Faster Erase Times for XC95216 and XC95108 Devices on HP 3070 Series Testers

XAPP113 July 22, 1998 (Version 1.0) Application Note

Summary

This application note describes an enhanced procedure for utilizing the new faster bulk erase capability of the XC95216 and XC95108 devices on the HP 3070 tester.

Xilinx Family

XC9500

Introduction

The bulk erase instruction (FBULK) has been enabled in JTAG ID revision 1 or greater silicon for the XC95216 and XC95108. The revision 1 designation is delineated by a 1 in the 4 msb of the JTAG ID register. FBULK groups together sectors and erases them at the same time. The erase time of XC95108 with 6 sectors is reduced from 20 to 5 seconds, while the XC95216 with 12 sectors is reduced from 40 to 5 seconds.

The revision 1 or greater silicon can be erased with existing vector files, but will erase slower because the FBULK instruction is not utilized. The revision 0 silicon does not have the FBULK instruction enabled. Since both revision 0 and 1 silicon may be encountered in production, two sets of erase vector files must be used: the existing set and a new set that includes the FBULK instruction. This application note describes the method for generating the FBULK vector files, identifying the silicon revision (0 or 1), and applying the correct set of vector files using the HP 3070.

Generating the Bulk Erase SVF file.

Serial Vector Format (SVF) files are used when programming XC9500 devices on automatic test equipment (ATE). The JTAG Programmer allows you to create **.svf** files for use with ATE systems. To create the default (version 0) SVF file, select:

Output → **Create SVF File...**

The **Create a New SVF File** dialog box will appear.

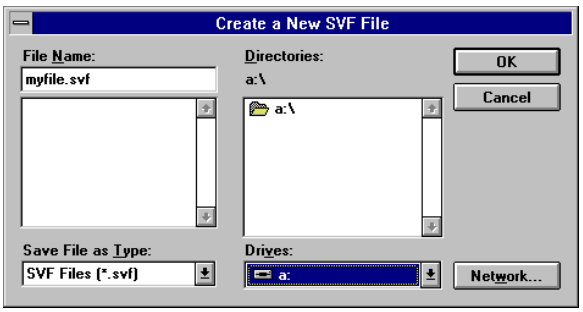

Figure 1: Create an SVF File

Select a name and a directory to create the new file in, then click **OK**.

Then select the device you wish to generate stimulus for and choose the appropriate action from the operations menu.

Note: Program, Verify, Erase, Functional Test, Get Device ID and Get Signature/Usercode are allowed operations in SVF mode.

Substituting with Version 1 Devices

After you generate SVF files for XC95108 or XC95216 Version 0 devices (the default), then, to take advantage of improved ISP capabilities available on Version 1 silicon devices you can generate Version 1 specific SVF files using the following techniques:

Using the Batch Tool (jtagprog)

Invoke the tool to generate SVF files:

```
jtagprog -svf
```

When specifying the part_type in the part command identify Version 1 silicon by appending "**_v1**" to the part name. For example, to specify a chain of Version 1 XC95216s and XC95108s:

```
part xc95216_v1:design216a
xc95108_v1:design108
xc95216_v1:design216b
```

Next, specify operations as usual to generate the required SVF files. Erase design108. This will generate the file "design108.svf." Run this through the **svf2vcl** translator.

Using the JTAG Programmer

In your $XILINX/data directory you will notice BSDL files with the following names:

xc95108.bsd
xc95108_v1.bsd
xc95216.bsd
xc95216_v1.bsd

The BSDL files with the "_v1" in their names describe the Version 1 silicon. To get the software to use Version 1 BSDL files for all devices, you must "trick" the application by renaming files as follows:

1. Rename xc95108.bsd to xc95108_v0.bsd
2. Rename xc95216.bsd to xc95216_v0.bsd
3. Rename xc95108_v1.bsd to xc95108.bsd
4. Rename xc95216_v1.bsd to xc95216.bsd

Invoke the JTAG Programmer and set it to generate SVF files as described earlier in this section. Erase design108. This will generate the file "design108.svf." Run this through the **svf2vcl** translator. When you use the JTAG Programmer, it will default to using the xc95216.bsd and xc95108.bsd files to describe the parts. This will allow access to all Version 1 features.

When you are done programming, remember to change the file names back so that the software will work correctly in non-SVF modes:

1. Rename xc95108.bsd to xc95108_v1.bsd
2. Rename xc95216.bsd to xc95216_v1.bsd
3. Rename xc95108_v0.bsd to xc95108.bsd
4. Rename xc95216_v0.bsd to xc95216.bsd

Separating Erase and Program Files using JTAG Programmer

If you currently implement the erase of the part as a separate file, then skip ahead to the next section. If erase and program are included in a single SVF file, then you will need to generate two new SVF files. One will do the erase for a JTAG revision zero part and the other will erase the program.

Using the batch tool (jtagprog)

1. Invoke the tool.

2. Specify the part command, this time leaving out the "**_v1**." For instance,

   part xc95108:design108 .

3. Erase design108.

4. In another window, rename design108.svf to erase_design108.svf.

5. Then, program the design.

   program -b design108

Another design108.svf will be available with the programming svf statements for design108.

Using the JTAG Programmer

1. Make sure you have renamed the BSDL files so that you are using revision 0 BSDL files.

2. Enter SVF mode as normal.

3. Select design108 and erase.

   Operation → Erase

4. In another window, rename design108.svf to erase_design108.svf.

5. Program the device. Make sure that you deselect the option "erase prior to programming."

   Operation → Program

The program will create another design108.svf with the programming SVF statements for design108.

Running the Appropriate Erase Vectors.

You should now have two separate erase VCL files. One does a faster bulk erase; call it u1_bulk_erase. The other does the old (sector) erase; call it u1_sector_erase.

The general steps for this procedure are as follows:

1. Create a Boundary Scan ID register executable test using the HP Boundary Scan software.

2. Generate the board test as usual.

3. Modify the executable test source code for the Boundary Scan ID test.

4. Modify the BT-Basic testplan to accept and branch on data passed to it from the Boundary Scan ID test.

Step 1. Generate the Boundary Scan ID Test

Run the HP Boundary Scan software as described in the HP3070 Boundary Scan manual. Make sure you specify an appropriate location for the output VCL file (usually in the "digital" directory).

First, you should set the "Default Device" (under "Macros" in the HP Boundary Scan application) to the reference designator for this device. This will create an executable test for you. Note that you must do the "Default Device" prior to running any other macros.

Next, generate an ID register test only. This is listed under the "Macros" menu as "ID Code Test". The software will

create a test which has comments in it for verifying the ID register of the device as it was defined in the BSDL file.

Save the file by doing a **File** → **Save** in the HP Boundary Scan software application.

Step 2. Generate the Board Test

Use Test Consultant to create your board test as usual.

Step 3. Modify Boundary Scan ID Test

Modifying the Boundary Scan ID test is rather simple. You only need to add six lines (more if you add comments) and modify four existing lines (as shown in this example). Refer to the example test for the locations of additions and modifications. Modify the executable test and compile it. See appendix A for an example VCL file. Note the lines in the file following comments of “ADDED THE FOLLOWING LINE.”

Step 4. Modify the Test Plan

Modify the test plan as follows:

- Add another variable to the dimension ("dim") statement for the array that the data passed from the ID test to the testplan. You should have syntax similar to this:

```
      dim ID_Results(10)   ! Dimension a
10 element numeric array.
```

  Note that you should dimension the array for more elements than you are passing back from the ID test. In this example, we are capturing only eight vectors, thus the array size of 10.
- If you plan to test the ID code in the "Digital" subroutine, you MUST add a "global" statement for the data array. It should look like the following:

```
      sub Digital
     global ID_Results(*)     ! Add this
global statement.
```

- The "test" statement for the ID test should look like this:

```
      test "digital/u1_ID_Code";
ID_Results(*)
```

- Now add your conditional branch statements. Note that the first element of the array represents the 28th bit in the ID register (starting from 0) in this example. Also note that to access the four bits representing the Version Code, you need to access elements 0, 2, 4, and 6 of the array. Elements 1, 3, 5, and 7 are indeterminate values.

An example for the testplan branching is shown below:

```
      if ID_Results(0) = 1 then
        test "digital/u1_bulk_erase"   !
Fast erase algorithm device
      else
        test "digital/u1_sector_erase"
! Slow erase algorithm device
      end if
```

At this point, you can test this on your board. Note that it would be relatively easy to pass the entire ID register back to the testplan if you wanted to.

However, a couple of points to note:

- This procedure assumes a single device in a scan chain. If there are multiple devices in the chain, this procedure can be used. The exception to this is when you create an ID test for the device you are interested in. Then you would need the HP Interconnect Boundary Scan software for this step.
- Performance for passing data from the testhead to BT-Basic can be slow. Limit this technique to only those cases where there is no other alternative, and in cases where throughput is a major concern.

Appendix A

```
!!!!    6    0    1  891056649  V909c
! Hewlett-Packard Boundary-Scan Software [960906]
! VCL created from BSDL (Version 0.0) file: /users/tomt/a.bsdl
! Date: Fri Mar 27 20:44:23 1998
! IEEE Std 1149.1-1990

!! Writing code for HP-3070 family.

! Parameters for Entity XC95108:
!   Instruction Length          8
!   Boundary Register Length  324
!   Device Inputs               0
!   Device Outputs              0
!   Device Bidirectionals     108 !! Warning, Disable methods need to be added.

!############################ ADDED THE FOLLOWING LINE ####################
test digital; Result(*)             ! Define parameter to pass to testplan.
```

```
sequential

family TTL !! Warning, Defaulted family

default device "u1"

! The following assignments are derived from Pin-Mapping DIE_BOND.

assign PB00_00  to pins "PAD22"
assign PB00_01  to pins "PAD18"
assign PB00_02  to pins "PAD19"
assign PB00_03  to pins "PAD26"
assign PB00_04  to pins "PAD20"
assign PB00_05  to pins "PAD21"
assign PB00_06  to pins "PAD24"
assign PB00_07  to pins "PAD23"
assign PB00_08  to pins "PAD25"
assign PB00_09  to pins "PAD32"
assign PB00_10  to pins "PAD27"
assign PB00_11  to pins "PAD29"
assign PB00_12  to pins "PAD30"
assign PB00_13  to pins "PAD31"
assign PB00_14  to pins "PAD33"
assign PB00_15  to pins "PAD36"
assign PB00_16  to pins "PAD38"
assign PB00_17  to pins "PAD37"
assign PB01_00  to pins "PAD133"
assign PB01_01  to pins "PAD129"
assign PB01_02  to pins "PAD131"
assign PB01_03  to pins "PAD3"
assign PB01_04  to pins "PAD134"
assign PB01_05  to pins "PAD2"
assign PB01_06  to pins "PAD6"
assign PB01_07  to pins "PAD4"
assign PB01_08  to pins "PAD5"
assign PB01_09  to pins "PAD9"
assign PB01_10  to pins "PAD8"
assign PB01_11  to pins "PAD10"
assign PB01_12  to pins "PAD11"
assign PB01_13  to pins "PAD12"
assign PB01_14  to pins "PAD14"
assign PB01_15  to pins "PAD15"
assign PB01_16  to pins "PAD16"
assign PB01_17  to pins "PAD13"
assign PB02_00  to pins "PAD39"
assign PB02_01  to pins "PAD41"
assign PB02_02  to pins "PAD42"
assign PB02_03  to pins "PAD48"
assign PB02_04  to pins "PAD46"
assign PB02_05  to pins "PAD47"
assign PB02_06  to pins "PAD43"
assign PB02_07  to pins "PAD49"
assign PB02_08  to pins "PAD50"
assign PB02_09  to pins "PAD57"
assign PB02_10  to pins "PAD51"
assign PB02_11  to pins "PAD53"
```

```
assign PB02_12  to pins "PAD45"
assign PB02_13  to pins "PAD54"
assign PB02_14  to pins "PAD55"
assign PB02_15  to pins "PAD56"
assign PB02_16  to pins "PAD65"
assign PB02_17  to pins "PAD62"
assign PB03_00  to pins "PAD105"
assign PB03_01  to pins "PAD112"
assign PB03_02  to pins "PAD113"
assign PB03_03  to pins "PAD111"
assign PB03_04  to pins "PAD116"
assign PB03_05  to pins "PAD117"
assign PB03_06  to pins "PAD109"
assign PB03_07  to pins "PAD118"
assign PB03_08  to pins "PAD120"
assign PB03_09  to pins "PAD125"
assign PB03_10  to pins "PAD121"
assign PB03_11  to pins "PAD122"
assign PB03_12  to pins "PAD128"
assign PB03_13  to pins "PAD124"
assign PB03_14  to pins "PAD126"
assign PB03_15  to pins "PAD123"
assign PB03_16  to pins "PAD127"
assign PB03_17  to pins "PAD130"
assign PB04_00  to pins "PAD64"
assign PB04_01  to pins "PAD67"
assign PB04_02  to pins "PAD70"
assign PB04_03  to pins "PAD60"
assign PB04_04  to pins "PAD72"
assign PB04_05  to pins "PAD74"
assign PB04_06  to pins "PAD66"
assign PB04_07  to pins "PAD76"
assign PB04_08  to pins "PAD78"
assign PB04_09  to pins "PAD71"
assign PB04_10  to pins "PAD80"
assign PB04_11  to pins "PAD82"
assign PB04_12  to pins "PAD73"
assign PB04_13  to pins "PAD83"
assign PB04_14  to pins "PAD86"
assign PB04_15  to pins "PAD81"
assign PB04_16  to pins "PAD87"
assign PB04_17  to pins "PAD75"
assign PB05_00  to pins "PAD77"
assign PB05_01  to pins "PAD88"
assign PB05_02  to pins "PAD89"
assign PB05_03  to pins "PAD100"
assign PB05_04  to pins "PAD91"
assign PB05_05  to pins "PAD93"
assign PB05_06  to pins "PAD90"
assign PB05_07  to pins "PAD95"
assign PB05_08  to pins "PAD97"
assign PB05_09  to pins "PAD92"
assign PB05_10  to pins "PAD99"
assign PB05_11  to pins "PAD101"
assign PB05_12  to pins "PAD96"
assign PB05_13  to pins "PAD104"
```

```
assign PB05_14  to pins "PAD106"
assign PB05_15  to pins "PAD110"
assign PB05_16  to pins "PAD107"
assign PB05_17  to pins "PAD98"
assign TCK      to pins "PAD63"
assign TDI      to pins "PAD59"
assign TDO      to pins "PAD114"
assign TMS      to pins "PAD61"
assign VCCINT_1 to pins *
assign VCCINT_2 to pins *
assign VCCINT_3 to pins *
assign VCCINT_VPP to pins *
assign VCCIO_1  to pins *
assign VCCIO_2  to pins *
assign VCCIO_3  to pins *
assign VCCIO_4  to pins *
assign VCCIO_5  to pins *
assign VCCIO_6  to pins *
assign VSSINT_1 to pins *
assign VSSINT_2 to pins *
assign VSSINT_3 to pins *
assign VSSINT_4 to pins *
assign VSSIO_1  to pins *
assign VSSIO_2  to pins *
assign VSSIO_3  to pins *
assign VSSIO_4  to pins *
assign VSSIO_5  to pins *
assign VSSIO_6  to pins *
assign VSSIO_7  to pins *
assign VSSIO_8  to pins *
assign VSSIO_9  to pins *

power         VCCINT_1, VCCINT_2, VCCINT_3, VCCINT_VPP,  VCCIO_1
power          VCCIO_2,  VCCIO_3,  VCCIO_4,  VCCIO_5,  VCCIO_6
power         VSSINT_1, VSSINT_2, VSSINT_3, VSSINT_4,  VSSIO_1
power          VSSIO_2,  VSSIO_3,  VSSIO_4,  VSSIO_5,  VSSIO_6
power          VSSIO_7,  VSSIO_8,  VSSIO_9
inputs             TCK,      TDI,      TMS
outputs            TDO
bidirectional  PB00_00,  PB00_01,  PB00_02,  PB00_03,  PB00_04
bidirectional  PB00_05,  PB00_06,  PB00_07,  PB00_08,  PB00_09
bidirectional  PB00_10,  PB00_11,  PB00_12,  PB00_13,  PB00_14
bidirectional  PB00_15,  PB00_16,  PB00_17,  PB01_00,  PB01_01
bidirectional  PB01_02,  PB01_03,  PB01_04,  PB01_05,  PB01_06
bidirectional  PB01_07,  PB01_08,  PB01_09,  PB01_10,  PB01_11
bidirectional  PB01_12,  PB01_13,  PB01_14,  PB01_15,  PB01_16
bidirectional  PB01_17,  PB02_00,  PB02_01,  PB02_02,  PB02_03
bidirectional  PB02_04,  PB02_05,  PB02_06,  PB02_07,  PB02_08
bidirectional  PB02_09,  PB02_10,  PB02_11,  PB02_12,  PB02_13
bidirectional  PB02_14,  PB02_15,  PB02_16,  PB02_17,  PB03_00
bidirectional  PB03_01,  PB03_02,  PB03_03,  PB03_04,  PB03_05
bidirectional  PB03_06,  PB03_07,  PB03_08,  PB03_09,  PB03_10
bidirectional  PB03_11,  PB03_12,  PB03_13,  PB03_14,  PB03_15
bidirectional  PB03_16,  PB03_17,  PB04_00,  PB04_01,  PB04_02
bidirectional  PB04_03,  PB04_04,  PB04_05,  PB04_06,  PB04_07
bidirectional  PB04_08,  PB04_09,  PB04_10,  PB04_11,  PB04_12
```

```
bidirectional  PB04_13,  PB04_14,  PB04_15,  PB04_16,  PB04_17
bidirectional  PB05_00,  PB05_01,  PB05_02,  PB05_03,  PB05_04
bidirectional  PB05_05,  PB05_06,  PB05_07,  PB05_08,  PB05_09
bidirectional  PB05_10,  PB05_11,  PB05_12,  PB05_13,  PB05_14
bidirectional  PB05_15,  PB05_16,  PB05_17
!
vector cycle  500n
receive delay 400n

!############################## ADDED THE FOLLOWING LINE ####################
capture TDO                          ! Define pin that we are capturing.

pcf order default Parallel is      TCK,      TMS,      TDI,      TDO

pcf order default Parallel is  PB00_00,  PB00_01,  PB00_02,  PB00_03,  PB00_04
pcf order default Parallel is  PB00_05,  PB00_06,  PB00_07,  PB00_08,  PB00_09
pcf order default Parallel is  PB00_10,  PB00_11,  PB00_12,  PB00_13,  PB00_14
pcf order default Parallel is  PB00_15,  PB00_16,  PB00_17,  PB01_00,  PB01_01
pcf order default Parallel is  PB01_02,  PB01_03,  PB01_04,  PB01_05,  PB01_06
pcf order default Parallel is  PB01_07,  PB01_08,  PB01_09,  PB01_10,  PB01_11
pcf order default Parallel is  PB01_12,  PB01_13,  PB01_14,  PB01_15,  PB01_16
pcf order default Parallel is  PB01_17,  PB02_00,  PB02_01,  PB02_02,  PB02_03
pcf order default Parallel is  PB02_04,  PB02_05,  PB02_06,  PB02_07,  PB02_08
pcf order default Parallel is  PB02_09,  PB02_10,  PB02_11,  PB02_12,  PB02_13
pcf order default Parallel is  PB02_14,  PB02_15,  PB02_16,  PB02_17,  PB03_00
pcf order default Parallel is  PB03_01,  PB03_02,  PB03_03,  PB03_04,  PB03_05
pcf order default Parallel is  PB03_06,  PB03_07,  PB03_08,  PB03_09,  PB03_10
pcf order default Parallel is  PB03_11,  PB03_12,  PB03_13,  PB03_14,  PB03_15
pcf order default Parallel is  PB03_16,  PB03_17,  PB04_00,  PB04_01,  PB04_02
pcf order default Parallel is  PB04_03,  PB04_04,  PB04_05,  PB04_06,  PB04_07
pcf order default Parallel is  PB04_08,  PB04_09,  PB04_10,  PB04_11,  PB04_12
pcf order default Parallel is  PB04_13,  PB04_14,  PB04_15,  PB04_16,  PB04_17
pcf order default Parallel is  PB05_00,  PB05_01,  PB05_02,  PB05_03,  PB05_04
pcf order default Parallel is  PB05_05,  PB05_06,  PB05_07,  PB05_08,  PB05_09
pcf order default Parallel is  PB05_10,  PB05_11,  PB05_12,  PB05_13,  PB05_14
pcf order default Parallel is  PB05_15,  PB05_16,  PB05_17
!
pcf order Scan is      TCK,      TMS,      TDI,      TDO

!Column-to-signal Map, signals 1 to 75
!TTTTPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP!
!CMDDBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB!
!KSIO00000000000000000000000000000000000000000000000000000000000000000000!
!    00000000000000000011111111111111111122222222222222222333333333333333333!
!    _____________________________________________________________________________!
!    00000000001111111100000000001111111100000000001111111100000000001111111!
!    01234567890123456701234567890123456701234567890123456701234567890123456!
!                                                                             !
!
!Column-to-signal Map, signals 76 to 112
!PPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPPP!
!BBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBBB!
!00000000000000000000000000000000000!
!34444444444444444455555555555555555!
!_____________________________________!
!10000000000111111110000000011111111!
```

```
!7012345678901234567012345678901234567!
!                                     !
!!
unit "IDCode_Test" ! Vector 1
pcf
use pcf order Parallel
"01ZXZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ"
"ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ"
use pcf order Scan
"11ZX"
"01ZX"
"11ZX"
"01ZX"
"11ZX"
"01ZX"
"11ZX"
"01ZX"
"11ZX"
"01ZX"
"11ZX"! Test-Logic-Reset
! Current instruction IDCODE (11111110), target register IDCODE[32].
"00ZX"
"10ZX"! Run-Test/Idle
use pcf order Parallel
"01ZXZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ"
"ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ"
use pcf order Scan
"11ZX"! Select-DR-Scan
use pcf order Parallel
"00ZXZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ"
"ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ"
use pcf order Scan
"10ZX"! Capture-DR
use pcf order Parallel
"00ZXZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ"
"ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ"
"10ZXZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ"
"ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ"
! Shift-DR
! Target IDCODE[32] = 00011001010100000110000010010011
! ID Code is 00011001010100000110000010010011
end pcf
message "1149.1 Device ID failed bit 0."
pcf
use pcf order Parallel
"001HZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ"
"ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ"
! 0
use pcf order Scan
"101X"
end pcf
message "1149.1 Device ID failed Manufacturer"
message "Code."
pcf
use pcf order Scan
"000H"! 1
```

```
"100X"
! Vector 25
"000L"! 2
"100X"
"000L"! 3
"100X"
"000H"! 4
"100X"
"000L"! 5
"100X"
"000L"! 6
"100X"
"000H"! 7
"100X"
"000L"! 8
"100X"
"000L"! 9
"100X"
"000L"! 10
"100X"
"000L"! 11
"100X"
end pcf
message "1149.1 Device ID failed Part Number."
pcf
use pcf order Scan
"000L"! 12
"100X"
"000H"! 13
"100X"
"000H"! 14
! Vector 50
"100X"
"000L"! 15
"100X"
"000L"! 16
"100X"
"000L"! 17
"100X"
"000L"! 18
"100X"
"000L"! 19
"100X"
"000H"! 20
"100X"
"000L"! 21
"100X"
"000H"! 22
"100X"
"000L"! 23
"100X"
"000H"! 24
"100X"
"000L"! 25
"100X"
"000L"! 26
```

```
"100X"
! Vector 75
"000H"! 27
"100X"
end pcf
message "1149.1 Device ID failed Version Code."
!############################## ADDED THE FOLLOWING LINE ####################
capture                              ! Define start of data capture block
pcf
use pcf order Scan
"000X"! 28          ###### LINE MODIFIED:  "TDO" from H to X
"100X"
"000X"! 29          ###### LINE MODIFIED:  "TDO" from L to X
"100X"
"000X"! 30          ###### LINE MODIFIED:  "TDO" from L to X
"100X"
"010X"! 31          ###### LINE MODIFIED:  "TDO" from L to X
"110X"! Exit1-DR
! Target IDCODE[32] = 00000000000000000000000000000001
end pcf
!############################## ADDED THE FOLLOWING LINE ####################
end capture                          ! End of data capture block
message ""
pcf
use pcf order Parallel
"01ZXZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ"
"ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ"
use pcf order Scan
"11ZX"! Update-DR
"01ZX"
"11ZX"! Select-DR-Scan
"01ZX"
"11ZX"! Select-IR-Scan
use pcf order Parallel
"01ZXZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ"
"ZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZZ"
use pcf order Scan
"11ZX"! Test-Logic-Reset
! Current instruction IDCODE (11111110), target register IDCODE[32].
"01ZX"
"11ZX"! Test-Logic-Reset

!############################## ADDED THE FOLLOWING LINE ####################
"ZZZX"                               ! Turn-off the drivers.
end pcf
!############################## ADDED THE FOLLOWING LINE ####################
fetch Result(*)                      ! Pass the data back to the testplan
end unit ! IDCode_Test Vector 94

! Vectors with TDI High:       2, (1.0e-06 sec)
! Vectors with TDI Low:       62, (3.1e-05 sec)
! Total time for test:            (4.7e-05 sec)
```