (if it is the red transmitter), or immediately (if it is the black transmitter). A transmit timeout occurs if

within 10 seconds from the beginning of the transmission the red processor didn't read the black to red word (in case of a black transmitter), or if the prep transmitter signal didn't return to 0 (in case of a red transmitter).

The receive part of the program waits for the arrival of a new word or to a receive timeout of 10 seconds (whichever comes first). If a word arrives on time it is compared to the expected word and if the data matches, the next word is expected. If the data does not match, a receive data error is detected.

When an error occurs, the appropriate counters are incremented and the test is restarted (without clearing the counters or clearing the error log). When the red processor decides to restart the test (either after an error, or when the user requested so by means of an on-line command) it resets the black processor and waits to be reset by the black processor. When the black processor decides to restart the test, it resets the red processor, and then both processors actually start the test.

During the test, each time the black processor detects a power restart interrupt it resets the red processor and restarts the test. Each time the red processor detects a power restart interrupt, it restarts the test (assuming that it was caused by the black processor which reset the red bus). The red processor resets the black processor bus, thus forcing it to initiate restarting of the test, every 10 minutes.

The program maintains various counters, some of them are displayed on the console, and the important ones can also be printed on the TTY by an on-line command.

The program structure is flexible and allows the user to slect a wide variety of options, thus, creating different test configurations. Some of the selections can be made during the test by means of the on-line commands, and some of them can only be made during the conversation which precedes the test; The conversation can always be restarted via an on-line command or by using

RESET-ATTENTION.

Each processor should have at least 4 K words of memory, a console at ^HFF80, a TTY interface at ^HFA00 (however, another TTY address can be used by storing the correct address at location TTYADD which is currently at ^H104). The optical isolators and the feed through capacitors should be appropriately connected; however, no KG is required for the operation of the program. For a successful operation each processor bus must also contain at least one interface card from the correct type (KGB or KGR). The program looks for devices in the range ^HC000 to ^HFE00.

14.1 Loading And Starting The Program

Before loading the program each processor bus should have power turned on. Power restart interrupts should be enabled (S71 in the console in the middle position which enables the power recovery int). Line frequency interrupts (JIFFY) should be enabled (S70 in the console in the ON position). The bus reset timer of each processor bus should be in the OFF (up) position.

The program is loaded to one of the processors to which a TTY is connected, and starts automatically with the conversation. At the end of the conversation the TTY can be disconnected from the first processor bus and connected to the other processor bus. The program is then loaded to the second processor memory, and the second conversation starts. At the end of the second conversation the test starts by the second processor.

The conversation can always be restarted by one of the following ways: using RESET-ATTENTION, restarting the program at ^H100, or via an on-line command (E). However, precautions must be taken to make sure that the other processor will not reset this processor. Such precautions can be: halting the second processor via the console or via an on-line command (H), or causing the other processor to restart the conversation (even if there is only

one TTY, since it will wait for the first answer and after plugging the TTY it can be restarted).

Causing a power restart interrupt to the black processor (by turning its power off and on) can be used to restart the test. The same effect can also be achieved by an on-line command (B) which can be given to either the red processor or the black processor. Note that after causing a manual power restart interrupt to the red processor it will start the test assuming that the black processor has reset its bus, and this may cause some error after which an automatic restart will occur.

14.2 The Initial Interaction With The User

The program starts by interacting with the user, which has to define the program's mode of operation. The answers to the questions are:

Y- yes, N- no, or E- exit.

"E" means exit and restart the program at the beginning of the conversation

Each of the above characters must be followed by a carriage-return. If an illegal pair of characters is typed by the user, the program prints the string "? (Y/CR,N,E)" and waits for a legal response.

The program interprets a single carriage-return as the character "Y" followed by a carriage-return.

Most of the questions and messages printed by the program are preceded by the processor identification and have the form: <COL>* where <COL> (color) can be either BLK or RED. The messages and questions printed by the program are described next.

1. PLI-OPTCAP-TST

The program name is the first message printed by the program.

2. <COL>* SUMMARY? (Y/CR,N,E)

This question is asked by the program only if the conversation is restarted after the test has begun. If the user replies positively, the current status and the error log are printed (see a later description). Typing the character "X" or the character ^O (control O) during the summary printing aborts the printing.

3. TYPE=tt ADDRESS=aaaa CODE=c

This message specifies the interface found by the program (the KGB or the KGR). CODE is the contents of bits 0-3 of the DT (device type) word in the device register block. If several interfaces with the same type are found, the program prints their details and the user has to select one of them by typing one digit (as explained by the program in this case).

3a. <COL>* HARD COPY? (Y/CR,N,E)

A positive answer should be given if a hard copy device is used. The program then assumes that this is a slow device, and introduces a special delay when the test begins. If a negative reply is given, the program assumes that the TTY is a fast device, and does not introduce the delay.

4. <COL>* CHANGES? (Y/CR,N,E)

If the user does not want to make any change in the program's mode of operation, then the character "N" should be typed. If this is the first iteration of the program, the answer "N" denotes the default answer to each of the following questions. If there are no changes the program continues on step 12. If there are changes, the conversation continues on the next step.

5. <COL>* TRANSMIT? (Y/CR,N,E)

The user can activate or disactivate the transmitter part of the program. The answer is stored in a program switch whose status can be complemented during the test via the on-line command "T". The default answer is "Y".

6. <COL>* TRANSMIT "RANDOM" PATTERN? (Y/CR,N,E)

If a positive answer is given, the transmitter part of the program will use a pseudo-random table as the source of data to be transmitted. If a negative answer is given, the transmitter part of the program will use a table consisting of 4 words: ^HFFFF, 0000, ^HFFFF, 0000. The default answer is "Y".

7. <COL>* RECEIVE? (Y/CR,N,E)

The user can activate or disactivate the receiver part of the program. The answer is stored in a program switch whose status can be complemented during the test via the on-line command "R". The default answer is "Y".

8. <COL># RECEIVE "RANDOM" PATTERN? (Y/CR,N,E)

If a positive answer is given, the receiver part of the program will use a pseudo-random table as the source of data to be compared with the received data. If a negative answer is given, the receiver part of the program will use a table consisting of 4 words: ^HFFFF, 0000, ^HFFFF, 0000. The default answer is "Y".

9. <COL># KG NOISE? (Y/CR,N,E)

A positive answer will cause the program to "play" with several signals controlling the KG operation, thus creating noise for the two tested data paths. The answer is stored in a program switch whose status can be complemented during the test via the on-line command "N". The default answer is "Y".

10. <COL># PRINT STATUS ON ERRORS? (Y/CR,N,E)

A positive answer will cause printing of the current status (without the error log) on each error occurrence, otherwise the status is not printed. However, remember that in either case the processor that has detected an error waits several minutes before restarting the test, and at that time the whole summary can be printed via the on-line command "S". The answer is stored in a program switch whose status can be complemented during the test via the on-line command "P". The default answer is "Y".

11. <COL># KEEP ERROR LOG UNTIL CLEARED? (Y/CR,N,E)

In case of a positive answer the program does not push more entries onto the error log after it is full with 5 data error entries (thus the first 5 data errors can be saved). In case of a negative reply, the latest 5 errors are kept in the error log, and previous errors are forgotten. In any case, the error log can be cleared during the test via the on-line command "C", or in the conversation as described in the next step. The answer is stored in a program switch whose status can be complemented during the test via the on-line command "K". The default answer is "Y".

12. <COL># CLEAR LOG? (Y/CR,N,E)

The error log from previous tests is cleared on a positive reply. The various counters are also cleared. A negative reply causes keeping of the error log and the counters from previous tests.

13. <COL># OTHER PROCESSOR READY? (Y/CR,N,E)

A negative reply should be given if this is the conversation with the first processor in the system. In this case message 14 is printed by the program. A positive reply should be given if this is the conversation with the second processor in the system and the first one has completed printing message 14. In this case, message 15 is printed by the program.

14. <COL>* HALTING (WAITING FOR OTHER PROCESSOR, OR TO

POWER OFF/ON) ...

This message is printed after a negative reply was given in step 13. The processor loops indefinitly, waiting for a power restart interrupt, which is normally caused by the other processor but can also be caused manually.

15. <COL>* RESETTING <COL>

The message indicates that this processor is resetting the other one for one second. It is printed after a positive answer was given in step 13, or during the test when a processor decided to reset the other (as a result of the on-line command "B", an error, or in the case of the red processor every 10 minutes).

16. <COL>* POWER RESTART: nnnn

Each time a processor gets a power restart interrupt it increments an appropriate counter and prints the above message. It then continues in step 17 (if it is the red processor), or in step 15 (if it is the black processor).

17. <COL>* STARTING TEST ...

This message is printed immediately before the test is started. After this point, the first transmission is delayed by one second and both receive timeout and transmit timeout are set to 10 seconds each. From this point the on-line commands are recognized by the program.

14.3 The On-Line Commands

The on-line commands are recognized by the program during the test itself; each consists of one character. After the program identifies the command it prints a short description of the command. The test continues during the on-line interaction without any delay (except of course in the cases of commands which explicitly halt the test or some part of it). The commands, the messages, and their meanings are described next.

B <COL>* BEGIN (RESTART) TEST

This command instructs the processor to restart the test. It resets the other processor and the test starts from the beginning with the same mode of operation. The error log and the counters are not modified. (The only exception are the counters associated with time, such as the JIFFY counter, or the 10 minutes counter, which are reset each time a processor starts the test.)

C <COL>* CLEAR LOG

The current error log is cleared and can except new errors. The various counters are also cleared.

E <COL>* EXIT

The test is aborted and the conversation is restarted. The error log and the counters are kept; they can be printed in the conversation and even be kept for the continuation of the test. The other processor must stop its testing; otherwise, it will probably detect errors. It can be halted (manually or via the on-line command "H"), or restart the conversation. The effect of the "E" command is identical to that of using RESET-ATTENTION.

F <COL>* FORCE ERROR

The next word transmitted by this processor is in error (and should be detected by the other processor).

H <COL>* HALT

This processor stops all activities (even all timers are frozen), and waits for the next on-line command to be typed by the user.

K <COL>* KEEP ERRORS (ON/OFF)

This command complements the current status of the program switch described in step 11 in the previous section. The new status of the switch is printed.

N <COL>* NOISE (ON/OFF)

This command complements the current status of the program switch described in step 9 in the previous section. The new status of the switch is printed.

O <COL>* OPEN: aaaa= rrrr wwww

The open command allows the user to read the contents of any location and optionally to modify it. aaaa (typed by the user) is a four digit hexadecimal address of the location to be read. rrrr is the contents of the opened location. wwww (typed by the user) is an optional new value to be stored in the opened location. Any non-hexadecimal character typed by the user, aborts the command. Thus, the space character can be used to abort the command if so desired.

P <COL>* PRINT STATUS (ON/OFF)

This command complements the current status of the program switch described in step 10 in the previous section. The new status of the switch is printed.

R <COL># RX (ON/OFF)

This command complements the current status of the program switch described in step 7 in the previous section. The new status of the switch is printed. The command activates or disactivates the receiver part of the program. If the receiver is activated by the command, it expects the first word in the selected data table (i.e., the transmitter of the other processor must also be activated shortly after this point). The initial receive timeout is set to one minute, and after the arrival of the first word, to 10 seconds.

S <COL># SUMMARY:

This command causes printing of the current status and the current error log (if it is not empty). The format of the printed tables is given in a later section. The cahracter "X" or the character ^0 (control 0) can be typed by the user to abort the printing.

T <COL># TX (ON/OFF)

This command complements the current status of the program switch described in step 5 in the previous section. The new status of the switch is printed. If the command activates the transmitter, it starts from the beginning of the selected data table.

X or ^0 (control 0)

These characters can be typed by the user during the short interruptions between printed lines of the summary; they abort the printing.

14.4 The Summary Format

The summary is printed when the on-line command "S" is typed during the test, during the conversation (at step 2), or after some error has been detected and the "print status on error" switch is ON. In the later case only the first part of the summary (the current status) is printed.

The printing can be aborted by typing the character "X" or the character ^O (control O) in the short interruptions between printed lines.

The summary consists of two parts: the current status, and the error log. The error log is not printed if no data errors are recorded in the log. The format of the two parts is described now.

CURRENT STATUS:

```
# RESTARTS       nnnn
# RX GOOD WORDS  nnnn
# TX WORDS       nnnn
# RX BAD WORDS   nnnn
# RX TIMEOUTS    nnnn
# TX TIMEOUTS    nnnn
```

The meaning of the above counters is:

#_RESTARTS- The number of times this processor has been restarted by a power restart interrupt. The counter can be cleared in step 12 of the conversation.

#_RX_GOOD_WORDS- The number of good words received by this processor. This counter can be cleared in step 12 of the conversation. #_TX_WORDS- The number of words transmitted by this processor. This counter can be cleared in step 12 of the conversation.

#_RX_BAD_WORDS- The number of bad words received

by this processor. (A bad word is a word which doesn't match the expected one.) This counter can be cleared in step 12 of the conversation.

#_RX_TIMEOUTS- The number of receive timeouts detected by this processor. This counter can be cleared in step 12 of the conversation. # TX TIMEOUTS- The number of receive timeouts. Note that the RED processor has no direct way of knowing whether the BLACK processor read its data. Transmit timeout in this case represents the case in which the prep signal didn't return to 0 within 10 seconds from its activation. This counter can be cleared in step 12 of the conversation.

The format of the error log is defined below.

ERROR LOG LAST -1 -2 -3 -4

Below this line, data associated with up to five errors is printed in four seperate groups. The latest entry in the error log is printed under the word LAST, the previous entry under -1, etc. The first group describes the bad (recieved) word:

BAD WORD:

This title precedes the bad word information.

NOT MASKED DATA-

The data as read from the path interface (KGB or KGR).

ABOVE DELAYED-

The data as read from the path interface about 100 micro-seconds after its previous reading (described in the above line).

MASKED DATA-

The rightmost 8/10 bits of the data read from the interface.

MASKED TABLE-

The rightmost 8/10 bits of the expected word (read from the selected data table).

STATUS & CTRL-

The contents of the status and control register of the path interface after the bad word has been received.

JIFFY-

The contents of the JIFFY counter when the bad word has been received.

PASS CNT-

The contents of a program pass counter when the error has been received. This counter counts the number of times the processor passes through its main loop consisting of: checking whether a word an be transmitted, checking whether a word has been arrived, polling the TTY, and updating the dconsole lights. This counter is reset each time a good word is received.

TABLE DATA-

The (not masked) contents of the expected word in the data table.

TABLE OFFSET-

An offset to the expected word in the data table.

TABLE ADDR- The absolute address of the expected word in the data table.

LAST OK TABLE- The (not masked) contents of the data table word that matched the last good received word.

The second group describes the word received immediately after the bad word. If no word is received within one second, then all the following items are ^HFFFF.

BAD WORD +1:

This title precedes the information associated with the word arrived after the bad word.

NOT MASKED DATA-

The meaning is analogous to that of the bad word.

STATUS & CTRL-

The meaning is analogous to that of the bad word.

JIFFY-

The meaning is analogous to that of the bad word.

LOOP CNT-

The contents of a counter counting the number of times the processor loops through a loop whose length is approximately 20 micro-seconds. It represents the allapsed time between the bad word and the following word.

The third group describes the second word received after the bad word. The descriptions of the items are analogous to those of the previous group. The information follows the title:

BAD WORD +2:

The fourth group of the error log contains several of the global counters at the time the bad word was received. Most of the counters have already been described in the "current status" section.

COUNTERS:

This title precedes the fourth group.

RX GOOD WORDS- The total number of good words received by this processor so far.

SAME FROM ERR- The number of good words received by this processor from the previous error (if any) to the error described in this entry of the error log.

RX BAD WORDS- The total number of bad words received by this processor including the error described in this entry of the error log.

RX TIMEOUTS- The total number of receive timeouts up to the time of this error.

TX TIMEOUTS- The total number of transmit timeouts up to the time of this error.

14.5 The Console Lights

The console lights display information that shows how the test progresses. The right part of both lights is associated with the transmission and the left part with the reception. The contents of the address lights from right to left is:

BITS_0-3: Bits 0-3 of the transmit timeout counter.
BITS_4-7: Bits 2-5 (red) or 8-11 (black) of the transmitted words counter. The bits have been selected such that the incrementing can be observed.
BITS_8-11: Bits 0-3 of the sum of the number of received bad words and the number of receive timeouts.
BITS_12-15: Bits 8-11 (red) or 2-5 (black) of the received words counter. The bits have been selected such that the incrementing can be observed.

The contents of the data lights from right to left is:

BITS_0-7: Bits 0-7 of the last transmitted word.
BITS_8-15: Bits 0-7 of the last received word.

If the user wants to use the console he first has to press ATTENTION (without RESET), and this stops the displaying. An additional pressing of the ATTENTION button restores the displaying.

14.6 The Error Messages

After most of the error types the processor prints some message, then prints the current status (if the "print status on error" is ON), then prints "WAITING ...", and finally enters the waiting state in which the on-line commands are recognized. The processor leaves the waiting state after a period of approximately 3.5 minutes (during which the whole summary can be printed) and restarts the test by resetting the other processor, or earlier, in response to a user command (such as "B" or "E"), or a power restart interrupt (which causes resumption of the test). The error messages that match the above description are:

<BLK># RED DIDN'T READ IN 10 SEC

This message is printed by the BLACK processor when a transmit timeout occurrs.

<RED># PREP ONESHOT STUCK ON

The message is printed by the RED processor when the prep signal does not return to 0 within 10 seconds. This is considered as a transmit timeout by the program.

<COL># RX TIMEOUT

The message is printed when a recieve timeout occurs.

<COL># DATA ERROR

The message is printed when a data error is detected by the processor.

<COL>* STACK FULL

The message is printed when the processor tries to push a new entry to the error log while it is full (with information about 5 errors). The information about the last error is lost and the error log is not modified.

<COL>* UNDEFINED INTERRUPT ON LEVEL 4. RESTARTING ...

After the occurrence of an undefined interrupt on level 4 the processor immediately restarts the test.

UNEXPECTED QUIT AT: aaaa

The address of the instruction that caused the unexpected quit is printed and the processor halts.

15 PWRTST

15.1 Operating Instructions

Loading the paper tape will automatically start the program. Jiffies (60 Hz. interrupts) will be counted. The operator should then repeatedly cycle the power off and on. The program will count the time between the receipt of the interrupt and the final loss of power. Upon power up, the program will self start and display the time to power fail and the number of power fail/restart cycles in the operator panel lights.

Auto-reset can be checked by making sure the enable switch (on the BCU) is on and then halting the processor. A power restart interrupt should occur in less than two seconds. Pressing RESET then ATTN on the operator panel or starting the program at BEGIN will restart the program after resetting the counters.

15.2 Parameters

If the operator panel is not located at FF80 (hex), change appropriate locations as per the following chart:

Location(Hex)	Default Value(Hex)	Function
1600		BEGIN
1684	FF80	Control Panel
1686	FF80	Control Panel

15.3 Indications and Errors

The address register on the operator panel contains the power fail count in the high order byte and the restart count in the low order byte. These counts should always be equal except for auto-resets (which only increment restarts).

The data register contains a count of jiffies in the high order four bits and should be observed to be always incrementing and the count of the time elapsed between the receipt of the last power fail interrupt and the actual loss of power in the low order 12 bits. Each count represents about 7 us. and the lag time is approximately 30 us. low due to delay in starting the count. This displayed count should always be greater than 150 hex.

There are four detected error conditions. Each causes the program to halt at location ERROR. The cause of the error can be determined from the contents of Register 1 as follows:

Value	Error
0	No module number on interrupt
1	QUIT
2	ILLOP
>7	High order bits set in module number on interrupt

For QUITs and ILLOPs the cause may be determined by examining locations 28-2C(hex) and 20-24(hex), respectively.

15.4 Algorithm

Level 4 interrupts are enabled. Upon an interrupt, the device code is examined for validity. Jiffies are counted. Upon power fail, the time to power loss is measured by counting instructions in a loop. These counts are then displayed after the power restore.

16 RLDCHK

16.1 Loading And Starting The Program

Before loading the program all busses should have power turned on. The bus reset timers of all busses should be in the OFF (up) position.

First PSTOP may be used to halt all processors. Then DDT (with starting address at ^h2000) may be loaded if the operator wants to use it later (the RLDCHK program does not need DDT for its operation). If DDT is loaded, be sure to protect that page from being used as a transfer buffer. Then the program is loaded and starts automatically. To restart the program from DDT or from the console the address ^h100 should be used. RESET-ATTENTION can be used to restart the program.

16.2 The Interaction With The User

The program starts by interacting with the user, which has to define the program's mode of operation. The answers to the questions are:

Y- yes, N- no, or E- exit (to the end of the program).

Each of the above characters must be followed by a carriage return. If an illegal pair of characters is typed by the user, the program prints the string "? (Y/CR,N,E)" and waits for a legal response.

The program interprets a single carriage-return as the character "Y" followed by a carriage-return.

The messages and questions printed by the program are described next.

1. RLDCHK REV XX-XX

The program name is the first message printed by the program, with the date of the progams versions release.

2. SINGLE BUS (Y/CR,N,E)

The program will operate in either enviornment, so the user must tell it how to proceed.

3. CHANGES? (Y/CR,N,E)

If the user does not want to make any change in the program's mode of operation, then the character N should be typed. If this is the first iteration of the program, the answer N denotes the default mode, which is described later.

If there are no changes, the program starts activating the devices according to the current mode of operation. The format of the printed tables is described in the next section.

If there are changes, the conversation continues in step 4.

4. HARD COPY? (Y/CR,N,E)

If the user replies negatively, implying a CRT device, then he should type any character to cause the printing of each table.

If the user replies positively, all tables are printed without pauses between tables. RESET-ATTENTION can be used to abort printing in this case and to restart the program. If the operator replies with E, then step 1 is repeated.

5. I/O DEVICES, CHANGES? (Y/CR,N,E)

. The user has to reply whether there is a change in the list of I/O devices to be tested. If the answer is Y, then message 6 is printed next. If the answer is N message 11 is printed. The default list of I/O devices contains all modem/RLD I/O devices.

6. XPATCHED? (Y/CR,N,E)

If the I/O devices should be tested, internally cross-patched, the answer Y should be given (this is the default). If the answer is N, then the devices are tested, externally looped. A looping plug can be used in this case to enable the test.

7. TEST ALL I/O DEVICES? (Y/CR,N,E)

If all devices are to be tested, then the answer should be Y (this is the default for the first iteration). If the answer is N, then the current table of I/O devices to be tested is cleared, and the user has to type up to 5 device addresses (e.g. E1A0, F1B0) as directed by the program in the iterated sequence of steps: 8-10.

MORE? (Y/CR,N,E)

The user has to answer if more entries are to be added to the currently discussed table. If the answer is Y, the program prints message 9. If the answer is N, then the table is complete and the conversation continues. This message is given even at the start of building the table.

It should be noted that the current table is cleared before the sequence of steps 8-10 is started, and therefore the user has to type the complete table (and not only the modifications).

9. ADDRESS=

Following this message the user has to type a 4 digit hexadecimal address (including leading 0's) as the next entry to the currently open table.

After 4 characters have been typed by the user, message 10 is printed by the program.

10. CONFIRM? (Y/CR,N,E)

The user is required to confirm an address typed as an answer to the previous message. If the answer is N, then question 8 is repeated, and the typed address is ignored.

If the answer is Y, then the address is added to the table. Then, if there is still room in the table, question 8 is printed, else the sequence of steps: 8-10 is terminated and the conversation continues.

11. MEM BUSSES, CHANGES? (Y/CR,N,E)

The user has to reply whether there is a change in the table of memory busses included in the test. If the answer is Y, then message 12 is printed. If the answer is N, message 13 is printed and the table is not changed.

12. TEST ALL MEMORY BUSSES? (Y/CR,N,E)

If all memory busses should take part in the test, then the answer should be Y (this is the default for the first iteration).

If the answer is N, then the current table of memory busses to be checked is cleared, and the user has to type map addresses of up to 3 pages (e.g. 0000, 0200) which are interpreted as starting addresses of the memory busses to be checked. A memory bus is a contigous sequence of (^h2000 byte) memory pages.

The addresses are typed by the user as directed by the program in the iterated sequence of steps: 8-10.

After this step is completed, question 13 is printed.

13. IGNORE-I/O BUSSES, CHANGES? (Y/CR,N,E)

The user has to reply whether there is a change in the list of I/O busses to be ignored in the test. The list is initially empty, i.e. no I/O bus is initially ignored. The I/O busses address ranges are: E000-E7F0, E800-EFF0, F000-F7F0, and F800-FBF0. The first ^h80 bytes are the BCH area, and no devices are searched there.

If the answer is Y, then the table of I/O busses to be ignored is cleared, and the user has to fill it by typing up to 3 I/O busses starting addresses (e.g. E000, E800). The addresses are typed by the user as directed by the program in the iterated sequence of steps: 8-10. If the answer is N, then the table is not changed.

After this step is completed, question 14 is printed.

14. IGNORE-PAGES, CHANGES? (Y/CR,N,E)

The user has to reply whether there is a change in the table of memory pages to be ignored during the test. The table is initially empty, i.e. no memory pages are initially ignored.

If the answer is Y, then the table of memory pages to be ignored is cleared, and the user has to fill it by typing up to 8 memory busses starting addresses (e.g. 0000, 0200). The addresses are typed by the user as directed by the program in the iterated sequence of steps: 8-10. If the answer is N, then the table is not changed.

After this step is completed, question 15 is printed.

15. SLOW DEVICES? (Y/CR,N,E)

The user has to reply whether there are slow devices in the system, for which the transfer can last more than 1 second. A 50 Kb modem should complete the transfer of the 0F00 words in approximately 1/3 second. The answer to this question determines the length of the period to which the program timer is set.

The default for the first iteration is 1 second, i.e. no slow devices. If unexpected results are obtained, it would be a good practice to rerun the program with the longer timer set.

The long timer is set to about 2.5 seconds. If longer timouts are required the timer loop can be increased.

16. TEST GOOD PAGES ONLY ONCE? (Y/CR,N,E)

If a positive reply is given, then each transfer (sucessful or unsuccessful) is performed once; otherwise, the user has to specify a number in the next step. The default is to test each good page only once.

17. TIMES=

Following this message, the user has to type a non-zero four digit hexadecimal number specifying how many times each sucessful transfer should be repeated. If the user asks for n times, each transfer between a common memory page and a device will be performed n times if all of them are successful, or until the first error occurs.

18. DO ONE TIME ABORT CONDITION TEST?

A positive reply will cause the program to use the first available buffer page and send the RLD packets which have been changed to cause the RLD to abort it's cycle. The tests show that a card which works with "good" data, will also detect and react correctly to "bad" data. There are seven distint errors introduced. The tests are skipped on a negative response.

19. DMA TRANSFERS

This title is printed before the actual transfers starts. It is followed by the tables, as described in the next section.

20. RESTART? (Y/CR,N,E)

The question is printed when all selected devices have been checked. Y causes restarting of the program with the current mode of operation which may be changed by the user, N causes restarting at step 3, CHANGES. E causes a transfer to the beginning of the program at step 1

21. UNEXPECTED QUIT AT: XXXX

The message is printed for the obvious reason. The program should be restarted and if the message is consistent then the program listing must be used to analyze the cause of the quit.

16.3 Printing The DMA Transfer Results

After the program's mode of operation is set as described in the previous section, the devices are activated one at a time. For each device the transfer is done sequentially for each selected memory page. The transfers' results are printed in tables, one per device.

The format of each table is as follows:

```
MULTIBUS FORMAT
---------------
DEVICE ADDR: XXXX   TYPE: YY   NAME: NNN   TPID: TT   RPID: RR   NUM: ZZ
-------------------------------------------------------------------------

MAP  RLD MTX MRX TPI RPI DAT NELB MQIT DQIT TQIT RQIT TERR RERR TSRS CNT
                                                                  (TST#)

VVVV  F   F   F   F   F   F   F   F   F   F   F   F   F   F   ttrr cccc
  .
  .
  .

WWWW  F   F   F   F   F   F   F   F   F   F   F   F   F   F   ttrr cccc

SINGLE BUS FORMAT ---------------- DEVICE ADDR: XXXX
TYPE: YY   NAME: NNN   TPID: TT   RPID: RR   NUM: ZZ
---------
-------------------------------------------------------------------

PAGE  RLD MTX MRX TPI RPI DAT NELB MQIT DQIT TQIT  RQIT
TERR              RERR              TSRS              CNT

(TST#)

VVVV   F   F   F   F   F   F   F   F   F   F   F   F
F   F   ttrr
  .
  .
  .

WWWW   F   F   F   F   F   F   F   F   F   F   F   F
F   F   ttrr
```

The header of each table consists of the device address, the device type, a three character mnemonic name, the transmit PID level, the receive PID level, and the device number (byte 2 in the device registers block).

Each row summarizes the transfer's results for one memory page, whose map address is denoted by VVVV or WWWW. A blank line is printed between memory busses.

The entry in most of the table fields is a single digit denoted by F. F can be one of the characters: "1", "0", or "?".

1- means that the event associated with the corresponding field has happened during the transfer.
0- means that the event has not happened.
?- means that the program does not have the necessary imformation (e.g. because of a quit), or the data was not correct.

The meaning of each event is as follows:

RLD- The data packets received by the RLD were correctly decoded and the expected data was found in memory.

MTX- The transmit part of the device had terminated the transfer before the end of the program timer period was reached (i.e. bit 13 in word 6 of the device registers block was 0).

MRX- The receive part of the device had terminated the transfer before the end of the program timer period was reached (i.e. bit 13 of word 3 of the device registers block was 0).

TPI- The transmit PID level was written into the correct PID.

RPI- The receive PID level was written into the correct PID.

DAT- The expected data was found in the receive buffer of the checked memory page.

NELB- The bit indicating no error and last buffer was on (i.e. bit 15 in word 2 of the device registers block was 1).

HQIT- The processor got a quit from the checked memory page. DQIT- The processor got a quit from one of the device registers.

TQIT- The device transmit quit bit was set (i.e. bit 8 in word 6 of the device registers block was 1).

RQIT- The device receive quit bit was set (i.e. bit 8 in word 3 of the device registers block was 1).

TERR- The device transmit error bit was set (i.e. bit 0 in word 5 of the device registers block was 1).

RERR- The device receive error bit was set (i.e. bit 0 in word 2 of the device registers block was 1).

TSRS- The last field in the table contains the hexadecimal contents of two status bytes in the device registers block: the transmit status (i.e. the contents of byte c in the device registers block), and the receive status (i.e. the contents of byte 6 in the device registers block).

CNT- This field is printed only when a not hard copy terminal is used. It specifies the value of the counter which counts the transfers to the page described in the printed line.

TST#- This field is printed instead of CNT for the one time tests It specifies the test number which was being performed for that cycle. The numbers refer the the seven ways that the packet data can be corrupted and cause the RLD card to abort. They are:

1. a non-doubled data bit is introduced into the packet
2. one bit of the packet checksum has been changed to have it not reflect the actual data
3. the RLD is asked to put data in a non exisant memory (device) location. Location FCC0 was used.

4-7 Each test here turns off one of the four padding bits in a packet. The card should be strapped to expect all four to be ones. 4=LSbit 7=MSbit

16.4 Interpretation Of The Results

A successful transfer for a common memory page is reflected in the printed table by a line whose first 6 fields (RLD-MELB) are 1, and the next 7 fields (MQIT-RERR) are 0. The contents of the last 2 fields (TSTT-RSTT) is device dependant.

A successful transfer for a common memory bus is reflected in the printed table by a group of identical consequtive lines, each representing a successful transfer.

Note that the program tries to transfer data for every memory bus (unless the user has selected another mode of operation), and therefore if a device is not connected to some memory bus (as in the case of combined M/I busses), then it will be reflected in the printed table as an unsuccessful transfer.

17 RTCTST

17.1 Operating Instructions

Load the tape and the program should start. If it halts, see the below parameters section. DDT-40 is loaded automatically in this version of RTCTST. The locations can be changed either through the control panel or via DDT-40. Pressing RESET then ATTN on the operator panel will cause a restart, or else start the processor at BEGIN.

17.2 Parameters

Location(Hex)	Default Value(Hex)	Function
1700		BEGIN
		HOUR
		MIN.
		SEC.
	E000	Pid Address
	E006	RTC Address
	FF80	Operator Panel

Note: A future REV will give the correct addresses for this diagnostic.

17.3 Indications and Errors

The clock count register will be displayed in the data (lower) lights. It should increment every 100us. In the address lights the left byte will increment for each "slow" pseudo-interrupt detected; this should be every 25.6ms. Similarly "fast" pseudo-interrupts are counted in the right byte; these should be every 1.6ms.

Hours, minutes, and seconds since program initialization (based on line frequency interrupt) are tallied in locations HOUR, MIN, and SEC respectively.

Errors cause halts in program execution as follows:

OK-2: The counter incremented by an illegal value. New value in R3, difference in R1.

SLOI-2: Unexpected PID level.

LINE-2: Unexpected level 4 interrupt.

U1: Unexpected level 1 interrupt.

ABRTR: Unexpected QUIT

UNIMP: Unexpected ILLOP

17.4 Algorithm

The clock counter is repeatedly read, making sure that the increment from the last read is zero, one, or two. After each clock read, the PID is read to check for clock pseudo-interrupts, which are counted. The line frequency interrupt independently counts real time.

18 TTYTST

18.1 Operating Instructions

Load the paper tape and the processor should halt. If the address of the PSB to be tested is other than FC10 (hex), enter the correct address in the proper location as per the information in the parameter section.
The program is started by pushing RUN on the operator panel. The printable ASCII character set will be output indefinitely. Pressing RESET, then ATTN will stop output and proceed to the echo test. Now any string typed in will be echoed back after a carriage return is typed. A null string (just a carriage return) will halt the program; pressing RESET then ATTN returns control to DDT.

18.2 Parameters

LOCATION (HEX)	DEFAULT VALUE(HEX)	FUNCTION
Register #1	FC10	TTY Addr.
Register #2	0 *	Padding

* If TI700 enter "6" for padding

Register #1= FFX2 X=0,2,4,6 for processors 0,1,2,3
Register #2= FFX4 respectively

18.3 Indications and Errors

All errors must be detected by the user.

18.4 Algorithm

During the output test characters with ASCII values 240-337 (octal) are output repeatedly. The echo test accepts up to 100 characters and then retypes them. The PSB is operated in the PDT (polling) mode.