9100/9105 Version 6.0 Software Release Customer Information

Version 6.0 of the 9100A system software is a major software release. The TL/1 program interpreter has been replaced with a faster, memory-efficient compiled TL/1 runtime system. The 6.0 release also contains new features, performance enhancements and fixes for reported bugs. These notes provide a brief description of new features in 6.0, instructions for installing version 6.0 software, a list of bugs fixed since the 5.0 software release, and a list of bugs known to exist in version 6.0.

> ***IMPORTANT NOTE:***
>
> ***Version 6.0 is not fully-compatible with prior software releases. Please read section 2, "Incompatibilities Between Version 6.0 and 5.0" before upgrading to Version 6.0.***

P/N 899658
May 1991 Rev. 1,6/91

Contents

This document accompanies version 6.0 of the 9100/9 105 operating software. The document is divided into sections:

SECTION		PAGE
1.	**New Features of Version 6.0 .....**	**5**
	This section gives a brief description of the new features in the **6.0** software release.	
2.	**Incompatibilities Between Version 6.0 and 5.0 ...**	**11**
	This section describes the incompatibilities between this software release and the previous release.	
3.	**Version 6.0 Upgrade Kit Contents. .....**	**14**
	This section describes the items you should have received with your software upgrade kit.	
4.	**installing New System Software. .....**	**16**
	This section descsribes how to update your unit from version **3.0** or later software to version **6.0.**	
5.	**Bugs Fixed in Version 6.0.. .....**	**19**
	T'his section describes the bugs fixed between version 5.0 and 6.0.	
6.	**Known Bugs in Version 6.0. .....**	**30**
	This section describes the bugs that are known to exist in version **6.0.**	

IMPORTANT NOTE:

Version 6.0 is not fully-compatible with prior software releases. Please read section 2 before upgrading to Version 6.0.

1. New Features of Version 6.0

1.1. Compiled TL/1 Runtime System

The TL/1 interpreter has been replaced with a compiled TL/1 runtime system. Larger programs can now be executed because the compiler-based runtime system is more memory-efficient than the interpreter was. In general, it is also faster than the interpreter, resulting in decreased board test times The compiled nmtime system also allows you to distribute compiled object programs which can be executed but cannot be modified.

When you execute a TL/1 program with the 6.0 release, the new TL/1 runtime system will automatically compile the program as it is read off the disk. The program is compiled directly into memory and the resulting object program is executed. This process is very fast and programs actually begin execution faster than they did under the TL/1 interpreter . Because the compilation happens automatically, your procedures for executing programs do not have to change.

However, using this method, the programs are compiled each time they are executed. For the best speed performance, programs can be pm-compiled from the editor (see section 1.2. "TL/1 Program Compiler"). Using this method, programs are compiled only once, and the executable object program is saved on disk. When you execute a pre-compiled program, the TL/1 runtime system simply loads the executable object program. This is the fastest method of executing programs.

Once a program has been pre-compiled, the TL/1 runtime stystem automatically maintains the compiled object file. If both the TL/1 source program and compiled object file are present when a program is executed, TL/1 compares the two files to make sure that the object file was generated from the source program. If they do not match (e.g. because the source program has been modified), the TL/1 runtime system will automatically recompile the program before executing it The new object program will be saved on disk.

IMPORTANT NOTE

Every effort has been made to insure identical behavior between TL/l programs executed under the interpreter and the compiled run-time system. However, a few differences were unavoidable. For example, because programs execute faster, software loops that were being used to detect time-out (e.g. loop for i = 1 to 1000) may need to be changed. Please read section 2, "Incompatibilities Between Version 6.0 and 5.0" for a list of the known-differences between the interpreter and the compiled run-time system. That section also includes a recommendation that you pre-compile all programs when you install version 6.0.

1.2. TL/1 Program Compiler

TL/1 programs can be compiled from the editor before they are executed. This pre-compilation will result in faster startup when a program is executed and lets you find compilation errors in the editor rather then when the program is executed.

Each of the directories that contains TL/1 programs now contains a COMPILE softkey. To compile a program, edit a UUT. PROGLIB or POD. Press the COMPILE softkey. select the TL/1 compiler (using the field-select key) and then enter the name of the program to be compiled. The wildcar dcharacter '*' can be used to compile multiple programs. Next, use the licld-select key to select YES in response to the prompt "USE CURRENT TL/l COMPILER OPTIONS", and then press the Return key to start the compile. If the program is successfully compiled, an executable object program (OBJPROG) will be saved on disk. The compilation can be aborted at any time by pressing the Quit key.

During compilation, error messages are displayed on the monitor screen. The T'L/l compiler has an option that allows you to save these messages in a text file. It also has options that control the type of warning messages that are issued. Warning messages are precautionary and attempt to identify features of

the TL/l program that could be bugs or are unnecessary. Refer to section 3.1.9 “Compiling a TL/l Program” in the Programmer’s ***Manual*** for a complete description of the-se options and how to change them.

1.3. TL/1 Program Debugger Improvements

The TL/1 debugger has been improved to allow debugging across multiple programs. This includes the ability to display called programs, set breakpoints in a called program, and step into a called program. Another improvement allows access to array variable elements. For fastest debugger start-up, pre-compile the TL/1 programs to be debugged.

A new VIEW softkey prompts for a program name and then displays the program. The STEP softkey now steps into a called program and automatically displays the called program. The BREAK softkey can now set breakpoints in any program that is displayed.

1.4. TL/1 Check

The TL/1 program checker is invoked when a program is being edited and the CHECK softkey is pressed. CHECK has been improved to detect many errors which previously were not found until program execution. CHECK now identifies the same errors that the new TL/1 compiler will identify, and has the effect of embedding compiler error messages into the program.

To check a program, edit the program and press the CHECK softkey. Use the field-select key to select YES in response to the prompt “USE CURRENT TL/1 COMPILER OPTIONS” and then press the Return key.

The program checker has options that control the types of warning messages that are issued. Warning messages are precautionary and attempt to identify features of the TL/1 program that could be bugs or are unnecessary. Refer to section 3.1.6 “Using the CHECK Function” in the ***Programmer’s Manual*** for a complete description of these options and how to change them.

1.5. New TL/1 Function: diagnoserom

The diagnoserom function provides access to the existing ROM diagnostics. This function can be used with your customized ROM tests or with Pod Quick Tests to provide diagnostics and fault conditions which are consistent with those of the testromfull ROM test.

Usage: diagnoserom (addr, upto. mask, addrstep)

- or -

diagnoserom addr <addr>, upto <upto> [, mask <mask>], addrstep <addrstep>

1.6. New TL/1 Function: sysinfo

The sysinfo function returns information about the system. Currently, the system software version or the model name can be obtained.

Usage: version = sysinfo get "/system/version"

version = sysinfo get "/system/model"
(returns one of the following: 9100A 9105A, 9100FT or 9105FT)

1.7. GFI Don't Care Logic Levels

GFI and UFI determine if a node is good by comparing a measured response with the expected response defined in a response file. Prior to 6.0, if an asynchronous level history or clocked level history was used as part of a node's expected response, an exact match on all three states (high, low, invalid) was required in order for the node to be good. It is now possible to specify one or more don't cure states in an expected level history. A ***don't cure*** state is ignored during the comparison of the expected and measured level histories.

To specify a don'r care level, edit a response file and enter a '?' character in the asynchronous or clocked level fields. The '?' is interpreted based on its position in the field. Each field is three columns wide. The columns represent (from left to right) high, tristate and low. Insert the *?' in the leftmost column to mean ***don't cure*** on high, in the middle column to mean ***don'f cure*** on tristate and **in the rightmost column to mean *don't cure*** on low. For example, "1?0" means don't care on tristate "??0" means ***don't cure*** on high and tristate.

Compatibility response files and databases that contain don't ***cure*** levels cannot be used with 5.0 or earlier software.

1 .a. UUT Directories Can Contain Parts

A UUT directory can now **contain part descriptions. Prior to 6.0,** part descriptions could only be stored in the Part Library (PARTLIB). This feature allows you to locally redefine parts as **required** by a specific UUT. The GFI and UFI compiler will use the following search path to resolve part references: First they will look in the UUT. If the part is not found there, they will look in the part library. If it is not found there, an error message will be displayed.

Compatibility: UUTs that are created with 6.0 or later software and contain parts can be used with 5.0 or earlier software (however, the parts will not ap pear in the UUT directory and cannot be accessed).

1.9. Editor Line Syntax Checking

The line syntax checking can now be disabled for TL/l programs and node lists.

To disable line syntax checking, edit a program or node list and simultaneously press the Shift key and the CHECK softkey. This key combination will toggle the line checker on and off. The current mode is displayed in the status line at the top of the CRT. The line checking mode is shared between programs and node lists and is saved across edit sessions. The current line must be correct before the line checking mode can be changed.

Comnatibility: programs and node lists that were saved with line syntax errors can be edited with 5.0 or earlier software.

1 .10. Terminal Emulator Improvements

The terminal emulator has been improved to support transferring TL/l programs and node lists. Prior to 6.0, only text files could be transferred. To send or receive a file, enter the file name and then use the field-selectkey to select the file type (TEXT or PROGRAM or NODE).

1 11. Editor Long Directory Style

The STYLE softkey controls the way that directories are displayed when a USERDISK. UUT, LIBRARY or POD is edited. The LONG style now displays the file size in addition to the modification date and time. The directory style will be retained during an edit session and will be reset to BRIEF each time the editor is started.

When a USERDISK, UUT LIBRARY or POD is copied to a port, the directory will be sent in the current style (BRIEF or LONG).

1.12. Save System/Calibration Settings

The front panel SETUP - SAVE menu can be used to save the current system settings or calibration settings on disk. These settings can be saved in the current userdisk, or in a specified UUT. Prior to 6.0, an error would occur if you tried to save the settings in a UUT that did not exist. Now, the UUT will automatically be created if it does not exist.

1 .13. E-Disk Changes

The front panel MAIN MENU - EDISK - LOAD menu has two new fields that allow optional loading of TL/l source programs (PROGRAM) and compiled programs (OBJPROG) into the E-disk. Prior to 6.0, TL/1 programs were always loaded into the E-disk

If all of your TL/1 programs have been compiled, the source TL/1 source programs are not needed when a program is executed. This means that you can load the E-disk faster and use less **memory** for the E-disk if you load the compiled object programs (which are required for execution), but do not load the TL/1 source programs into the E-disk.

If you specify that OBJPROGs should be loaded, and PROGRAMs should not be loaded, the system will generate a warning message if it finds that a PROGRAM needs to be compiled, or an OBJPROG needs to be recompiled (e.g. the TL/1 source program was modified, but not recompiled).

2. Incompatibilities Between Version 6.0 and 5.0

Every effort has been made to insure identical behavior between TL/1 programs executed under the interpreter and the compiled runtime system. However, some differences were unavoidable, and these differences are described below.

2.1. TL/1 Software Timing Loops

Compiled TL/1 programs execute faster than interpreted TL/1 programs. Therefore, any software timing loops in TL/1 programs may need to be modified. An example of a software timing loop appears below:

```
loop for i= 1 to 1000
end loop
```

2.2. Implicit UUT Timing

Programs may also contain implied timing that assumes a relationship between TL/1 execution speed and UUT activity. For example, a program may send a sequence of commands to an electromechanical device such as a vacuum fixture, relay or disk drive. This program may have worked properly with the 5.0 software because the device had enough time to carry out each command before it received the next command from the TL/1 program. However, with the increased execution speed of 6.0, the TL/1 program may send commands to the device before the device is ready.

2.3. TL/1 Programs Can't Be Executed Due to Compilation Errors

There are many kinds of program errors that were not found by the TL/1 interpreter until program execution. An example of such an error is assigning a string value to a numeric variable. With the interpreter, you would encounter these errors one-at-a-time when a statement with an error was executed. If these errors were in a branch of the code that had not been executed previously, you may not even be aware of these errors.

With the compiled runtime system, all of these types of errors are discovered before the program is executed, when it is compiled. This lets you be confident that fewer runtime errors will occur to stop execution of your programs.

However, because the compiled T'L./l rtmtime system examines every program statement when it compiles the program, some programs may not compile, and the new compiled runtime system cannot execute a program unless it compiles successfully.

This means that some programs that you were able to execute with the interpreter cannot be executed with the compiled runtime system until you fix all the compilation errors. Typically, these errors are very easy to fix and often involve things like misspelled variable names, incorrect arguments to functions, etc.

IMPORTANT NOTE

It is strongly recommended that you pre-compile all your TL/l programs from the editor before executing them with version 6.0. Pre-compiling *your* programs will allow you to find all the compilation errors at one time, in the editor environment where it is convenient to fix the programs.

Programs that contain compilation errors cannot be executed until the errors are fixed.

2.4. TL/1 Variable Initialization

Global string variables are now initialized to the empty string "". They were not initialized before.

The default value for floating variables has been changed to 0.0. The default value used **to be 1.234568E-123.**

2.5. TL/1 Arithmetic Overflow

TL/1 now wraps through zero on arithmetic overflow and underflow. For example, the expression **"$FFFFFFFF+1"** will result in a value of zero. The expression "0- 1" will result in a value of **$FFFFFFFF.**

Arithmetic overflow and underflow used to be detected in some cases and would result in a program error.

2.6. Response Files

Response files that use the new don'f care feature to describe asynchronous or clocked level histories cannot be used with version 5.0 or earlier.

2.7. GFI/UFI Databases

GFI and UFI databases that include ***don't cure*** levels for the asynchronous or clocked level history cannot be used with 5.0 or earlier software.

3. Version 6.0 Upgrade Kit Contents

3.1. 9100A-7200-903 (Software Support Upgrade)

This kit upgrades version 5.0 9100 or 9105 systems to version 6.0.

System Disk 1 -- Ver 6.0	Technical User's Manual (text)
System Disk 2 -- Ver 6.0	TL/1 Quick Reference Card
System Disk 3 -- Ver 6.0	IEEE Quick Reference Card
IEEE-488 Disk -- Ver 6.0	Manual Update Package
Master Userdisk 1 -- Ver 6.0	Release Notes
Master Userdisk 2 -- Ver 6.0	
Software Support Agreement Documents	

3.2. 91 OOA-7200-904 (Software Support Upgrade)

This kit upgrades version 5.0 9100 systems with programming to version 6.0.

System Disk 1 -- Ver 6.0	Technical User's Manual (text)
System Disk 2 -- Ver 6.0	Programmer's Manual (text)
System Disk 3 -- Ver 6.0	TL/1 Quick Reference Card
IEEE488 Disk -- Ver 6.0	IEEE Quick Reference Card
Programmer's Disk -- Ver 6.0	Manual Update Package
Master Userdisk 1 -- Ver 6.0	Release Notes
Master Userdisk 2 -- Ver 6.0	
Software Support Agreement Documents	

3.3. 91 OOA-903

This kit upgrades version 3.0 or later 9100 or 9105 systems to version 6.0.

System Disk 1 -- Ver 6.0	Technical User's Manual (text)
System Disk 2 -- Ver 6.0	Automated Opns Manual (text)
System Disk 3 -- Ver 6.0.	TL/1 Reference Manual (text)
IEEE-488 Disk -- Ver 6.0	TL/1 Quick Reference Card
Master Userdisk 1 -- Ver 6.0	IEEE Quick Reference Card
Master Userdisk 2 -- Ver 6.0	Release Notes
Software Support Agreement Documents	

3.4. 91WA-904

This kit upgrades version 3.0 or later 9100 systems with programming to version 6.0.

System Disk 1 -- Ver 6.0	Technical User's Manual (text)
System Disk 2 -- Ver 6.0	Automated Qpns Manual (text)
System Disk 3 -- Ver 6.0	TL/l Reference Manual (text)
IEEE-488 Disk -- Ver 6.0	Programmer's Manual (text)
Programmer's Disk -- Ver 6.0	TL/1 Quick Reference Card
Master Userdisk 1 -- Ver 6.0	IEEE Quick Reference Card
Master Userdisk 2 -- Ver 6.0	Release Notes
Software Support Agreement Documents	

4. Installing New System Software

The procedure for installing new system software depends on whether the mainframe is a 9100 or 9105.

IMPORTANT NOTE

If you are upgrading to Version 6.0, please read sections 1 and 2 of this document before installing the system software. Version 6.0 is not fully-compatible with prior software versions.

4.1. 9100 Software Installation Procedure

The following procedure should be followed to install a new version of the system software on a 9100 which is running version 3.0 or later software.

IMPORTANT NOTE

If your 9100 is running Version 2.2 or earlier software, it must be upgraded to Version 3.0 before Version 6.0 can be installed. Failure to update from 2.2 to 3.0 before installing Version 6.0 will leave your 9100A in an unusable state.

IMPORTANT NOTE

Once you have begun the installation procedure, do not restart or power down your 9100 until all disks are copied. If you do, you run the risk of leaving your 9100 in an unusable state.

(1) Copy each of the new System Disks and Optional Features Disks to the hard disk by selecting the command:

MAIN: COPY DISK FROM DR1 TO HDR

under the MAIN MENU key on the operator's interface.

If your 9100 does not have IEEE-488, you do not have to copy the IEEE Optional Feature Disk. If your 9100 does not have a programmer's station, you do not have to copy the Prograrnmcr's Optional Feature Disk.

(2) If your upgrade includes new Master User Disks, you may copy them to the 9100A hard disk by selecting the command:

MAIN: COPY DISK FROM DR1 TO HDR

under the MAIN MENU key on the operator's interface.

IMPORTANT NOTE

If you saved modified copies of any programs from the Master User Disks onto your 9100 hard disk, and you did not change their names, the modified programs will be overwritten when you load new Master User Disks.

(3) Restart your 9100 by depressing the recessed RESTART button (on the mainframe right side panel at the front), or by turning the power switch (on the back of the mainframe near the power cord) off, then on.

(4) After the system completes its self-test and loads the system software from the hard disk, a ready message appears on the operator's display.

If the ready message does not appear, see the "Power-up of a 9100A section in the ***Technical User's Manual.***

(5) ***The*** original system disks should be stored in a safe place as backups.

4.2. 9105 Software Installation Procedure

The following procedure should be followed to install a new version of the operating software on the 9105.

(1) Restart your **9105 by depressing the recessed** RESTART button (on the mainframe right side panel at the front), or by turning the power switch (on the back of the mainframe near the power cord) off, then on.

(2) After the system performs its self-test, you are prompted to insert the disk labelled "System Disk 1" into Drive 1 (the upper micro-floppy disk drive). When the disk is read, you are prompted to insert the next system disk into Drive 1. This step is repeated for each system disk.

(3) After the System Disks are read, you are asked whether there are any "Optional feature disks" to be loaded. The IEEE-488 disk is an Optional Feature disk. You do not have to load this disk if your 9105 does not have IEEE-488, or if your 9105 has IEEE-488 but you do not want to use this capability. If there are no feature disks to be loaded, press CLEAR/NO.

To load a feature disk, press ENTER/YES. Insert the disk into Drive 1 when prompted. When the disk has been read, the mainframe asks for any more feature disks. If there are additional feature disks to be loaded, press ENTER/YES. If there are no more feature disks, press CLEAR/NO.

It is a good idea to load all the feature disks at least once to verify they are readable on your machine.

(4) When the power-up procedure is completed, a ready message appears on the operator's display.

If the ready message does not appear, see the "Power-up of a 9105A" section in the ***Technical User's Manual.***

(5) Make a working copy of each disk. The originals should be stored in **a safe place as backups. To** copy the disks, follow the procedure in "Duplicating a Disk Using a 9100A/9105A in the ***Technical User's Manual.***

5. Bugs Fixed in Version 6.0

The following bugs reported in previous versions of the system software have been corrected in version 6.0. The bug descriptions are furnished for reference only; they are not needed for programs running under version 6.0 or later.

9100A-247: TL/1 sometimes loads programs more than once.

DESCRIPTION

Occasionally unusual disk activity was observed when a TL/1 program was executed in a loop. These programs were usually pod special programs.

The problem was that the program was named FOO but was invoked as foo. The operating system looks for the program on disk case insensitively, and when it finds the program, it executes it. However, it stored the program in a case sensitive symbol table, as required by the published semantics of TL/1 The next time the program was invoked, the symbol table was searched for foo. F0 0 was in the symbol table, but not foo, so the program was loaded again.

The bug is that the actual name of the program was not checked once the program was loaded.

9100A-248: TL/1 should preserve linkage & space when exercising **faults.**

DESCRIPTION

Space and linkage should be the same for each iteration of an exerciser. It isn't being newly set, so if an exerciser does a setspace(), second and subsequent iterations may perform different UUT accesses from the first iteration, which is bad.

9100A-265: Editor PASTE misbehaves if there are blanks at the end **of the line.**

DESCRIPTION

It is not possible to PASTE at the end of a program line if the line ends in one or more blanks. Instead of the pasted text being **put** at the end of the line, it is inserted before the trailing blanks, and all of the blanks remain at the end of the line.
For example, on the following line, with the cursor at the end:
print "Test line" <cursor is here>
PASTE text has this result:
print "Test line"!This is the pasted text <end of line here>

9100A-266: 9100 **cannot always tell if pod database name does not match pod type.**

DESCRIPTION

Plug in a Z80 pod (with the UUT cable in the selftest socket), and enter a POD NAME of 6802. The 9100 will load the pod database, and return the error message "enabled line causes timeout" (which is normal when the pod is in selftest). The pod database loaded now is for the 6802 pod (as evidenced by the ADDR OPTION, ENABLE line names, and POD NAME), however the pod will pass selftest and seems to work normally for other functions as well.

Selftest. at least, should be able to tell that the pod type does not match the database that is loaded, since the pod reset string includes its name. The POD NAME entered should also be checked against the actual pod installed.

This seems to happen specifically with the Z80 pod and the 6802 database. Other combinations of pod and database mismatches do give an error message if the wrong name is entered, but the mis-matched database is still loaded.

9100A-267: Debugger SHOW **and** SET **do not work for array references.**

DESCRIPTION

SHOW and SET in the debugger do not work for army references:
1) Array indices cannot be specified in SHOW or SET

2) **SHOW** and **SET can be** used with just the army name and this appears to show the lowest subscripted clement.

9100A-268: HELP in the E-DISK LOAD command can be confusing

DESCRIPTION

When choosing a UUT to load into the E-DISK, the user (as per normal) can push the HELP button **when** in the UUT **NAME** field of the E-DISK load command on the application interface, and get a list of the UUTs on the "current disk". However, if the user sets the DRIVE field in the E-DISK LOAD command to a disk different than the "current disk" (as set in the SETUP command), the command still lists out the contents of the "current disk". It would be intuitive that it should list out the contents of the disk that the user chose in the E-DISK LOAD command. This can be confusing.

9100A-269: Debugger exits when the QUIT key is pressed

DESCRIPTION

Consider the program fragmenr

```
program a
    execute large_program0
    ...
end a
```

In the debugger, if you execute program a, then press the QUIT key during the time large_program is being loaded from the disk, then ask to SHOW an undeclared name, there is a brief flurry of disk activity, then the debugger exits suddenly and the application display shows the message "OBJECT DOES NOT EXIST".

9100A-271: Hyper-Help bug with greater than 105 files.

DESCRIPTION If there are more than 105 of a given object when hyper-help is invoked to select a file from that object type, Hyper-Help beeps and displays only 105 of the files. You may then scroll past the end of the files, lose the cursor, and never get it back. The No/Clear button exits Hyper-Help correctly.

9100A-274: Problems with debugger STEP

DESCRIPTION

If you are at the last statement of a function and press STEP (F1), the debugger won't necessarily single step. It may continue.
Also, if you are at the line before a function that will raise a fault, you might think STEP (F1) would stop you on the first line of the fault handler. It doesn't.

9100A-276: 'gfi status' returns "untested" when it shouldn't

DESCRIPTION

If you perform GFI from the front panel, and then execute a TL/l program that calls 'gfi status', then 'gfi status' always returns a status of "untested". It doesn't seem to know about the pins which were tested from the front panel.

9100A-280: Displayed data truncated in **READ VIRTUAL.**

DESCRIPTION

The displayed data is truncated when READ VIRTUAL is performed from the operator's interface, if the EXTADDR and ADDR values have more than 8 digits total. For example, if the data value read is $FFFF,

READ VIRTUAL, EXTADDR 2C)00000 ADDR 0 =
FFFF

will be displayed correctly, but

READ VIRTUAL EXTADDR 2o00000 ADDR 64 =
FFF
^

is truncated at the right hand edge of the display. This means that the whole data value (of 16 bits) cannot be read by the operator.

9100A-281: Confusing error message when illegal UUT name entered

DESCRIPTION

If the UUT is inadvertently changed to something illegal, then trying to execute TEST BUS (with a9 132 pod) gives the error message:

Bad Object Name: 1st char not alphanumeric

To repeat,
1) Press TEST-BUS ENTER (and bus test runs)
2) Press EXEC.
3) Press REPEAT (which enters an "_" in the UUT field).
4) Now, press ALPHA ENTER. You get an error message.
5) Instead, Press ALPHA REPEAT. Bus Test works.
The problem is that "_" is an illegal UUT name, so an error occurs when you attempt to search for the program in uut "_".

9100A-282: **Error message doesn't display argument name.**

DESCRIPTION

In the following program, function bar is called with an illegal argument 'blah'. The error message that results is "" is not a legal argument." The name 'blah' should appear between the **quotes.**

```
program foo
    function bar
        return
    end function
    bar blah 1
end program
```

9100A-287: TL/1 exercisers not found if fault was re-faulted.

DESCRIPTION

If a built-in fault is handied by a handler, then re-raised via the refault command, TL/1 cannot find the exerciser for the fault.

9100A-289: Memory Gobbler -- Looping 9132 Pod Self Test

DESCRIPTION

Looping Selftest on a 9132 Pod eats memory in 4Kchunks. This is on a working POD, with selftest passing. I know it exists on with a 386 pod support package, but believe it to be with any support package. Testing ONLY Sync Module when bug discovered.

9100A-293: 9100 ignores illegal parameter pod fault on write virtual

DESCRIPTION

The 9100 ignores the illegal parameter fault bit when rctumcd by a 9132 pod on a write virtual operation. The 9132 sets the fault when illega1 data is written to a special address. This is intended to let the user know he wrote illegal data, but WC cannot give him the message.

9100A-295: Trailing characters after valid floating point number ignored by TL/1 isflt

DESCRIPTION

The isflt operator evalutes a string expression and returns 1 if it is a valid floating point expression, and 0 if it is not. Currently, isflt returns 1 (valid) on a particular invalid floating point expression.
Strings of the form "2.3e" or "2e" or "2E-" are considcrcd valid by isflt; however, they are treated as syntax errors when a floating point number is set to their floating point equivalents. The trailing characters after a valid floating point number are ignored by isflt. The function uses only the leading (valid) part.

9100A-297: Disk ID error after sector COPY

DESCRIPTION

If you do a sector copy (COPY /DR1 to /DR1 and immediately afterwards try to edit /DR1, you get the following error:
"Disk ID Error"

9100A-299: 9105 hangs when you try to open a file on the hard disk

DESCRIPTION

If you execute the following program on a 9105, it will hang:

```
program x
    ch = open device "/HDR/JANET", as "output"
    close channel ch
end program
```

9100A-302: IEEE remote interface needs delay between print statements

DESCRIPTION

On a talker/listener device operating in remote mode,when successive 'print on ieee_device statements arc issued from TL/1 without a sufficient time delay between the statements, the talker/listener device breaks out of remote mode before the data from the second print statement hasbeen sent, and displays

"FLUKE DIGITAL TEST STATION READY."

9100A-304: TestRAMfast **bug**

DESCRIPTION

When running RAM TEST with a data mask that is not FFFF, a RAM TEST fault message shows that data was written to bits that were zeroed out in the data mask.

9100A-305: IEEE REMOTE get/setspace bug

DESCRIF'TION

When TL/1 setspace() is used within the REMOTE interface, it does not take effect until the remote interface is exited and restarted. i.e. (from IEEE controller's point-of-view with 9 1 XX as Talk/Listen)

Controller	91XX
GoToRemote	<fine...>
setspace (space 3)	<fine...>
read (adclr 0)	<reads at address 0 in default address space (not 3)>
GoToLocal	<fine...>
GoToRemote	**<fine...>**
read (addr 0)	<reads at address 0 in address space 3!>

9100A-306: IEEE REMOTE cannot podsetup()

DESCRIPTION

Because of the syntax of podsetup commands, they cannot be called through the remote interface.
For example, the TL/1 call:

podsetup 'report intr' "off

has no representation in the remote message syntax.

The fix will allow the following invocation syntax:

podsetup ('report intr' "off")

This really represents a shortcoming of the implcmcntation because it currently does not allow any spaces in identifier names separated by single quotes, whereas TL/1 docs allow them.

This problem also shows up where we have a TL/1 program with a space in the name, such as 'test program'(). This related problem should also be fixed. This fix will allow the following:

'test program' (slotname value)

9100A-309: TL/1 **hangs up when you use** 'remove menu' command

DESCRIPTION

If the following TL/1 program lines are executed, it will lock up the system

```
define menu "mx-I ", label "First Entry"
define menu "mx-2". label "Second Entry"
define menu "mx-3". label "Third Entry"

remove menu "mx-1" (or)
remove menu "mx-2" (or)
remove menu "mx-3"
```

Removing whole menus does work, however. For example, 'remove menu "mx"' will work.

9100A-317: **Shift-Field Select** does not work in PROGRAM info window

DESCRIPTION

The Shift-Field Select key does not change the field value for the WRITE PROTEC T field in PROGRAM info windows.

9100A-339: **Learn key won't clear error**

DESCRIPTION

When editing a response file, if the cursor is moved beyond the last node entry, and the learn key is pressed, an error message appears, "*' is not a valid ref-pin name" which is correct. If you then move the cursor back up to a valid node and press the learn softkey, it again displays the same error message, and will not allow you to learn any nodes. Note that it seems to have modified the response file by the cursor movement, even though the modified flag is not displayed. You can press the end file key, and the cursor will move to that last line you moved the cursor to. even though they are all blank.

9100A-399: softkey labels disappear after Offset Window HELP

DESCRIPTION

If you are in the response OFFSET window and press HELP- while you are being prompted, and then press QUIT to abort from the prompts, the softkey labels disappear.
To reproduce, press this sequence of keys:
OFFSET (to enter offset window)
EXEC (to execute a program)
HELP (when you are being prompted for program name -- the help window appears)
HELP (to remove the help window -- the prompt reappears)
QUIT (to abort the execute prompts)
The softkey labels are missing.

9100A-415: Stale Parameters on TEST ROM HYPER

DESCRIPTION

If you execute a TEST ROM HYPER after a TEST RAM HYPER, the RAM TEST parameters appear on the 3 line display BEFORE pressing the enter key and calling the program. The correct default parameters appear AFTER pressing the ENTER key.

9100A-427: Parameters for TEST ROM/RAM QUICK don't go away

DESCRIPTION

This is the same as 9100A-415, except for the QUICK option. The pod dependent test parameters don't get cleared off. To see this, have a pod that has the QUICK tests (eg 68000). and press:

IROMI TEST ROM QUICK [ENTER]
IRAM] TEST RAM QUICK [ENTER]
[ROM]

and notice that the parameters still exist, and the parameters that show up, are the parameters for the TEST RAM QUICK test. Also, notice that pressing [ENTER] again, brings up the correct parameters for TEST ROM QUICK, and pressing [ENTER] yet another time finally executes the test.

9100A-438: Editor SEARCH inactive if cursor in leftmost column

DESCRIPTION

In the editor, when you press the SEARCH softkey you are prompted to enter the search string. If you use the Shifted-cursor keys to edit the default search pattern, and then leave the cursor at the leftmost column of the prompt, when you press RETURN nothing happens.

9100A-449: Improper front-panel arguments displayed for TEST BUS

DESCRIPTION

I first did an "EXEC UUT TD93991 PROGRAM TRANSLATE". When this program executes, it prompts for arguments. I did not execute the program after the prompts were displayed, but instead tried to run BUS TEST, which displayed the program TRANSLATE arguments. This is similiar to 9100A-427.

9100A-481: Some Master Userdisk Part Descriptions are Incorrect

DESCRIPTION

The following parts (which are contained on Master Userdisk 2 in the parthb directory) have been corrected:

2675,74151,28S42,2016,41128,74657A,7472,SLIDESW,
74373,7413,74122 and 74123.

6. Known Bugs in Version 6.0

The following bugs are known to exist in version 6.0 software. A description of each bug is provided. A workaround is described if one is known.

9100A-249: Marking characters on a long line can mess up attributes

DESCRIPTION

This bug involves MARKing characters on the bottom line of the screen so that the screen has to scroll. The line attributes (holding) are wrong.

To reproduce:
1) Edit a text file. Put a long line in it that takes up several rows on the CRT. Position it on the CRT so that the first row of the long line is on the bottom of the CRT. (The second row of the long line is off the screen.) Put the cursor on the first row of the long line.
2) Press MARK and right arrow until the screen scrolls and you've marked a few characters on the second row of the long line.
3) Press left arrow a few times. The bolding is not removed as the cursor backs up.

NOTE: The editor does YANK/CUT the correct # of characters, it's only the bolding that is messed up.

9100A-250; Editor positions cursor improperly on long lines.

DESCRIPTION

In the example below, position the cursor on the line marked with * . Then press the <End Line> key or the <Begin Lino key followed by the <Retun> key.
Example below:

```
declare
   global string dvm = " 02" !8840 on IEEE address 2
   global numeric ieee_out
   global numeric ieee_in
   global numeric disp
end declare
* ieee_out = open device "/port1", as "output", mode "unbuffered" !CR print term
```

icee_in = open device "/portl", as "input" disp = open device "/term2", as "output"

9100A-253: various bugs using **TL/1 editor**

DESCRIPTION

Here are a couple of intermittent bugs: When scrolling up or down the screen will go blank. On the left side of the screen "<"s appear in inverse video. The up and down scrolls lock up, but one can recover by pressing the Begin File key. Also, duplicate lines are made when scrolling up and down. The last line on the bottom is duplicated and stays on screen until it is scrolled past. Also, scrolling up and down causes stuck beeper on 9100.

WORKAROUND

Press the <Begin File> key.

9100A-256: **Application interface loses cursor.**

DESCRIPTION

1. Place the front panel cursor in rightmost field of the address option display (DWORD of 80386 MEMORY DWORD).
2. Execute a program that leaves the machine in a memory space with less fields in the address option. (I/O space of 80386)
3. Try to do a read or other function from the front panel.

When you go down to the address option field, the cursor disappears

WORK AROUND

A left arrow key will make the cursor appear.

9100A-257: **Editor REPLACE can't replace ' with "** in programs

DESCRIPTION

On the following TL/1 line:

t = 't'

REPLACE ' with " and it worked for the first quote. Do a SHIFT REPLACE for the second quote and the following message is displayed:

String not closed by end of line. Missing double quote?

9100A-261: TEST BUS menu selection display disappears.

DESCRIPTION

Perform TEST BUS from the front panel. Select another operation such as READ ADDR. Press the BUS TEST key. At this point the bus test softkey labels are missing. There should be two selections displayed.

WORK AROUND

Press BUS TEST key a second time, or press the left arrow key to get the selections to be displayed.

9100A-264: Editor display problems.

DESCRIPTION

Using the up-arrow and down-arrow keys in the editor, the screen gets garbled, with just a ">" shown in the left margin. Sometimes when using the up-arrow and down-arrow keys, lines are duplicated.

WORK AROUND

This appears to be related to the problem of displaying long lines (lines which wrap around). A work-around for both cases is to hit the INFO key, which forces the screen to redisplay properly.

9100A-273: FutureNET translate bug in CADTrans.

DESCRIPTION

A customer uses a (DATA statement after the (SIG statement and before the (NAME statement. FutureNET incorrectly throws away these net names.

WORK AROUND

Remove the (DATA statement from the FutureNET netlist.

9100A-278: MORE INFORMATION LED left on incorrectly.

DESCRIPTION

When doing a TEST-BUS with the 9132A POD, and a fault message causes the MORE INFORMATION LED to come on, the LED does NOT go off when the CONT key is pressed, even though the **new message fits in the window.** The actual fault was BAD CHIP SELECT on the ROM MODULE, caused by tied high W/-R line.

9100A-279: PASSED message is not erased correctly.

DESCRIPTION

After doing a bus test on a 9132A POD, you have to press the BUS TEST key twice to erase the PASSED message. A single press of another key such as the down arrow will erase the message. I believe a single press of BUS TEST used to erase the PASSED. (Repeat never did).

9100A-283: Stack overflow error in UUT compiler.

DESCRIPTION

When compiling a UUT, an "Infinite Recursion or Stack Overflow" error occurred.

WORK AROUND

This can occur if the reflist contains several hundred entries and they are in alphabetical order. To workaround this, move the middle entry to the beginning of the reflist and recompile. For example, if the reflist is 600 lines long, move the entry at line 300 to line 5 (lines 1-4 of the reflist are title lines).

9100A-284: Cannot RESTORE Pod Name between different packag-estyles of the same POD.

DESCRIPTION

Using a 68000 Pod, with 3 different Packages covered by three different POD NAMES, I did a SETUP POD NAME to the desired POD. I then saved system settings in a UUT. When restarting a cold system, RESTORE SYSTEM SET-TINGS leaves the PODNAME at the default pod instead of changing to the stored podname.

WORK AROUND

Possibly do a manual setup pod name before restoresystcm settings??? I don't know of any bad side effects. The reverse order, setup after restore probably changes some setup info back to default???

9100A-286: Empty TL/1 loop cannot be stopped.

DESCRIPTION

The following five line program will cause the system to'lock-up" when run in the debugger. By commenting out the "close channel..." line, the "bug" will disappear and the program can be stopped by pressing the QUIT key.

```
program bug
      ochannel = open device "/term2", as "output",
                        mode "unbuffered"
      ochannell = open device "/term2/win"
      close channel ochannell
      loop
      end loop
end program
```

9100A-292: old parameter list displayed for 9132 bustest

DESCRIPTION

If RAMTEST HYPER is executed on a 9 132 pod, then BUS TEST key is pressed, the RAMTEST HYPER parameters appear on the 3-line display after the words TEST BUS.

WORK AROUND

Press the left-arrow key until the cursor is in the left-most field. This will cause the BUS TEST parameters to appear.

9100A-296: Constant Bell At End Of Editor Line

DESCRIPTION

A constant bell can occur when scrolling to the end of a line with the RIGHT SCROLL key pressed down. At the end of the line, the bell continuously sounds, and scrolling up, down,

left or right, does not silence the bell. However, pressing Control-C will silence the bell This is intermittent and difficult to reproduce.

WORK AROUND

Press Control-C.

9100A-298: **Editor-Debugger Wrap-Around** Error

DESCRIPTION

I was using the debugger on a program. I pressed QUIT to exit the debugger. When in the debugger, the top line was a wrap-around with one character extended to the next line. The debugger was exited, but upon an attempt to shift the screen back over, the screen was frozen with the wrap-around still in place. Using the up-arrow and down-arrow keys caused the current line number to update even though the screen was frozen. 1 pressed SCROLL FORWARD and the screen updated. An attempt to duplicate the error was unsuccessful.

WORKAROUND

Press the <Scroll Forward> key.

9100A-323: pred.O UUT compiler crashes on 6.0 response files

DESCRIPTION

At release 6.0 the file format for response files changed to support “don’t care” logic levels for GFI. The response files are not backward compatible. If you save a response file in the new format and try to compile it with a 5.0 or earlier UUT compiler, the system will crash and needs to be rebooted.

WORK AROUND

To determine if an incompatible file format caused the system crash, reboot the system and edit the response file that was being compiled. If the file is incompatible, the editor will display the error message: “Load failed (old format)“.

A 6.0 response file can be converted to a pre-6.0 file format as follows:

1) load 6.0 software on your 9100 and then COPY the response file to a text file.
2) edit the text file and replace any '?' characters that appear in the Async LVL or Clk LVL fields with a space.
3) load 5.0 or earlier software on your 9 100 and then COPY the text file to a response file.

9100A-380 COPY to Port does not copy all of RESPONSE file data

DESCRIPTION

When using COPY in the editor to print RESPONSE files (copy the file to a port) the Priority Pin section of the file is not copied. Doing a COPY to coerce the RESPONSE file to a TEXT file also loses the Priority Pin data. This means that there is no way for a user to get a printout of all of the information in a RESPONSE file.

9100-398: QUIT key not always recognized in OFFSET window

DESCRIPTION

If you enter the response OFFSET window and begin LOOPing, the QUIT key should abort the program execution. However, sometimes you must press the QUIT key several times before it is responded to.

WORKAROUND

Pres the QUIT key several times.

9100A-459: Intermittent Hanging of IEEE 488

DESCRIPTION

IEEE488 communication between a 9400FT and a Fluke 1722A occasionally hangs. It is difficult to reproduce the problem, but it seems to happen more often when the 9100 is tmasferring a file to the 1722A and the operator aborts the transfer by **pressing a** control-C on the 1722A keyboard.

WORK AROUND

If this problem occurs, it usually can be corrected by resetting the ADDRESS, FUNCTION or TLK TERM from the front panel of the 9 100.

9100A-489: Bus Error when copying userdisks onto MDR

DESCRPTION

On some 9100s, you cannot copy userdisks onto the hard disk. If you try to copy /DRl to /HDR (from the front panel), the copy doesn't work. The following error message is displayed, and the 9 100 has to be rebooted:

```
Root: application exited abnormally
Internal Error 102: Bus Error
Press any key to continue
```

If you encounter this bug, please call the Fluke Field Support Group at (206) 3564786.