# Boundary Scan User's Guide

# Boundary Scan User's Guide

# Boundary Scan User's Guide

## Introduction

TRACE32 boundary scan function supports:

- Interactive board connection test

- Automated board connection test

- FLASH programming (See **"FLASH Programming via Boundary Scan"** in Onchip/NOR FLASH Programming User's Guide, page 88 (norflash.pdf).

Additionally, all boundary scan instructions and data registers, which are described in the BSDL file can be accessed.

## Intended Audience

PCB developers who want to:

- Check interconnects and signals for initial operation of their board

- Test board mounting for developer and engineering boards

Software developers who want to:

- Measure port states

- Drive signals on printed circuit boards (PCBs)

- Program and read out external flash memories

IC manufacturers who want to:

- Run built-in self tests of their ICs

- Enable special test features, which can be controlled via boundary scan

# How This Manual is Organized

- **"What to know about Boundary Scan":** A short introduction to the boundary scan.

- **"Configuration of the Boundary Scan Chain"**: Describes how to set up a boundary scan chain and do basic checks.

- **"General Operation"**: Describes boundary scan operation and working with the BSDL GUI.

- **"Interactive Board Test"**: Describes how to do interactive board testing.

- **"Automated Board Test"**: Describes how to set up a test for a printed circuit board (PCB) script and run the test.

- **"Special Tests"**: Describes more test and debug features.

# Related Documents

For information about NOR FLASH programming, please refer to:

- **"Onchip/NOR FLASH Programming User's Guide"** (norflash.pdf)

For information about eMMC programming, please refer to:

- **"eMMC FLASH Programming User's Guide"** (emmcflash.pdf)

For information about serial FLASH programming, please refer to:

- **"Serial FLASH Programming User's Guide"** (serialflash.pdf)

For information about the boundary scan commands (BSDL), please refer to:

- **BSDL** in **"General Commands Reference Guide B"** (general_ref_b.pdf)

# List of Abbreviations

| BSDL | Boundary Scan Description Language |
|------|-----------------------------------|
| DR   | Data Register |
| IC   | Integrated Circuit |
| IR   | Instruction Register |
| PCB  | Printed Circuit Board |
| TAP  | Test Access Port |

| TCK | Test Clock Input |
|---|---|
| TDI | Test Data Input |
| TDO | Test Data Output |
| TMS | Test Mode Select |
| TRST_N | Test Reset Input |

# What to know about Boundary Scan

Boundary scan is a method for testing interconnects on PCBs and internal IC sub-blocks. It is defined in the IEEE 1149.1 standard.

For boundary scan tests, additional logic is added to the device. Boundary scan cells are placed between the core logic and the ports.
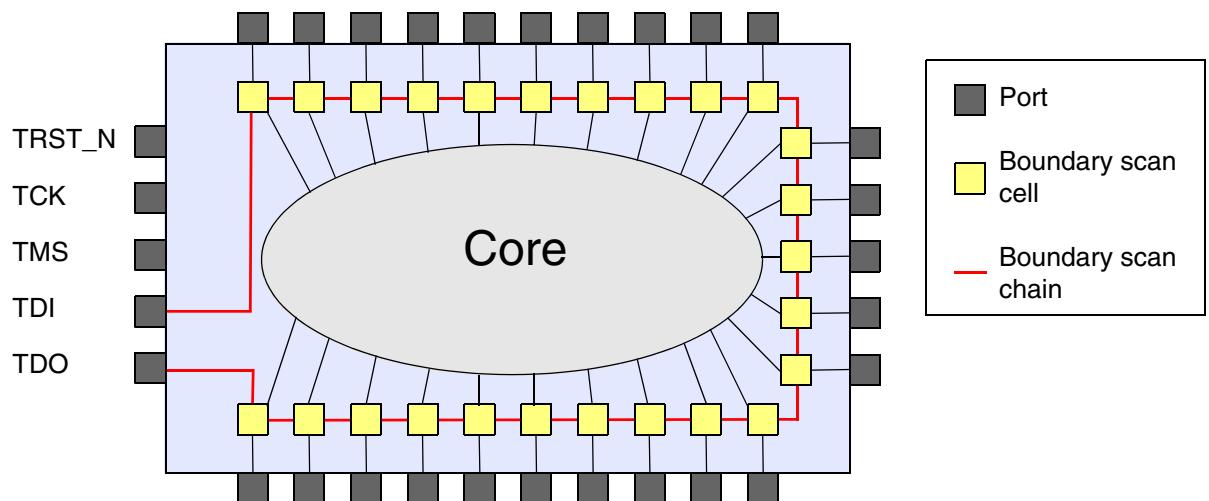


In normal mode, these cells are transparent and the core is connected to that ports. In boundary scan mode, the core is isolated from the ports and the port signals are controlled by the JTAG interface.

The boundary scan cells are connected to a serial shift register, which is referred to as the boundary scan register. This register can be used to read and write port states.

Following boundary scan instructions are defined in the IEEE standard:

- BYPASS (mandatory): TDI is connected to TDO via a single shift register.

- SAMPLE (mandatory): Takes a snapshot of the normal operation of the IC.

- PRELOAD (mandatory): Loads data to the boundary scan register.

- EXTEST (mandatory): Apply preloaded data of the boundary scan register to the ports.

- INTEST (optional): Apply preloaded data of the boundary scan register to the core logic.

- RUNBIST (optional): Executes a self-contained self-test of the IC.

- CLAMP (optional): Apply preloaded data of the boundary scan register to the ports and selects the bypass register as the serial path between TDI and TDO.

- IDCODE (optional): Reads the device identification register.

- USERCODE (optional): Reads and writes a user programmable identification register.

- HIGHZ (optional): Places the IC in an inactive drive state (e.g. all ports are set to high impedance state).

In addition, IC specific instructions may be defined.

The structure of the boundary scan chain and the instruction set are described with the Boundary Scan Description Language (BSDL). BSDL is a subset of the Very High-level Design Language (VHDL). The BSDL files are provided by the IC manufacturer.

# Configuration of the Boundary Scan Chain

The steps below provide an overview of configuration process and are described in detail in the following sections.

1. Configure and initialize Boundary scan engine. (See **Configure Boundary Scan Engine**)

2. Load the BSDL files in the correct order. (See "**Loading the BSDL Files**".)

3. Initialize the boundary scan chain. (See "**Initialization of the Boundary Scan Chain**".)

4. Check the configuration of the boundary scan chain. (See "**Check the Configuration**".)

# Configure Boundary Scan Engine

To start from a defined state use command **BSDL.RESet** first. It will initialize the boundary scan engine to:

- Empty boundary scan chain (i.e. all loaded BSDL files will be removed)

- Removes any previously defined FLASH configuration

- Sets parking state to Run-Test/Idle

- Disables set-and-run mode

- Disables two-step-DR mode

- Disables result checking

- Sets the current state of the boundary scan chain to unknown (i.e. a **BSDL.SOFTRESET** or **BSDL.HARDRESET** command must executed first to start boundary scan tests)

### Interaction with Debug Function

If only boundary scan functions are used, the debug functions should be turned off with:

```
; disable the debugger
SYStem.Down
```

For FLASH programming with boundary scan the system must be in down mode (**SYStem.Mode Down**).
It is also recommended to lock the JTAG interface during FLASH programming with boundary scan:
```
JTAG.LOCK
... ;FLASH programming via boundary scan
JTAG.UNLOCK
```

If boundary scan and debugging should be used at the same time, the boundary scan configuration and initialization should be done prior debug configuration:

```
; disable the debugger
SYStem.Down
;Initialize the boundary scan engine
BSDL.RESet
;Load the BSDL file(s)
BSDL.FILE file1.bsdl
;Reset the boundary scan chain
BSDL.SOFTRESET

; now configure the debugger
...
```

It is also required to set the boundary scan instruction register each time when a boundary scan data register is accessed (i.e. use always **BSDL.RUN** and not **BSDL.RUN DR**).

## The Parking State

Each boundary scan operation starts and stops from the parking state. The default parking state is the Run-Test/Idle state. It can be modified with the **BSDL.ParkState** command.

If the boundary scan park state is different from the debug park state, unintended side effects may occur. To avoid this, it is recommended to set the parking state of the boundary scan engine to the parking state of the debug engine:

```
; initialize the boundary scan engine
BSDL.RESet
; Set the parking state to Select-DR-Scan (e.g. for Arm or MIPS)
BSDL.ParkState Select-DR-Scan
```

If it is not possible to use the same parking state and only boundary scan functions are used, the JTAG interface should be locked:

```
; disable the debugger and lock the JTAG interface
SYStem.Down
JTAG.LOCK
;Initialize the boundary scan engine
BSDL.RESet
;Load the BSDL file(s)
BSDL.FILE file1.bsdl
;Reset the boundary scan chain
BSDL.SOFTRESET
; do boundary scan functions
...
;unlock the JTAG interface
JTAG.UNLOCK
```

# Loading the BSDL Files

The boundary scan chain of a PCB is configured by loading the required BSDL files in the correct order. The BSDL file for the IC connected to the TDO port of the PCB must be loaded first (IC1). The IC connected to the TDI port must be loaded as the last file.
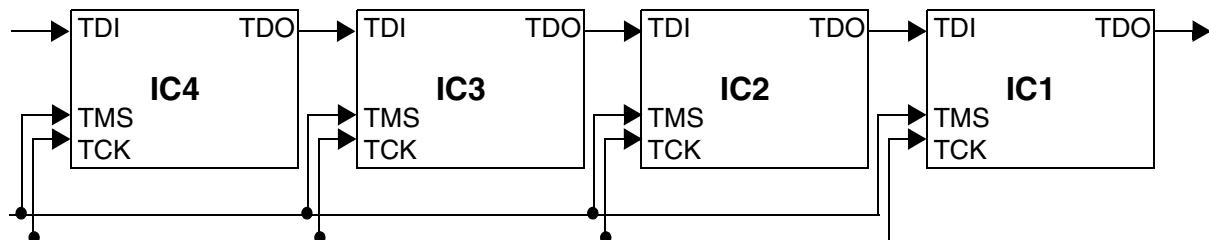


Use the following command to open the configuration window for boundary scan tests:

| | |
|---|---|
| **BSDL.state** | Opens the **BSDL.state** window. |
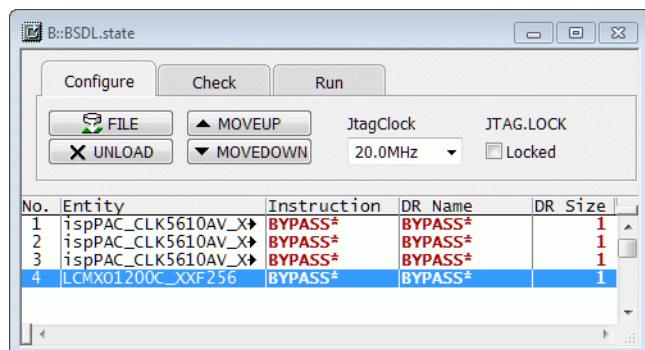


The **FILE** button adds a boundary scan file to the current position, the **UNLOAD** button removes the currently selected file.

Within the **BSDL.state** window, the currently selected file can be positioned with the **MOVEUP** and **MOVEDOWN** buttons. The list shows the current configuration of the boundary scan chain, i.e. the list entry no.1 is IC1 (connected to TDO of the PCB), the last list entry is connected to the TDI of the PCB.

Alternatively, the BSDL files can be loaded with the command **BSDL.FILE**.

```
BSDL.FILE ispCLOCK5610Av_isc.bsm        ; Load BSDL file for IC1

BSDL.FILE ispCLOCK5610Av_isc.bsm        ; Load BSDL file for IC2

BSDL.FILE ispCLOCK5610Av_isc.bsm        ; Load BSDL file for IC3

BSDL.FILE LCMXCO1200C_ftBGA256.bsdl     ; Load BSDL file for IC4
```

# Initialization of the Boundary Scan Chain

After loading the BSDL files, the boundary scan chain should be initialized.

```
BSDL.SOFTRESET                    ; do a sequential TAP reset (through TAP
                                  ; state Test-Logic-Reset)
```

Some ICs may require a reset via TRST_N (e.g. to latch the device into boundary scan mode).

```
BSDL.HARDRESET                    ; do an asynchronous TAP reset via TRST_N
```

Both commands will reset the TAP controllers for all ICs in the boundary scan chain and halt in *Select-DR-SCAN* state. Usually only one of these commands is required, only if an IC in the boundary scan chain requires a TRST_N reset and at least one IC in the boundary scan chain has no TRST_N port, then apply both commands in the following order:

```
BSDL.HARDRESET

BSDL.SOFTRESET
```

# Check the Configuration

After the configuration of the boundary scan chain, the BYPASS mode and the IDCODE of all ICs in the chain should be checked. This means that the debugger verifies if JTAG works correctly and if the BSDL files match the selected ICs.

On tab **Check** of the **BSDL.state** window these checks can be done with the buttons **BYPASSall** and **IDCODEall**. The check results are shown next to the buttons.
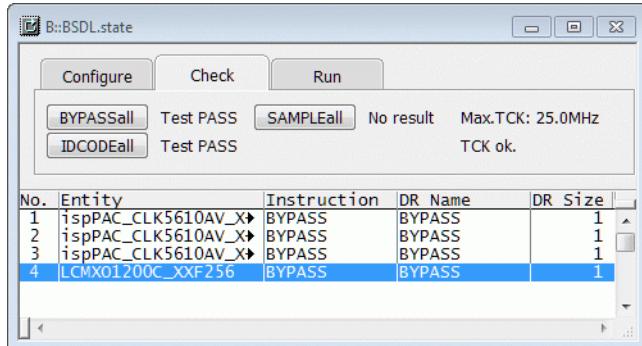


Alternatively, the checks can be performed with the following commands:

```
BSDL.BYPASSall                    ; Check BYPASS mode for all ICs

BSDL.IDCODEall                    ; Check the IDCODE of all ICs
```

If a check fails, an error will be reported.

For PRACTICE scripts, the functions **BSDL.CHECK.BYPASS()** and **BSDL.CHECK.IDCODE()** can be used for checking *BYPASS* and *IDCODE* of the boundary scan chain.

```
...
if !bsdl.check.bypass()
     PRINT %ERROR "Bypass test failed"
else
(
    if !bsdl.check.idcode()
         PRINT %ERROR "ID code test failed"
    else
    (
        ...                                    ; further commands
```

# General Operation

## Basic Mode of Operation

A boundary scan step consists of two phases:

1.    Preparation of the boundary scan chain with the **BSDL.SET** command

2.    Execution of the boundary scan instruction and data settings with the **BSDL.RUN** command

During the preparation phase, instruction and data register settings are modified, but not shifted to the boundary scan chain. All modifications are stored in shadow registers on the host side. For each data register in the BSDL file there is a set of shadow registers, which will store the status of the boundary scan chain until the end of the boundary scan test.

With the **BSDL.RUN** command, the current status of the boundary scan chain is shifted to the hardware registers of the ICs on the PCB.

## Preparation of the Boundary Scan Chain

The general command for the preparation of the boundary scan chain is:

| | |
|---|---|
| **BSDL.SET** *<chip_number> <set_selection>* | Modify settings *<set_selection>* for IC *<chip_number>* |

An arbitrary number of **BSDL.SET** commands can be applied before execution. All modifications are incremental, i.e. a previously modified register bit is only overwritten, if a **BSDL.SET** command addresses the same bit.

The boundary scan chain is prepared by modifying the instruction registers, the data register and the boundary scan register, as described in the following sections.

### Modification of Instruction Registers

An instruction register is modified with:

| | |
|---|---|
| **BSDL.SET** *<chip_number>* **IR** *<instrname>* | Set instruction *<instrname>* for IC *<chip_number>* |

**Example**:

```
BSDL.SET 3. IR BYPASS                          ; Set instruction BYPASS for IC3
```

Instructions can be only set, if they are defined in BSDL file of the IC *<chip_number>*. If an instruction register is modified, the appropriate fields in the **BSDL.state** window will change their color to red.
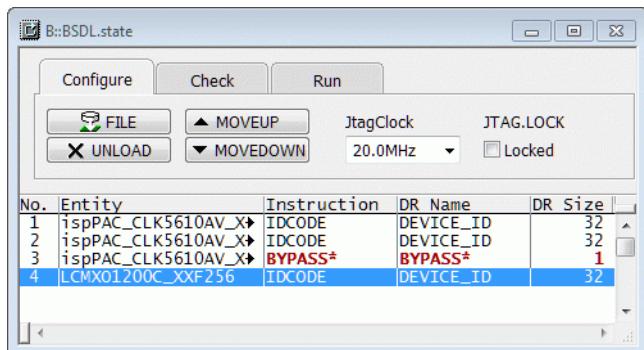


## Modification of a Data Register

The general command for modifying a data register is:

**BSDL.SET** *<chip_number>* **DR** *<bitslice>* *<setvalue>*          Set *<bitslice>* of current data register to value *<setvalue>* for IC *<chip_number>*

The *<bitslice>* can be a single bit, a range (i.--k.) or the whole data register (*). *<setvalue>* can be **ZERO**, **ONE** or a hexadecimal number for write data register. With **ExpectH**, **ExpectL** and **ExpectX** the expected value for the read data register can be set.

To check the values of the read data register, set the result check to **ON**:

```
BSDL.CHECK ON                      ; Enables result checking
```

**Examples**:

```
BSDL.SET 4. DR * ZERO             ; Set all bits of the current data
                                  ; register of IC4 to zero

BSDL.SET 4. DR 4.--10. ONE        ; Set  bits 4-10 of the current data
                                  ; register of IC4 to one

BSDL.SET 4. DR 16.--32. 0xa5a5    ; Set bits 16-32 of the current data
                                  ; register of IC4 to 0xa5a5
```

# Modification of the Boundary Scan Register

The boundary scan register can only be modified if one of the following instructions is set: *SAMPLE*, *PRELOAD*, *EXTEST*, *INTEST*).

There are two ways to modify the boundary scan register:

1.   The **BSDL.SET BSR** command.

2.   The **BSDL.SET PORT** command.

### The BSDL.SET BSR Command

With the **BSDL.SET BSR** command bit slices of the boundary scan register can be modified:

| | |
|---|---|
| **BSDL.SET** *<chip_number>* **BSR** *<bitslice> <setvalue>* | Set *<bitslice>* of the boundary scan register to value *<setvalue>* for IC *<chip_number>* |

It has the following *<setvalues>*:

•   ZERO, ONE: sets *<bitslice>* to zero or one

•   SAFE: sets *<bitslice>* to *SAFE* state acc. BSDL file

•   SAMPLE: sets *<bitslice>* to previously sampled data

•   DISable, ENable: disables/enables output drivers for all ports in *<bitslice>*

•   Drive0, Drive1: drives 0/1 for all ports in *<bitslice>*

•   ExpectX, ExpectH, ExpectL: sets expected read data of *<bitslice>* to ignore (X)/high (H)/low (L)

•   Hexadecimal number: sets *<bitslice>* to this value

Besides these additional *<setvalues>* all other *<setvalues>* of the **BSDL.SET DR** command can be used.

```
  BSDL.SET 4. BSR * ZERO            ; Set all bits of the boundary scan
                                    ; register of IC4 to zero

  BSDL.SET 4. BSR 4.--10. SAFE      ; Set bits 4-10 of the boundary scan
                                    ; register of IC4 to SAFE state

  BSDL.SET 4. BSR 16.--32. SAMPLE   ; Set bits 16-32 of the boundary scan
                                    ; register of IC4 to previously
                                    ; sampled data
```

Use the **BSDL.SET PORT** command for the direct modification of ports:

| | |
|---|---|
| **BSDL.SET** *<chip_number>* **PORT** *<port_name>* *<port_value>* | Set *port <port_name>* to value *<port_value>* for IC *<chip_number>* |

The *<port_name>* must be a valid port name from the BSDL file of the IC *<chip_number>*.

```
BSDL.SET 4. PORT PL7A 1          ; Set port PL7A of IC4 to drive 1

BSDL.SET 3. PORT PS1 H           ; Set port PS1 of IC3 to read high level
```

# Execution of the Boundary Scan Commands

After completing the preparations of the boundary scan chain, you can apply the modified settings to the hardware registers of the boundary scan chain.

```
BSDL.RUN                          ; Execute an IR-Scan and then a
                                  ; DR-SCAN operation
```

To apply only the instruction register settings to the boundary scan chain, use this command:

```
BSDL.RUN IR                       ; Execute an IR-SCAN operation
```

To apply only the data register settings to the boundary scan chain, use this command:

```
BSDL.RUN DR                       ; Execute a DR-SCAN operation
```

If the result check is set to **ON**, the bits of the read data registers of all ICs in the boundary scan chain will be compared to the expected values. If the result mismatches, an error will be reported at the message line. For more details about the error, open the **AREA.view** window

# Working with the GUI

## The BSDL.state Window

The **BSDL.state** window consists of two parts: the dialog area with 3 tabs **Configure**, **Check**, **Run** and the list with the loaded boundary scan structures.

### Configure Tab

Use the configure tab to configure the boundary scan:

•        Load BSDL files

•        Remove boundary scan structures from the boundary scan chain

•        Arrange boundary scan structures

•        Set JTAG clock (TCK frequency)

•        Lock JTAG interface
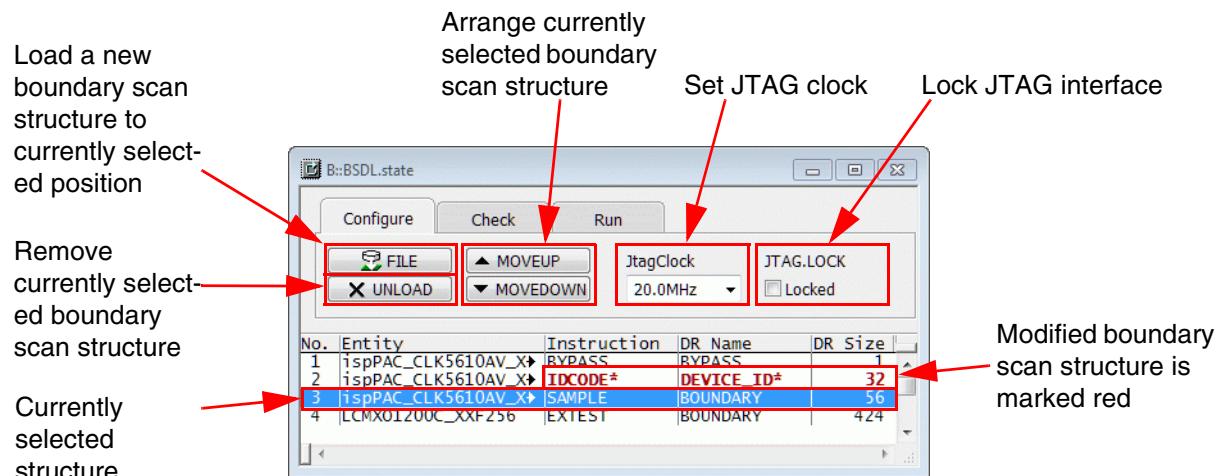


In the list area the loaded boundary scan structures for the ICs in the boundary scan chain are shown. The columns give information about:

•        **No.**: Position of the IC in the boundary scan chain (TDI -> ICn -> ... -> IC1 -> TDO)

•        **Entity**: Name of the IC entity (from the BSDL file)

•        **Instruction**: Current instruction of the IC

•        **DR Name**: Name of the current data register of the IC (depending on the current instruction)

•        **DR Size**: Size of the current data register of the IC (depending on the current instruction)

On the **Check** tab, you can start the checks of the boundary scan chain by clicking the buttons **BYPASSall**, **IDCODEall**, and **SAMPLEall**.

The check results are displayed next to the buttons as shown in the screenshot below.

BYPASSall
button and
check result

Max. TCK for the
boundary scan
chain (from the
BSDL files) and
compare result
with current TCK
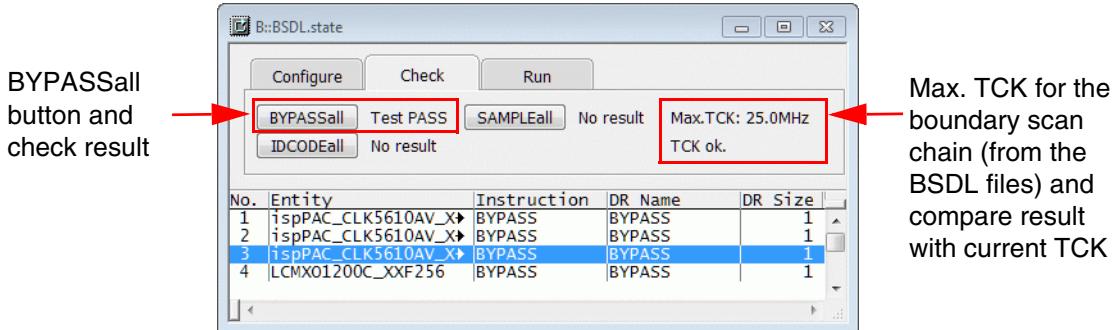
The **BYPASSall** check performs a *BYPASS* test of the boundary scan chain.

The **IDCODEall** check performs an *IDCODE* test for all ICs in the boundary scan chain.

The **SAMPLEall** check takes a *SAMPLE* from all ICs in the boundary scan chain.

The check results can be:

- **No result**: test was not executed (**BYPASSall**, **IDCODEall**, **SAMPLEall**)

- **Test PASS**: test was executed and passed (**BYPASSall**, **IDCODEall**)

- **Test FAIL**: test was executed and failed (**BYPASSall**, **IDCODEall**)

- **Test done**: test was executed (**SAMPLEall**)

**Run Tab**

On the **Run** tab, you can execute the boundary scan settings by clicking the buttons **RUN IR**, **RUN DR**, and **RUN**. You can release **SOFTRESET** and **HARDRESET** for the boundary scan chain and set execution options:

- **CHECK**: enable/disable read result checking

- **SetAndRun**: enable/disable immediate takeover of data register modifications

- **TwoStepDR**: enable/disable two *DR-SCAN* cycles for **RUN/RUN DR** command

In addition, you can set instructions by right-clicking the desired row and selecting an entry from the context menu (see screenshot below).
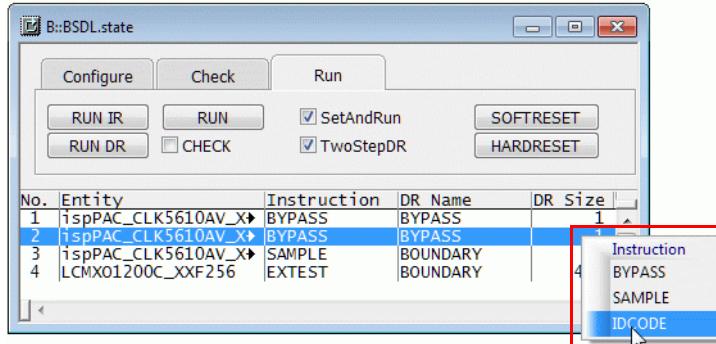
Context menu for quick modification of the ICs instruction Only *BYPASS*, *SAMPLE* and *IDCODE* possible, other settings must be done in the **BSDL.SET** window

With the context menu, you can set the 3 "read-only" instructions (*BYPASS, SAMPLE, IDCODE*) for the selected list entry.

# The BSDL.SET Window

The **BSDL.SET** window for an IC is either opened with a double click on the IC's entry in the **BSDL.state** window or with the command:

```
BSDL.SET 3.                          ; Opens the configuration and result
                                     ; window for IC 3
```

In the **BSDL.SET** window all instructions defined in the BSDL file can be set, options can be selected, data register bits can be set and the data register results can be viewed.

The content of the data area depends on the selected instruction, the *Filter data* settings and the *DR mode*.

The data area provides the following views:

1.     Sample view

2.     Set Write view

3.     Set Read view

4.     File Info
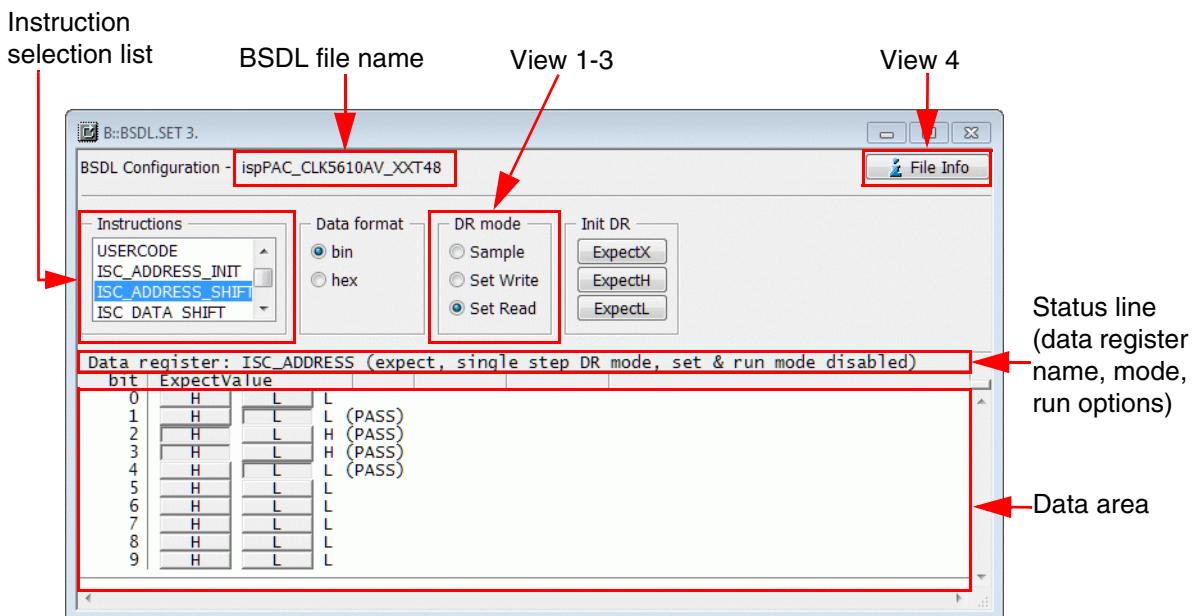
## Sample View

In the **Sample** view the results of the last data register read operation is shown. With **Spotlight** enabled, register bit changes are highlighted.



## Set Write View

In the **Set Write** view, the current data write register can be initialized and single data register bits can be set.

If the *Set and Run* mode is disabled (**BSDL.SetAndRun OFF**), the data write register is only prepared. To apply these settings to the boundary scan chain a **BSDL.RUN DR** command is required.

With *Set and Run* mode enabled, each modification of the data write register will be immediately transferred to the boundary scan chain.

Data register initialization

Modified data register bits are marked

Register buttons: toggles register bit value (0/1)

Enable buttons: enables/disables port output driver (if defined in BSDL file for this port)

Safe buttons: enables/disables register bit *SAFE* state (if defined in BSDL file for this bit)

The *Init DR* / *Init BSR* buttons ignores the filter settings, i.e. all data register bits are set to the specified value.

**Set Read View**

In the **Set Read** view, the expected values for the data read register can be verified and modified.



In the *Last Result* column, the sampled results of the last read operation and the check results can be viewed.

## File info

The toggle button **File Info** / **Test Results** switches between the data register view and the file information view. When file information view is selected, the read data from the BSDL file of the IC can be viewed.



Toggle button

# Interactive Board Test

For interactive tests of the PCB's interconnects, the following steps should be executed:

1.  System setup

2.  Configure boundary scan chain

3.  Check boundary scan chain

4.  Configure run mode

5.  Execute level and connection tests

It is recommended to bring the system in down state. This avoids interferences from the debugger to the boundary scan tests. The following example will do the first 4 steps:

```
;System setup
SYStem.Down
SYStem.JtagClock 20MHz
BSDL.RESet

;Configure boundary scan chain
BSDL.FILE ispCLOCK5610Av_isc.bsm
BSDL.FILE ispCLOCK5610Av_isc.bsm
BSDL.FILE ispCLOCK5610Av_isc.bsm
BSDL.FILE LCMXCO1200C_ftBGA256.bsdl

;Check boundary scan chain
BSDL.HARDRESET
BSDL.SOFTRESET

IF !BSDL.CHECK.BYPASS()
(
    BSDL.BYPASSall
    PRINT %ERROR "Bypass test failed"
    ENDDO
)
IF !BSDL.CHECK.IDCODE()
(
    BSDL.IDCODEall
    PRINT %ERROR "ID code test failed"
    ENDDO
)

;Configure run mode
BSDL.SetAndRun ON
BSDL.TwoStepDR ON

;Execute level and connection tests....
```

# Configure Run Mode

For interactive tests, the two commands **BSDL.SetAndRun** and **BSDL.TwoStepDR** are useful. When enabled:

- A **BSDL.RUN DR** command will be automatically executed when a data register bit is modified (**BSDL.SetAndRun ON**)

- Each **BSDL.RUN DR** command will be execute two *DR-SCAN* operations (**BSDL.TwoStepDR ON**)

## Set and Run Mode

With the *Set and Run* mode enabled (**BSDL.SetAndRun ON**), any change of a data register bit will be immediately executed, no matter if it was modified on the command line or interactively in a **BSDL.SET** window.

As a result, each modification of a data register bit or data register bit slice will immediately do the following:

1. Execute an *IR-SCAN* operation if any instruction register in the boundary scan chain was modified.

2. Execute a *DR-SCAN* operation.

## Two Step DR Mode

A single *DR-SCAN* operation will first capture the data read register and then apply the new data write register:



Capture (i.e. sample) port state from previous cycle **n-1**

Shift in current port state **n** and shift out previous port state **n-1**

Update (i.e. apply) current port state **n**
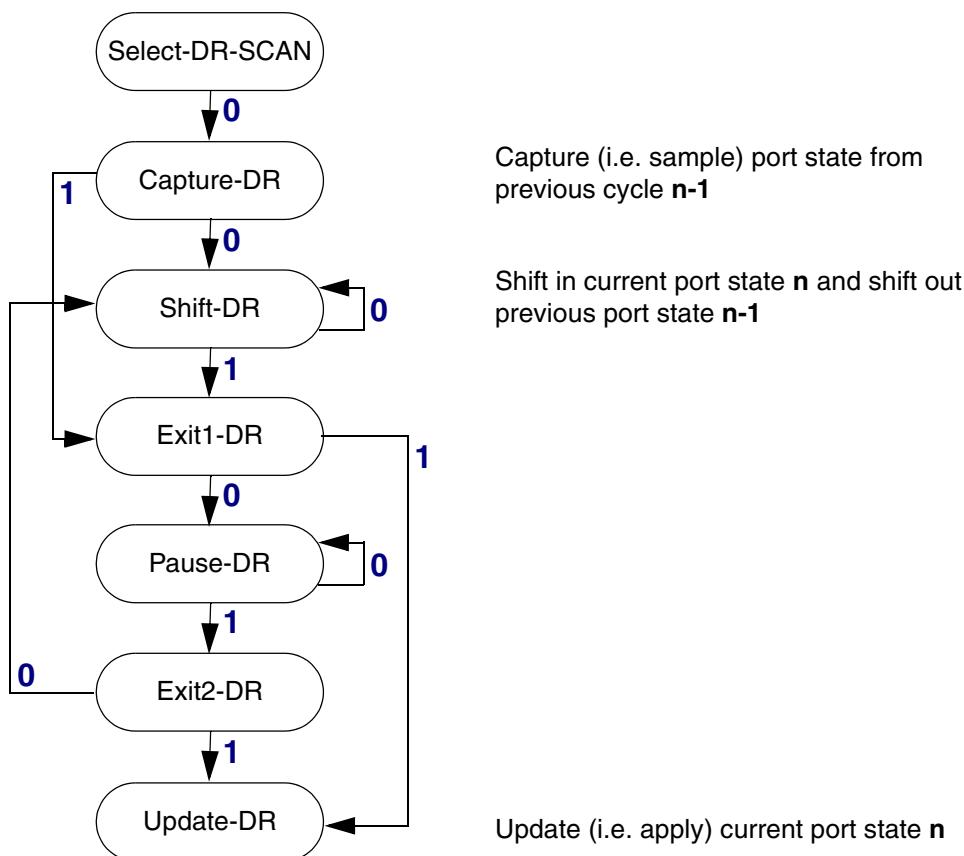
To see the effect of the new write data on the sampled port state a second *DR-SCAN* operation is required. With the "*Two Step DR* mode" enabled (**BSDL.TwoStepDR ON**), this second *DR-SCAN* operation is executed automatically.

| BSDL.TwoStepDR OFF | BSDL.TwoStepDR ON |
|---|---|
| Capture n-1 | Capture n-1 |
| Shift in n<br>Shift out n-1 | Shift in n<br>Shift out n-1 (will be discarded) |
| Update to n | Update to n |
| | Capture n |
| | Shift in n<br>Shift out n |
| | Update to n |
| **Result:**<br>**Boundary write registers in state n**<br>**Result in read register window: n-1** | **Result:**<br>**Boundary write registers in state n**<br>**Result in read register window: n** |

# Execute Level and Connection Tests

## Level Tests

The current signal levels can be checked by executing *SAMPLE* mode. There are several ways to do this:

- Set all required ICs in *SAMPLE* mode:

    - Select **SAMPLE** from the context menu for each IC in the **BSDL.state** window <u>or</u>

    - Select **SAMPLE** in the instruction list box in the **BSDL.SET** window for each required IC and click the **RUN** button on tab **Run** of the **BSDL.state** window.

- Click the **SAMPLEall** button on the **Check** tab of the **BSDL.state** window.

The results can be observed in the **BSDL.SET** windows of the respective ICs (*DR mode: Sample*).
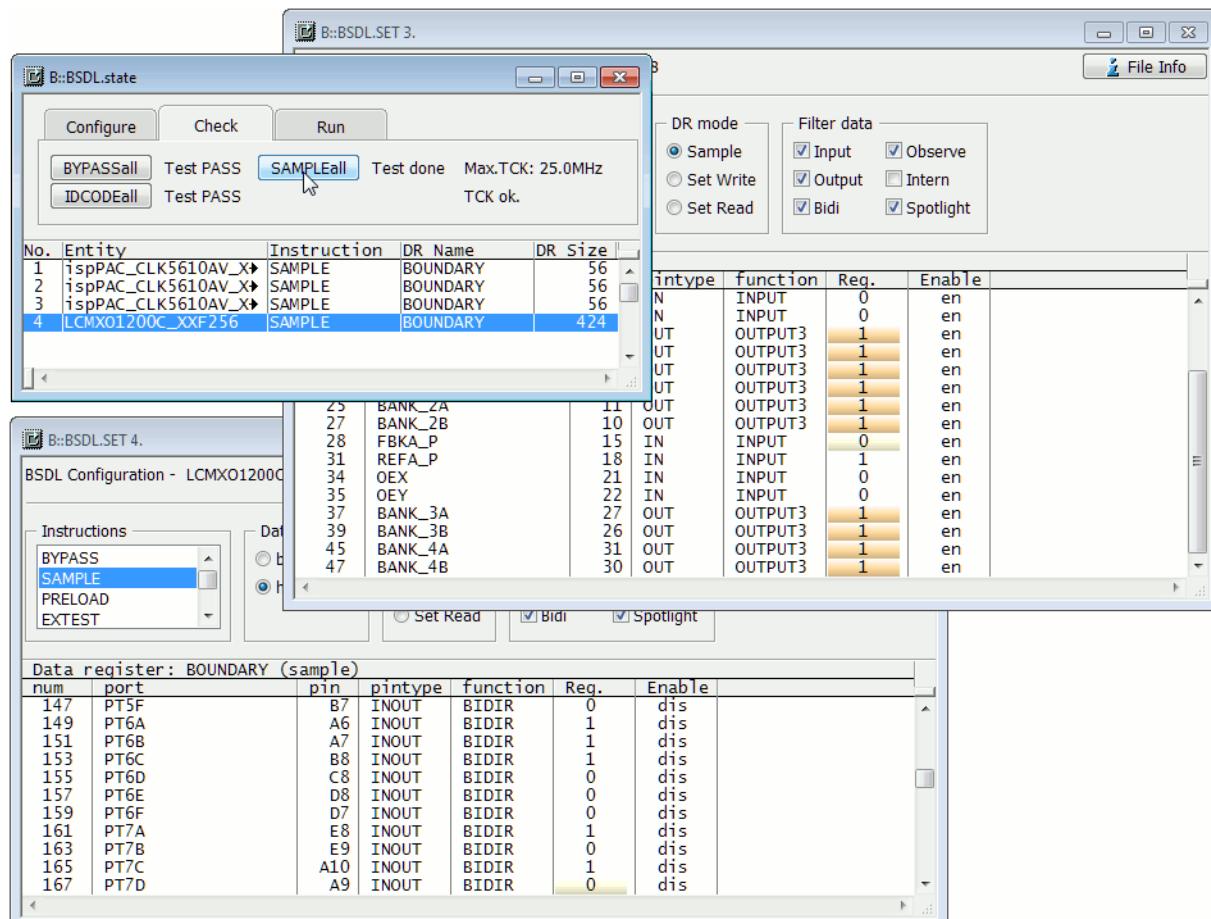


The *Reg.* column in the data area of the **BSDL.SET** window shows the sampled register value, the *Enable* column shows the enable state of the register bit:

- Register function *OUTPUT2* or *OUTPUT3*:

  *en*: Output driver is enabled

  *dis*: output driver is disabled

- Register function *BIDIR*:

  *en*: Output driver is enabled, cell operates as a driver (output)

  *dis*: Output driver is disabled, cell operates as a receiver (input)

- Other register functions are always *en*

## Connection Tests

Connection between ICs with boundary scan registers can be tested, by setting the signal driving IC into *EXTEST* mode and the signal receiving IC into *SAMPLE* mode.

| | |
|---|---|
| ⚠ | Caution:<br>To avoid the risk of damaging the board, the boundary scan register of any signal driving IC (*EXTEST* mode) must be set to a safe state before applying any register settings, e.g. with **BSDL.SET** *<chip_number>* **BSR * SAFE**.<br>Be sure to initialize all driving ICs before enabling the *Set and Run* mode. |

The recommended preparation procedure for interactive connection tests is:

1. Disable the *Set and Run* mode for the first initialization of *EXTEST* mode.

2. Set the required ICs into *SAMPLE* or *EXTEST* mode.

3. Initialize all ICs in *EXTEST* mode to a safe state.

4. Enable the *Two step DR* mode.

5. Enable the *Set and Run* mode.

6. Opens the **BSDL.SET** windows of the required ICs.

7. Set *DR mode* of the receiving ICs (*SAMPLE* mode) to *Sample.*

8. Set *DR mode* of the driving ICs (*EXTEST* mode) to *Set Write.*

9. Set *Filter data* as required.

After this preparation procedure, the connections can be tested. In the following example, there exists a direct connection from port PL7A of IC4 to port PS1 of IC3:



IC4

Enable output driver of port PL7A and toggle signal level

IC3

Observe the signal changes on port PS1 according to the driving signal

# Automated Board Test

For automated tests of the PCB's interconnects, a script with the following steps is required:

1.    System setup

2.    Configure boundary scan chain

3.    Check boundary scan chain

4.    Prepare boundary scan chain

5.    Run tests

It is recommended to bring the system in down state, to avoid interferences from the debugger to the boundary scan tests. The following example will do the first 3 steps:

```
;System setup
SYStem.Down
SYStem.JtagClock 20MHz
BSDL.RESet
BSDL.TwoStepDR OFF
BSDL.SetAndRun OFF

;Configure boundary scan chain
BSDL.FILE ispCLOCK5610Av_isc.bsm
BSDL.FILE ispCLOCK5610Av_isc.bsm
BSDL.FILE ispCLOCK5610Av_isc.bsm
BSDL.FILE LCMXCO1200C_ftBGA256.bsdl

;Check boundary scan chain
BSDL.HARDRESET
BSDL.SOFTRESET

IF !BSDL.CHECK.BYPASS()
(
    BSDL.BYPASSall
    PRINT %ERROR "Bypass test failed"
    ENDDO
)
IF !BSDL.CHECK.IDCODE()
(
    BSDL.IDCODEall
    PRINT %ERROR "ID code test failed"
    ENDDO
)
;prepare boundary scan tests...
;run tests...
```

| NOTE: | The *Two Step DR* mode and the *Set and Run* mode must be turned off for automated tests (**BSDL.TwoStepDR OFF**, **BSDL.SetAndRun OFF**). |
|---|---|

# Prepare Boundary Scan Chain

The preparation of the boundary scan chain includes the following steps:

- Set all ICs, which are not required for the board test, to *BYPASS* mode

- Set receiving ICs to *SAMPLE* mode and initialize to expected value X

- Set driving ICs in *PRELOAD* mode

- Initialize driving ICs to a safe state

- Enable result checking

These steps can be programmed as follows:

```
;Prepare boundary scan chain:
;Set unused ICs in BYPASS mode, set receiving IC in SAMPLE mode
BSDL.SET 1. IR BYPASS
BSDL.SET 2. IR BYPASS
BSDL.SET 3. IR SAMPLE
BSDL.SET 3. BSR * ExpectX
;Initialize signal driving IC
BSDL.SET 4. IR PRELOAD
BSDL.SET 4. BSR * SAFE
;Enable result checking
BSDL.CHECK ON
```

# Run Tests

A test consists of individual test cycles. A test cycle will sample the results from the previous cycle and apply the driving signals for the next cycle:

```
;Cycle n
;Prepare receiver: expected values from previous cycle
BSDL.SET 3. PORT PS1         <expect [n-1]>
BSDL.SET 3. PORT PLL_BYPASS <expect [n-1]>
;Prepare driver: apply new drive signals
BSDL.SET 4. PORT PL7A <drive [n]>
BSDL.SET 4. PORT PL6D <drive [n]>
;Execute test: Sample cycle n-1 results, apply cycle n settings
BSDL.RUN DR
```

All **BSDL.SET** *<chip_number>* **PORT** *<port_name>* commands are incremental, i.e. if a port is set to e.g. *H* in one cycle and is not modified in the next cycles, this port will always be checked for high signal level. To turn off the signal level check for a certain port, use this command: **BSDL.SET** *<chip_number>* **PORT** *<port_name>* **X**

In automated board test scripts it is recommended to use the command **BSDL.SET** *<chip_number>* **PORT** *<port_name> <value>* to modify the boundary scan registers. But the command **BSDL.SET** *<chip_number>* **BSR** *<bitslice> <value>* can also be used.

If a cycle runs without errors, this message will be printed to the **AREA** window for each IC which has set expected values:

Check IC*<n>* BOUNDARY: PASS

If an error was found, an error message is printed:

Check IC*<n>* BOUNDARY: FAIL

The execution of the automated board test script stops after the first error.

# Full Example

In the following example two direct unidirectional connections from port PL7A of IC4 to port PS1 of IC3 and from port PL6D of IC4 to port PLL_BYPASS of IC3 will be tested.

```
;Example for PCB Test
;Check unidirectional connection IC4.PL7A -> IC3.PS1
;Check unidirectional connection IC4.PL6D -> IC3.PLL_BYPASS
;
;System setup
SYStem.Down
SYStem.JtagClock 20MHz
BSDL.RESet
BSDL.TwoStepDR OFF
BSDL.SetAndRun OFF

;Configure boundary scan chain
BSDL.FILE ispCLOCK5610Av_isc.bsm
BSDL.FILE ispCLOCK5610Av_isc.bsm
BSDL.FILE ispCLOCK5610Av_isc.bsm
BSDL.FILE LCMXCO1200C_ftBGA256.bsdl

;Check boundary scan chain
BSDL.HARDRESET
BSDL.SOFTRESET

IF !BSDL.CHECK.BYPASS()
(
    BSDL.BYPASSall
    PRINT %ERROR "Bypass test failed"
    ENDDO
)
IF !BSDL.CHECK.IDCODE()
(
    BSDL.IDCODEall
    PRINT %ERROR "ID code test failed"
    ENDDO
)


;Prepare boundary scan chain:
;Set unused ICs in BYPASS mode
BSDL.SET 1. IR BYPASS
BSDL.SET 2. IR BYPASS
;Initialize signal receiving IC
BSDL.SET 3. IR SAMPLE
BSDL.SET 3. BSR * ExpectX
;Initialize signal driving IC
BSDL.SET 4. IR PRELOAD
BSDL.SET 4. BSR * SAFE
;Enable result checking
BSDL.CHECK ON
```

```
;Cycle 0: First cycle -> no expected values, switch drivers to EXTEST
;Prepare driver: drive 00
BSDL.SET 4. PORT PL7A 0
BSDL.SET 4. PORT PL6D 0
BSDL.RUN
;Execute test: Apply initial driver settings
BSDL.SET 4. IR EXTEST
BSDL.RUN IR

;Cycle 1
PRINT "Check connections @Low"
; Prepare receiver: expect Low Low
BSDL.SET 3. PORT PS1        L
BSDL.SET 3. PORT PLL_BYPASS L
;Prepare driver: drive 11
BSDL.SET 4. PORT PL7A 1
BSDL.SET 4. PORT PL6D 1
;Execute test: Sample cycle 0 results, apply cycle 1 settings
BSDL.RUN DR

;Cycle 2
PRINT "Check connections @High"
;Prepare receiver: expect High High
BSDL.SET 3. PORT PS1        H
BSDL.SET 3. PORT PLL_BYPASS H
;Prepare driver: drive 01
BSDL.SET 4. PORT PL7A 0
BSDL.SET 4. PORT PL6D 1
;Execute test: Sample cycle 1 results, apply cycle 2 settings
BSDL.RUN DR

;Cycle 3
PRINT "Check connections alternating  @Low@High"
;Prepare receiver: expect Low High
BSDL.SET 3. PORT PS1        L
BSDL.SET 3. PORT PLL_BYPASS H
;Prepare driver: drive 10
BSDL.SET 4. PORT PL7A 1
BSDL.SET 4. PORT PL6D 0
;Execute test: Sample cycle 2 results, apply cycle 3 settings
BSDL.RUN DR

;Cycle 4: Last cycle -> no new drive signals
PRINT "Check connections alternating  @High@Low"
;Prepare receiver: expect High Low
BSDL.SET 3. PORT PS1        H
BSDL.SET 3. PORT PLL_BYPASS L
;Execute test: Sample cycle 3 results
BSDL.RUN DR
;Test finished, print PASS result
PRINT ""
PRINT "Test Done (PASS)."
ENDDO
```

# Test of Non-Boundary Scan Devices

Non-boundary scan devices can be tested, when all inputs can be controlled by boundary scan enabled devices and all outputs are connected to boundary scan enabled devices.

**IC1**
(with boundary scan)

**IC2**
(with boundary scan)

## NAND

Port1_A — A

Port1_B — B — Q

Port2_A

Port2_B

.

.

| Input A | Input B | Output Q |
|---------|---------|----------|
| 0 | 0 | 1 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 0 |

## D-FlipFlop

Port1_C

Port1_D — D    Q

Port1_E

Port1_F — >    $\overline{Q}$

Port1_G       R

Port2_C

Port2_D

Port2_E

Port2_F

Port2_G

| Input R | Input D | Clock | Output Q | Output $\overline{Q}$ | Output $Q_{NEXT}$ | Output $\overline{Q}_{NEXT}$ |
|---------|---------|-------|----------|-----------------------|-------------------|------------------------------|
| 1 | X | X | 1 | 0 | - | - |
| 0 | 0 | ↑ | $Q_{PREV}$ | $\overline{Q}_{PREV}$ | 0 | 1 |
| 0 | 1 | ↑ | $Q_{PREV}$ | $\overline{Q}_{PREV}$ | 1 | 0 |
| 0 | X | Non rising | Q | $\overline{Q}$ | - | - |
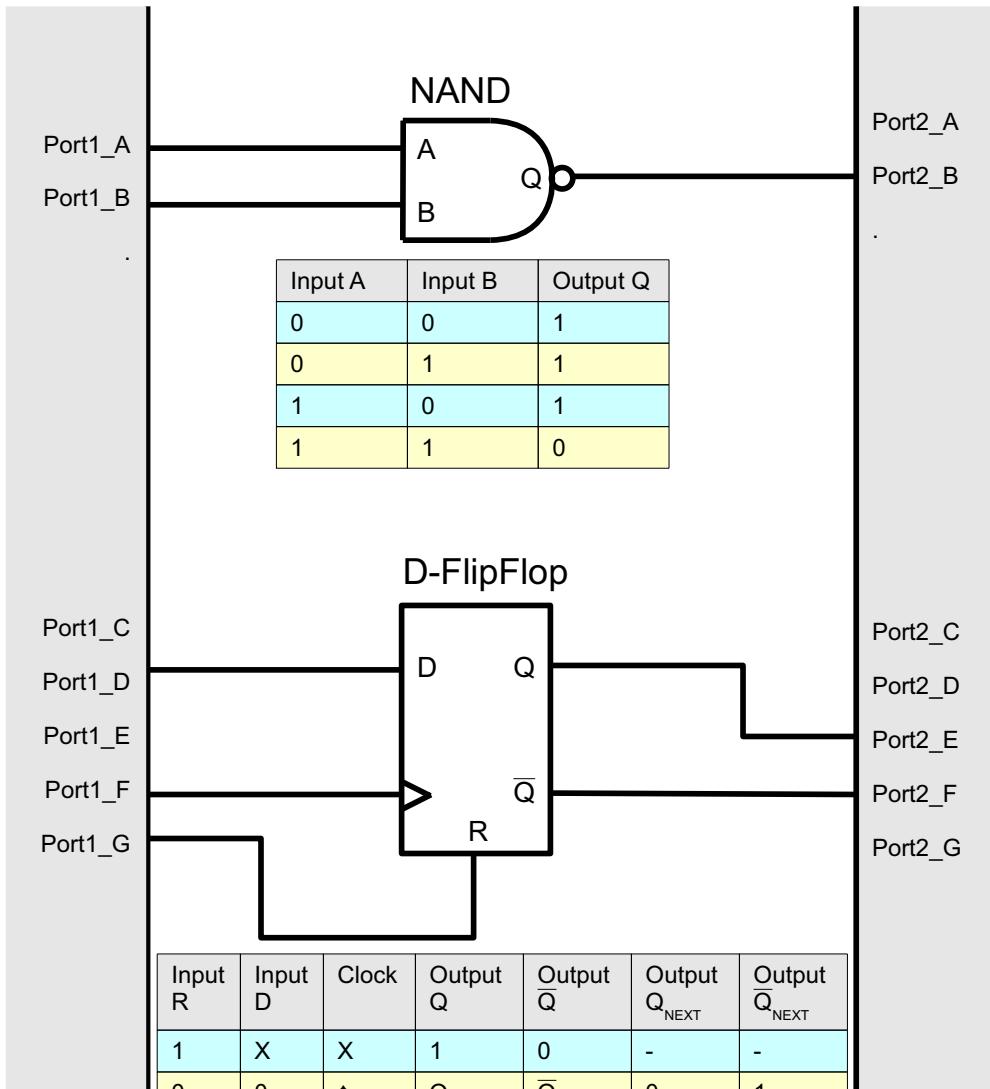
The following example will test the NAND gate and the D-flip-flop:

```
; System setup
SYStem.Down
BSDL.RESet

; Configuration of the boundary scan chain
BSDL.FILE IC1.bsdl   ; BSDL file fo IC1
BSDL.FILE IC2.bsdl   ; BSDL file fo IC2

; Check boundary scan chain
BSDL.SOFTRESET

IF !BSDL.CHECK.BYPASS()
(
    BSDL.BYPASSall
    PRINT %ERROR "Bypass test failed"
    ENDDO
)
IF !BSDL.CHECK.IDCODE()
(
    BSDL.IDCODEall
    PRINT %ERROR "ID code test failed"
    ENDDO
)

;Prepare boundary scan chain:
;set driving IC in EXTEST, receiving IC in SAMPLE mode
BSDL.SET 1. IR EXTEST
BSDL.SET 2. IR SAMPLE

;Initialize signal driver and receiver
BSDL.SET 1. BSR * SAFE
BSDL.SET 2. BSR * ExpectX
;Enable result checking
BSDL.CHECK ON


;Test the NAND gate
PRINT "Test NAND gate"
;Prepare driver: drive A:0 B:0
BSDL.SET 1. PORT Port1_A 0        ;Input A
BSDL.SET 1. PORT Port1_B 0        ;Input B
BSDL.RUN

;Sample previous result (A:0 & B:0), drive A:0 B:1
PRINT "    check 0 & 0"
BSDL.SET 2. PORT Port2_B ExpectH  ;Output Q
BSDL.SET 1. PORT Port1_A 0
BSDL.SET 1. PORT Port1_B 1
BSDL.RUN
```

```
        ;Sample previous result (A:0 & B:1), drive A:1 B:0
        PRINT "      check 0 & 1"
        BSDL.SET 2. PORT Port2_B ExpectH
        BSDL.SET 1. PORT Port1_A 1
        BSDL.SET 1. PORT Port1_B 0
        BSDL.RUN


        ;Sample previous result (A:1 & B:0), drive A:1 B:1
        PRINT "      check 1 & 0"
        BSDL.SET 2. PORT Port2_B ExpectH
        BSDL.SET 1. PORT Port1_A 1
        BSDL.SET 1. PORT Port1_B 1
        BSDL.RUN


        ;Sample previous result (A:1 & B:1)
        PRINT "      check 1 & 1"
        BSDL.SET 2. PORT Port2_B ExpectL
        BSDL.RUN
        ;Test NAND gate finished
        BSDL.SET 2. PORT Port2_B ExpectX


        ;Test the D-FlipFlop
        PRINT "Test D-FlipFlop"
        ;       Prepare driver: D:1 Clock:0 R:1
        BSDL.SET 1. PORT Port1_D 1          ;Input D
        BSDL.SET 1. PORT Port1_F 0          ;Clock
        BSDL.SET 1. PORT Port1_G 1          ;Input R
        BSDL.RUN


        ;Sample reset
        PRINT "      check reset"
        BSDL.SET 2. PORT Port2_E ExpectH  ;Output Q
        BSDL.SET 2. PORT Port2_F ExpectL  ;Output ^Q
        BSDL.RUN


        ;do a clock cycle, output should not change
        BSDL.SET 1. PORT Port1_F 1
        BSDL.RUN
        BSDL.SET 1. PORT Port1_F 0
        BSDL.RUN


        ;set D to 0 and do a clock cycle, output should not change
        BSDL.SET 1. PORT Port1_D 0
        BSDL.RUN
        BSDL.SET 1. PORT Port1_F 1
        BSDL.RUN
        BSDL.SET 1. PORT Port1_F 0
        BSDL.RUN


        ;release reset and test data
        PRINT "      check D:0"
        BSDL.SET 1. PORT Port1_G 0
        BSDL.RUN
```

```
;Input D already 0 from previous test
;do a clock cycle, output should change after rising clock
BSDL.SET 1. PORT Port1_F 1
BSDL.RUN
BSDL.SET 2. PORT Port2_E ExpectL
BSDL.SET 2. PORT Port2_F ExpectH
BSDL.SET 1. PORT Port1_F 0
BSDL.RUN
PRINT "    check D:1"

;set D:1
BSDL.SET 1. PORT Port1_D 1
BSDL.RUN

;do a clock cycle, output should change after rising clock
BSDL.SET 1. PORT Port1_F 1
BSDL.RUN
BSDL.SET 2. PORT Port2_E ExpectH
BSDL.SET 2. PORT Port2_F ExpectL
BSDL.SET 1. PORT Port1_F 0
BSDL.RUN

;Test D-FlipFlop finished
BSDL.SET 2. PORT Port2_E ExpectX
BSDL.SET 2. PORT Port2_F ExpectX
BSDL.CHECK OFF


;Test finished, print PASS result
PRINT ""
PRINT "Test Done (PASS)."
ENDDO
```

# Special Tests

This chapter describes how to:

- Use the boundary scan oscilloscope

- Access IC specific data registers.

## Boundary Scan Oscilloscope

The following command can be used to enable a simple boundary scan oscilloscope:

| | |
|---|---|
| **BSDL.SET** *<chip_number>* **OPTION BSRHISTORY ON** | Enables the boundary scan oscilloscope for IC *<chip_number>* |

The command stores up to 32 samples for each bit of the boundary scan register. To view the waveforms in the **BSDL.SET** window, select **Sample** in the DR mode box.
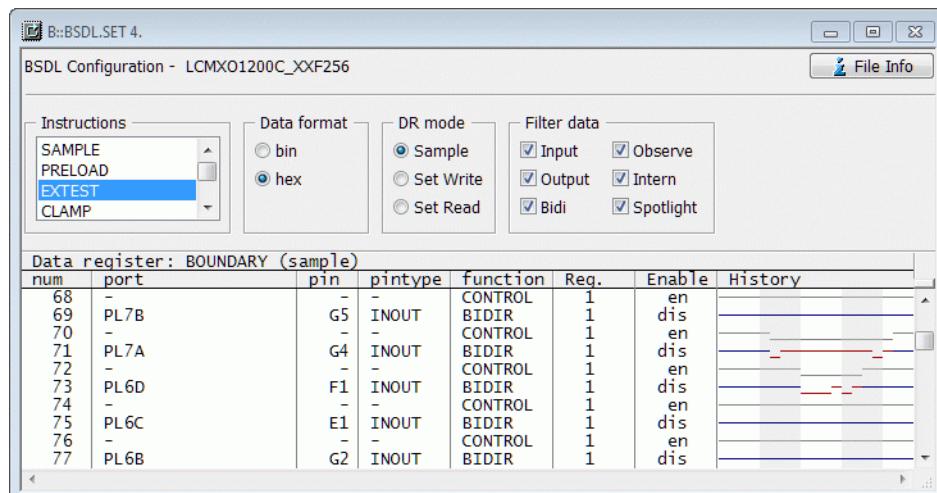


Each time a **BSDL.RUN** or **BSDL.RUN DR** is command is executed, a sample is stored. The background color of the history area changes its color every 5 samples. The signal colors provide information about the IO direction of the port:

- Blue: port is in input mode.

- Red: port is in output mode.

- Gray: port is in Z state or register bit is an internal cell.

To clear the history area, disable the **BSRHISTORY** option:

```
BSDL.SET 3. OPTION BSRHISTORY OFF      ;Disables the boundary scan scope
```

# Other Instructions and Data Registers

Each instruction defined in the BSDL file can be set and the associated data register can be read and written to. This is useful, for example, if the IC provides additional functionality or test functions via the JTAG interface. The BSDL file can only provide information about the instruction names, instruction codes and the name and size of associated data registers. The meaning of the data register bits must be obtained from other sources such as the chip manufacturer's documentation.

| | A *DR-Scan* operation always includes an *Update-DR* with the shifted-in data. This could cause unintended behavior, when the meaning of the write data register bits are unknown. |
|---|---|

In the **BSDL.SET** window, general data register masks are provided:
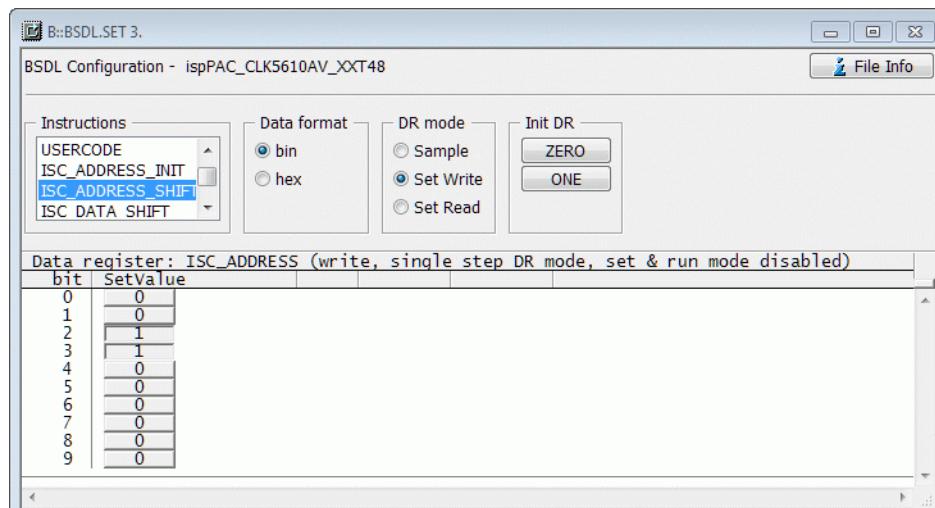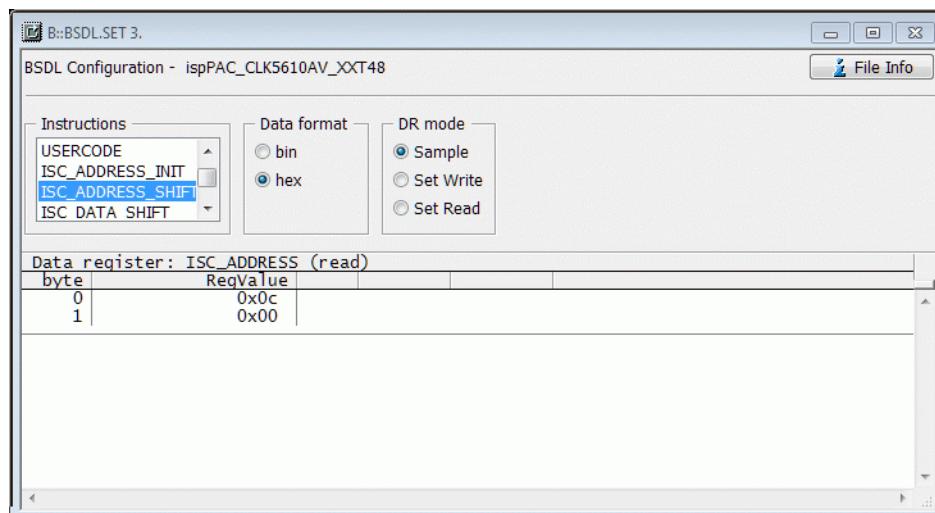
On the command line the data register bits of a non-boundary scan register can be modified with the command:

**BSDL.SET** *<chip_number>* **DR** *<bitslice>* *<setvalue>*     Set *<bitslice>* of current data register to value *<setvalue>* for IC *<chip_number>*

When result checking is enabled (**BSDL.CHECK ON**) and the expected values are set for the data register, a result message is printed to the **AREA** window after each **BSDL.RUN** or **BSDL.RUN DR** operation:

Check IC*<n>* *<data_registername>*: PASS | FAIL

with

* *<n>*: Number of the IC

* *<data_registername>*: Name of the current data register

NOTE:          If a script is used for result checking, script execution stops after the first fail.

# Tips and Tricks

To improve the readability of the data area in the **BSDL.SET** window, you can use the command:

| | |
|---|---|
| **BSDL.SET** *<chip_number>* **OPTION MARKLINES ON** | Enables alternating line colors for IC *<chip_number>* |