AN13806

How to Perform Boundary Scan for LPC86x Based on µTrace and Trace32

Rev. 0 — 8 May 2023 Application note

Document Information

Information	Content
Keywords	LPC86x, Boundary Scan, BSDL
Abstract	This document describes how to enter the boundary scan mode, how to perform boundary scan test based on Boundary-Scan Description Language (BSDL) on LPC86x series and provides an overview of JTAG and boundary scan technology.

1 Overview

LPC86x is a microcontroller based on Arm Cortex-M0+ for embedded applications. It supports Joint Test Action Group (JTAG) boundary scan. This document:

- Describes how to enter the boundary scan mode.
- Describes how to perform boundary scan test based on Boundary-Scan Description Language (BSDL) on LPC86x series.
- Provides an overview of JTAG and boundary scan technology.

To understand this document better, basic knowledge of JTAG and boundary scan is required.

2 JTAG and boundary scan

2.1 Introduction

JTAG/boundary scan is an interface containing four ports. The interface allows access to the special embedded logic on most chips. JTAG/boundary can provide several functions that can contain any or all of the following:

- Probe-less device connectivity test.
- Logic programming for Flash memory, CPLD, and FPGA.
- Debug logic in microprocessors and microcontrollers used for software debugging, or test connections with peripheral devices at speed without embedded software.

2.2 Development history

Architecture for Test Access Port (TAP) and Boundary Scan is defined in IEEE Std 1149.1. The development history of this standard is summarized as follows:

- 1985 Joint European Test Action Group (JETAG) was formed.
- 1986 Joint European Test Action Group (JETAG) was renamed as Joint Test Action Group (JTAG).
- 1986-1988 JTAG Technical Subcommittee developed and published a series of proposals for a standardized form of boundary scan.
- 1988 the last of these proposals, JTAG Version 2.0, was offered to the IEEE Testability Bus Standards Committee (P1149) and was accepted by P1149. JTAG proposal became the basis of a standard within the Testability Bus family.
- 1990 From 1990, JTAG developed a supplement for correction, clarification, and enhancement.
- 1993 IEEE Std 1149.1aTM-1993
- 1994 IEEE Std 1149.1b-1994
- 2001 IEEE Std 1149.1-2001
- 2013 IEEE Std 1149.1-2013

2.3 Basic principle

Boundary scan is a method for testing interconnects on PCBs and internal IC subblocks. For boundary scan tests, additional logic is added to the device. Boundary scan cells are placed between the core logic and the ports.

In boundary scan test, each primary input and output signal on a device is supplemented with a multipurpose memory element called as boundary scan cell. These cells are connected to a shift register, which is referred to as the boundary scan register. This register can be used to read and write port states.

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

Figure 1 shows the basic principle of boundary scan.

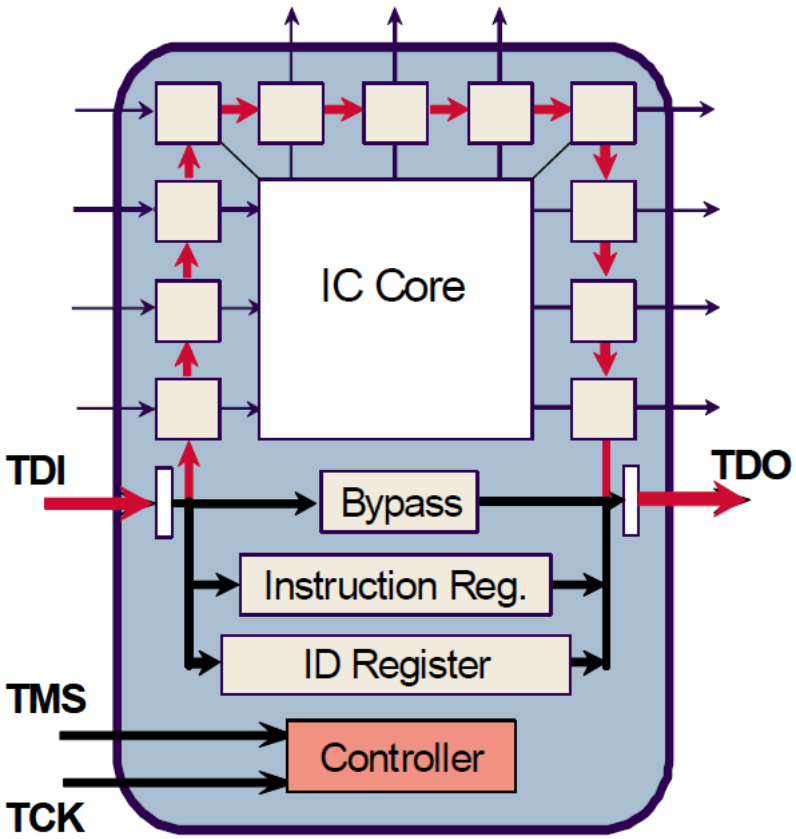

Figure 1. Boundary scan basic principle

2.4 Instruction set

Table 1 describes boundary scan instructions defined in the IEEE Std 1149.1.

Table 1. Standard instruction set

Instruction	Mandatory/Optional	Description
BYPASS	Mandatory	TDI is connected to TDO via a single shift register.
SAMPLE	Mandatory	Takes a snapshot of the normal operation of the Integrated Circuit (IC).
PRELOAD	Mandatory	Loads data to the boundary scan register.
EXTEST	Mandatory	Applying preloaded data of the boundary scan register to the ports.
INTEST	Optional	Applying preloaded data of the boundary scan register to the core logic.
RUNBIST	Optional	Executing a self-contained self-test of the IC.
CLAMP	Optional	Applying 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	Reading the device identification register.

Table 1. Standard instruction set*...continued*

Instruction	Mandatory/Optional	Description
USERCODE	Optional	Reading and writing a user programmable identification register.
HIGHZ	Optional	Placing the IC in an inactive drive state (for example, all ports are set to high impedance state).

2.5 JTAG Test Access Port (TAP)

TAP is a general-purpose port that can provide access to many test support functions built into a component, including the test logic. It is composed as **a minimum of the three input connections** which are **TCK**, **TMS**, **TDI**, and **one output connection** which is **TDO**. An optional fourth input connection, $\overline{\text{TRST}}$, is provided for asynchronous initialization of the test logic.

Table 2 describes TAP signals.

Table 2. TAP signal description

Signal Name	I/O type	Description
TCK	Input	Providing the clock for the test logic.
TMS	Input	Value of the signal presented as TMS at the time of a rising edge at TCK determines the next state of the TAP controller. The circuit that controls test operations.
TDI	Input	Serial test instructions and data received by the test logic at TDI.
TDO	Output	Serial output for test instructions and data from the test logic.
$\overline{\text{TRST}}$	Input	Provided for asynchronous initialization and active low.

Note: *JTAG is used on LPC86x devices for boundary scan and the production test only. It cannot be used for debugging purposes.*

2.6 BSDL

BSDL is based on the syntax and grammar of Very high-speed integrated-circuit Hardware Description Language (VHDL). BSDL is not a general-purpose hardware description language. It describes key aspects to implement the boundary scan within a particular component.

Table 3 lists elements contained in the BSDL file.

Table 3. BSDL elements

Element	Description
Entity Description	Statement for device name or functionality
Generic Parameter	Description for package or pin mapping
Logical Port Description	Description for pin type such as in, out, inout, linkage
Standard Use Statement	References external definitions
Component Conformance Statement	Standards to follow
Device Package Pin Mapping	Description for pin mapping
Scan Port Identification	Pin description on device for JTAG TAP including TCK, TMS, TDI, and TDO.

Table 3. BSDL elements*...continued*

Element	Description
Compliance Enable Description	Pins involved in entering boundary scan mode and level applied to the pins. It is useful for chip enter boundary scan mode.
Instruction Register Description	Instruction length and instruction code. Sometimes it includes device-specific instruction, also called as private instruction.
Register Access Description	Registers corresponding to specific instructions
Boundary-Scan Register Description	List recording the boundary scan cells and functionality of these boundary scan cells.

2.7 More information on JTAG and boundary scan

For more information on JTAG and boundary scan, see the links below:

- Homepage for JTAG and boundary scan: https://www.jtag.com/
- IEEE Std 1149.1
  - version 1990: https://standards.ieee.org/standard/1149_1-1990.html
  - version 2001: https://standards.ieee.org/standard/1149_1-2001.html
  - version 2013: https://standards.ieee.org/standard/1149_1-2013.html

3 Building boundary scan test environment

3.1 Introduction to boundary scan test tool suite

In this application note, boundary scan test uses the tool set from Lauterbach which is all-in-one debug and trace solution for Cortex-M. This tool set includes the following two parts:

- **µTrace for Cortex-M**
  µTrace for Cortex-M is one of architecture-specific products from Lauterbach. The features are as below:
  - On-chip/external flash programming, debug, trace, and **JTAG boundary scan**.
  - Recommended for **single-core microcontrollers with Cortex-M**.
  - Recommended for **multi-core microcontrollers with solely Cortex-M** (single debug port)
  - 256 Mbytes trace memory
  - USB 3 interface to the host computer
  - TRACE32 streaming up to 150 Mbytes/s
  - TRACE32 mixed signal probe supported

Figure 2. µTrace for Cortex-M debugger

For more information about µTrace for Cortex-M, see Figure 3.

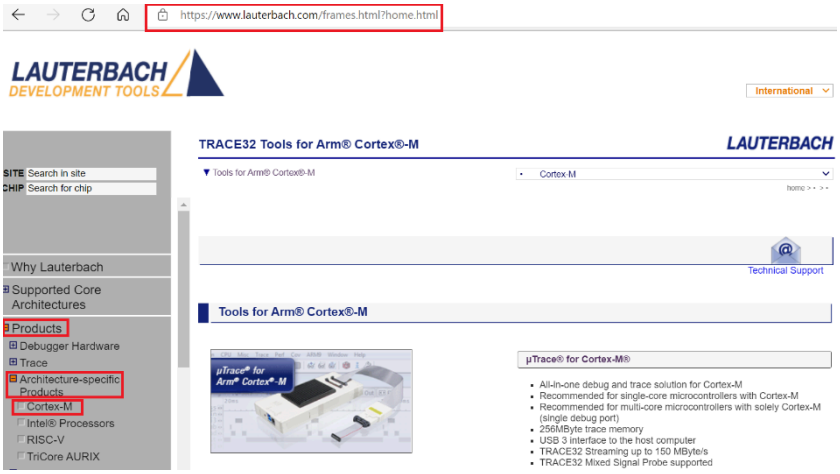

Figure 3. µTrace for Cortex-M introduction

- **TRACE32**

  TRACE32 is a simulation test tool developed by Lauterbach. It runs on PC. It is used with µTrace for Cortex-M for on-chip/external Flash programming, debugging, tracing, and JTAG boundary scan. It supports various processor architectures, including standard processors, such as, Arm, MIPS, PowerPC and DSP, soft cores, and coprocessors.

  For boundary scan, TRACE32 provides GUI operations for interactive test and supports script for automated test. To execute many commands, such as, system settings, JTAG, and BSDL, make a script that contains these commands. To complete the test, execute the script. The execution is efficient and reduces the possibility of errors in the command-line mode.

  TRACE32 supports command-line input. The command-line input is at the bottom of the TRACE32 main page, starting with **B::**. To complete the operations, input commands, such as, system reset, system settings, BSDL file loading, and boundary scan test.

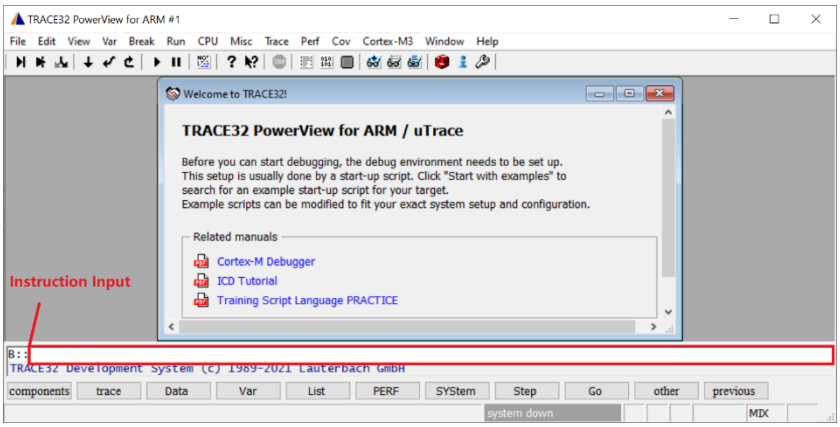

Figure 4. Main page for TRACE32 for Arm

To download TRACE32, see Figure 5.

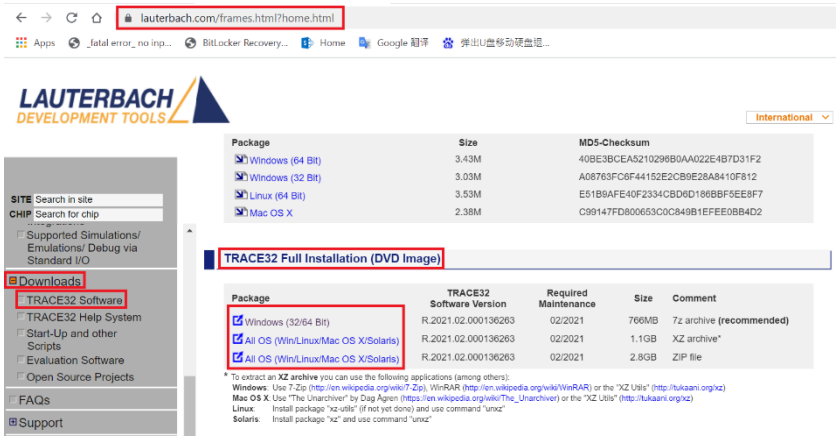

Figure 5. Downloading TRACE32

3.2 Hardware connection

µTrace for Cortex-M consists of:

- Universal debugger hardware
- Debug cable specific to the processor architecture

Figure 6 shows the schematic diagram of the hardware connection.

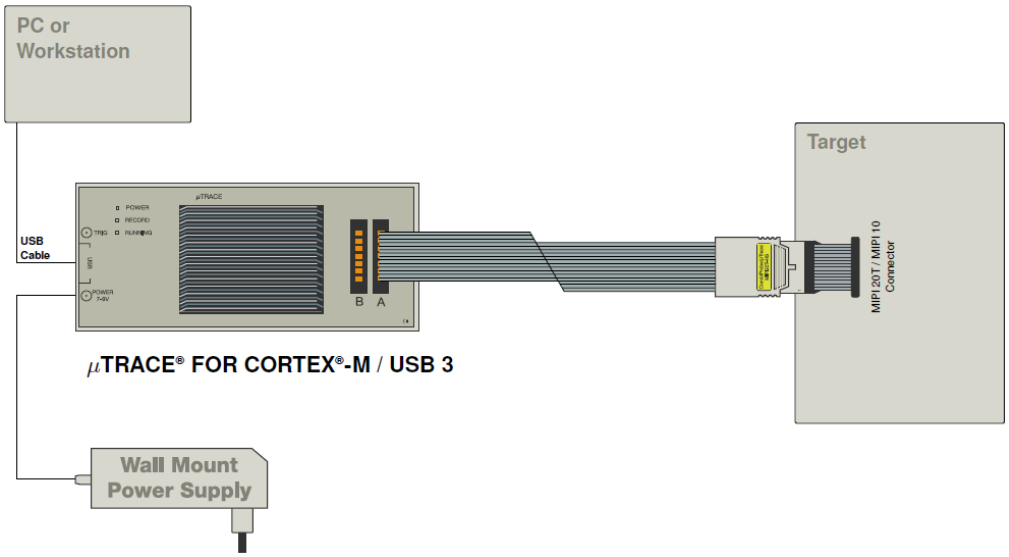

Figure 6. Schematic diagram of hardware connection

Below lists the suggestions for users:

- To prevent the debugger or target being damaged, do not plug or unplug the debugger while the target is powered on. The power on/off sequence is as below:
  - Power on: debugger > target
  - Power off: target > debugger
- To prevent damages to the debugger or target, double check the debugger interface direction by locating the pin1.

Taking LPCXpresso860-MAX board as an example:

1. Connect the µTrace for Cortex-M debugger to the MAX board through the MIPI20T JTAG interface.
2. Connect the µTrace for Cortex-M debugger to the PC through the USB cable. Power on the debugger with a 5 V power adapter. Open the Device Manager on PC. Lauterbach equipment appears in Trace32 devices, as shown in Figure 7. If Lauterbach equipment does not appear, check the connection.

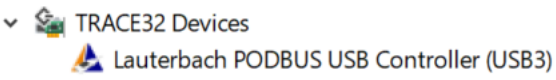

Figure 7. Lauterbach equipment appears in Device Manager

3. Connect the USB port marked as J4 to the PC to supply power to the board.

3.3 Entering boundary scan mode

For the LPC86x series, ***entering boundary scan mode*** can be achieved by ***continuously driving the RESET pin to LOW***.

To perform boundary scan testing, follow these steps:

1. Erase any user code residing in flash.
2. Power up the part with the RESET pin pulled HIGH externally.
3. Wait for at least 250 µs.
4. Pull the RESET pin LOW externally.
5. Perform boundary scan operations.

6. Once the boundary scan operations are completed, assert the $\overline{TRST}$ pin to enable the SWD debug mode and release the RESET pin (pull HIGH).

4 Interactive boundary scan test

To perform boundary scan test using µTrace for Cortex-M debugger and TRACE32 software, follow the steps below:

1. Open the TRACE32 software and choose **ARM32 USB**.

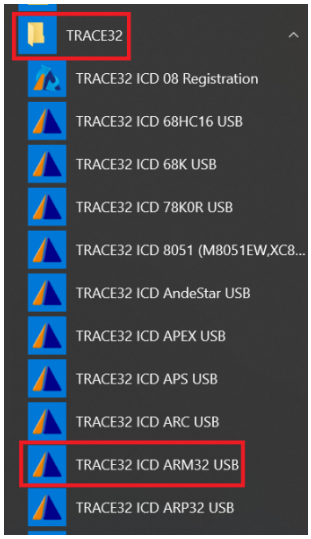

Figure 8. Opening TRACE32 for ARM32

2. Figure 9 shows the main page of TRACE32 for ARM32. If the status bar at the bottom of the main page shows **power down** instead of **system down**, check the power supply of the debugger and the connection with the JTAG interface of the MAX board.

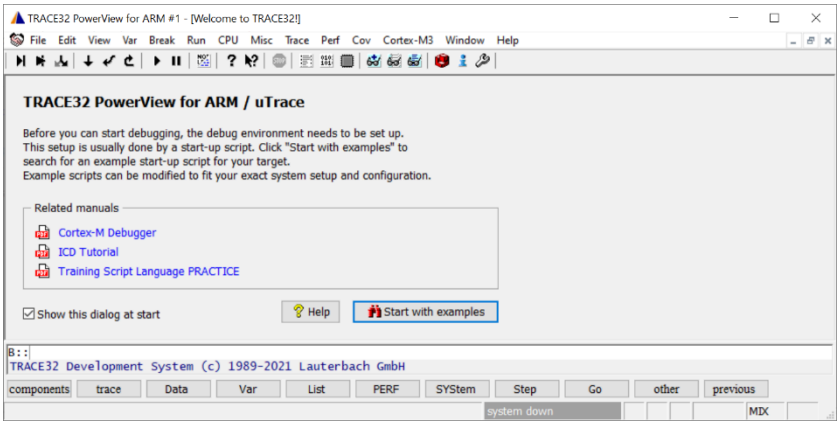

Figure 9. Main page of TRACE32 for ARM32

3. Click **CPU** -> **System Settings…** in the menu bar, and the system setting dialog appears. Perform system settings as shown in Figure 10.

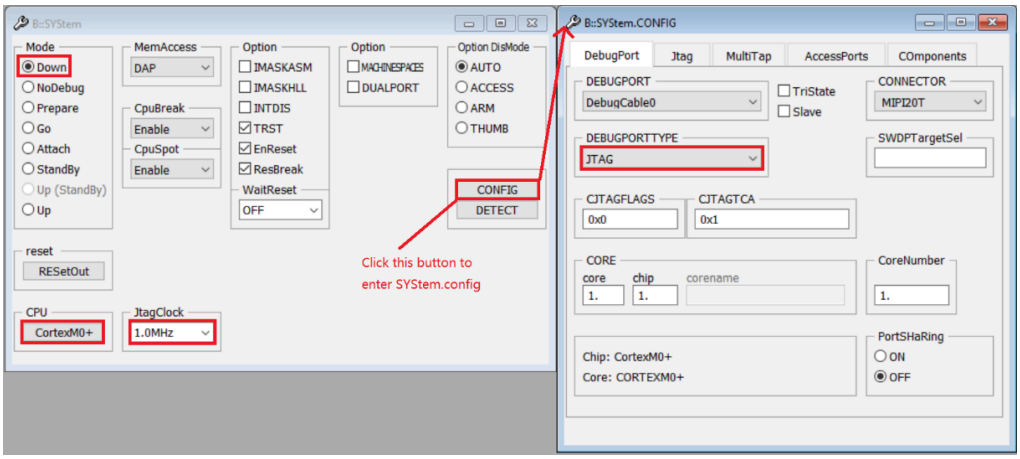

Figure 10. System settings

4. Type in below commands, with each command followed by entering:
   - `BSDL.RESet`
   - `BSDL.ParkState Select-DR-Scan`
   - `BSDL.state`
5. The **BSDL.state** window appears, as shown in Figure 11. Click the **FILE** button and load the BSDL file you want to validate.

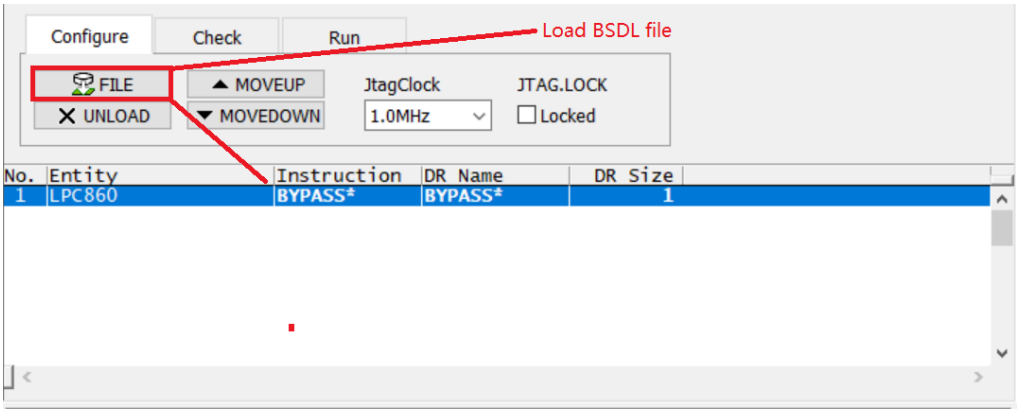

Figure 11. BSDL state window

6. After loading the file, type in command as below:
   - `BSDL.SOFTRESET`
7. Switch to the **Check** tab of the BSDL.state window. To see whether both results can pass, click **BYPASSall** and **IDCODEall**, as shown in Figure 12 and Figure 13. Double-click the entity name in Figure 13, and the IDCODE test result can be seen in the **BSDL.SET** window, as shown in Figure 14.

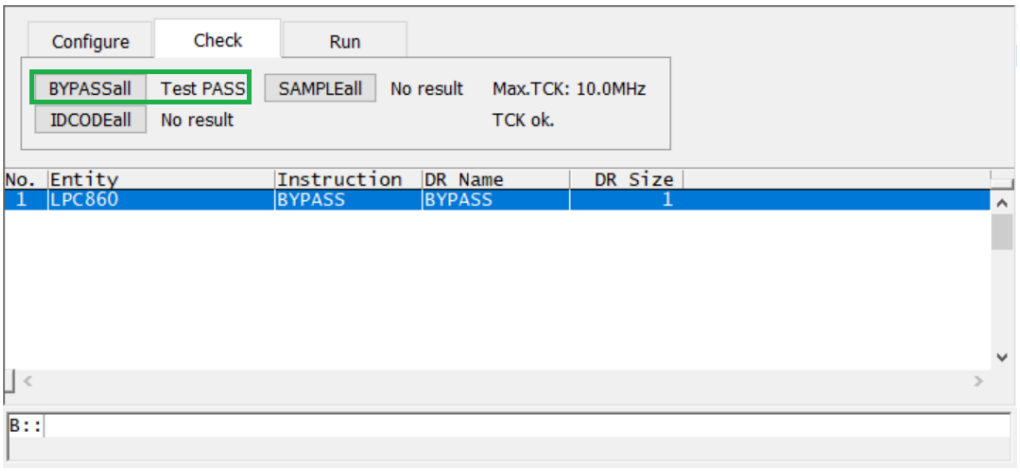

Figure 12. Checking BYPASS

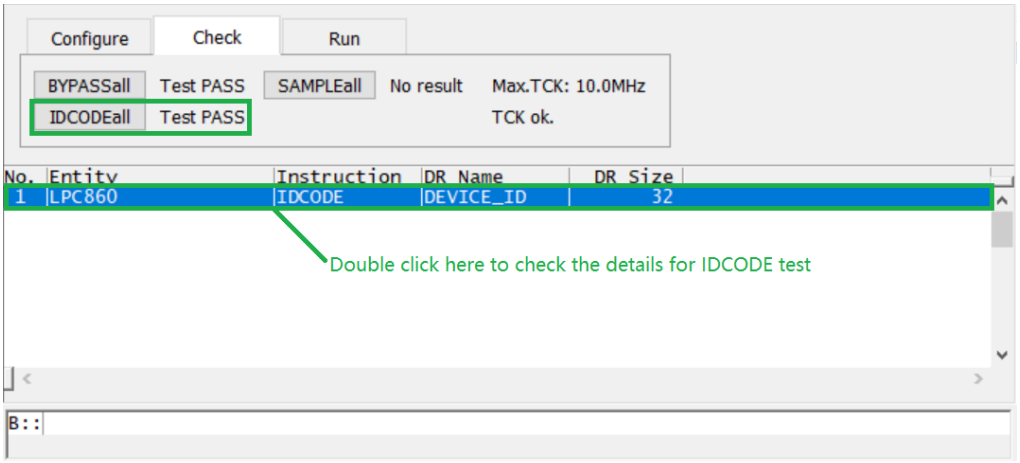

Figure 13. Checking IDCODE

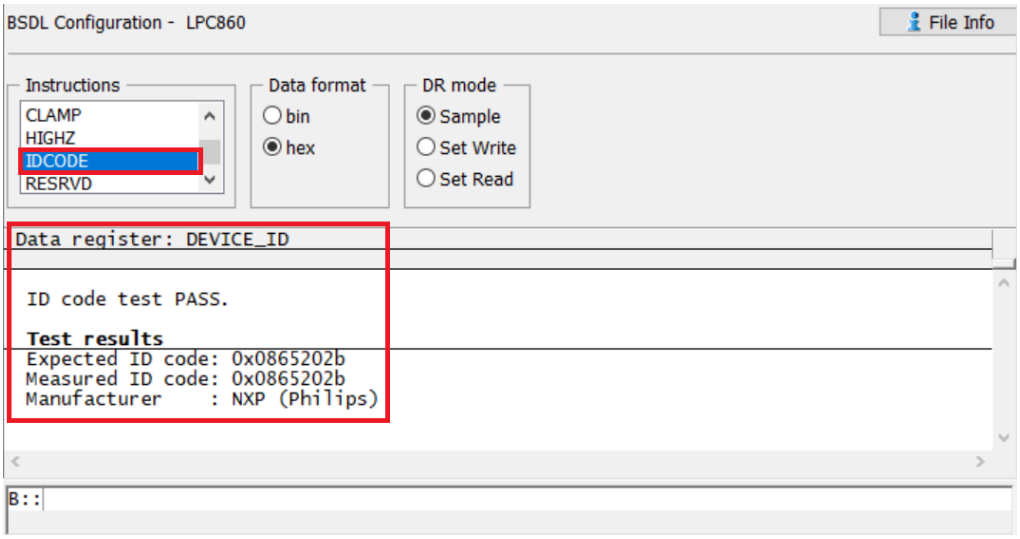

Figure 14. IDCODE test result

8. Click the **SAMPLEall** button, and **No result** becomes **Test done**. Double-click the entity name as shown in Figure 15, and the SAMPLE test result can be seen in the **BSDL.SET** window, as shown in Figure 16.

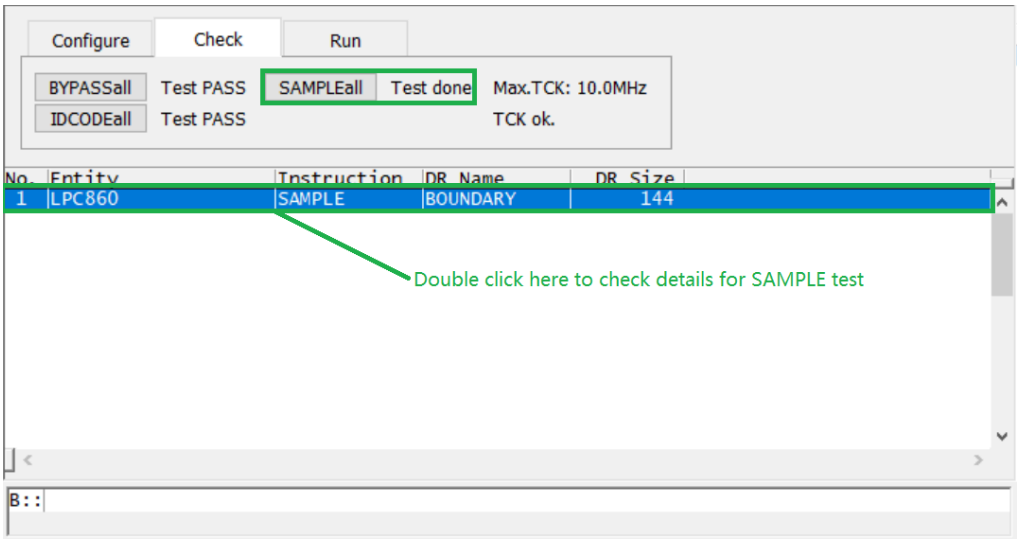

Figure 15. Checking SAMPLE

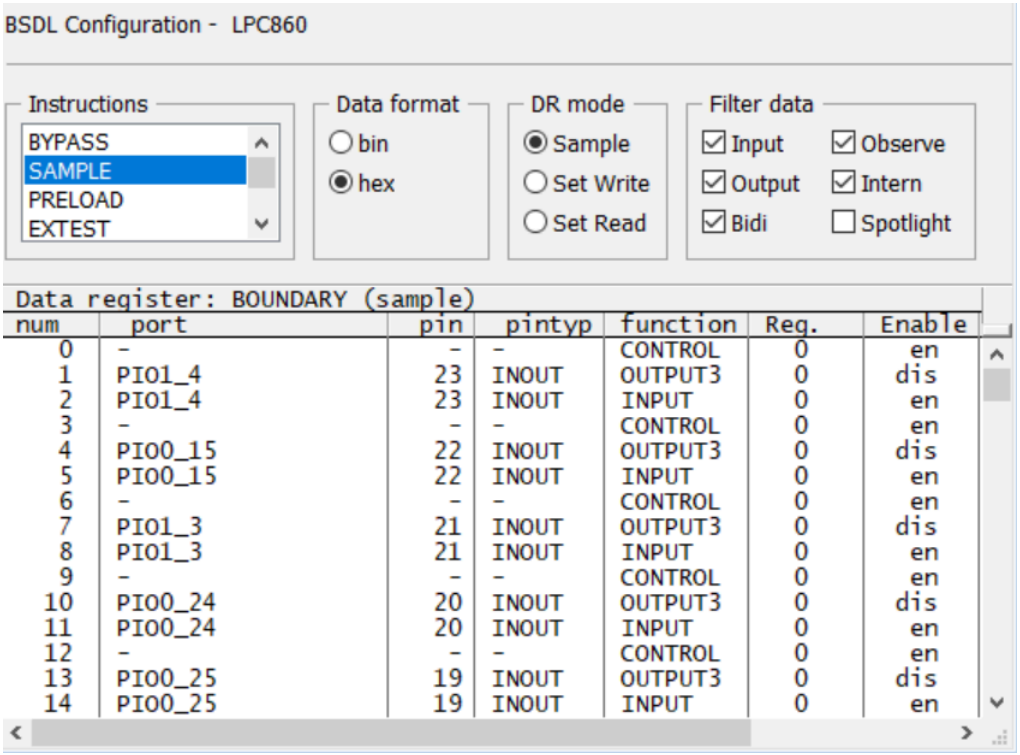

Figure 16. SAMPLE test result

To judge whether the SAMPLE test passes or not, take pin `P0_7` as an example to perform the following steps:

a. Apply high level to this pin. Use the BSDL.RUN command to run the SAMPLE test. Search the line where port is `PIO0_7` and function is INPUT. Read the register value in Reg. column and it is 1, as shown in Figure 17.

b. Apply low level to this pin. Check the sample result and it is 0, as shown in Figure 18.

Therefore, pin `P0_7` passes SAMPLE test.

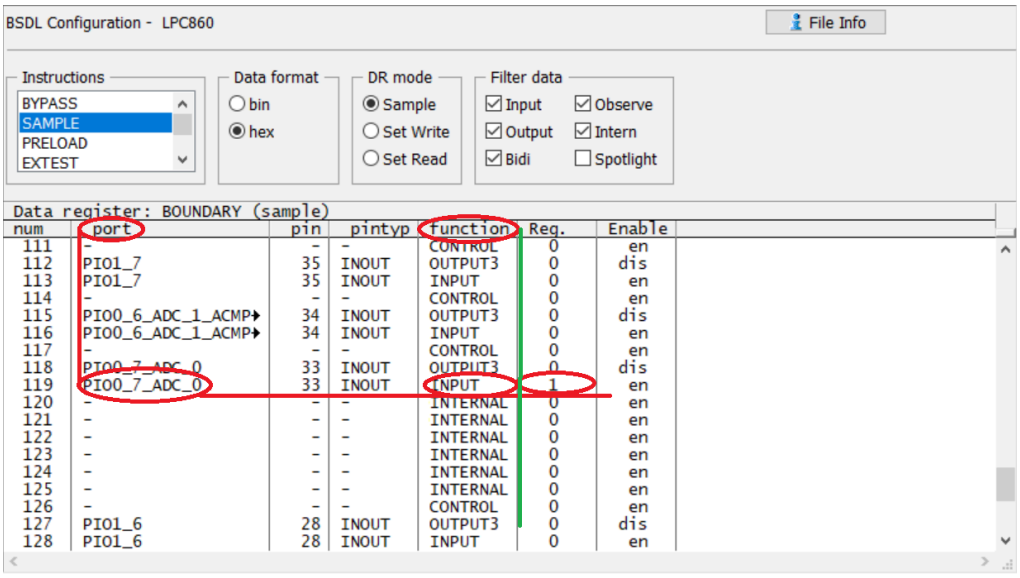

Figure 17. Sample high level on P0_7

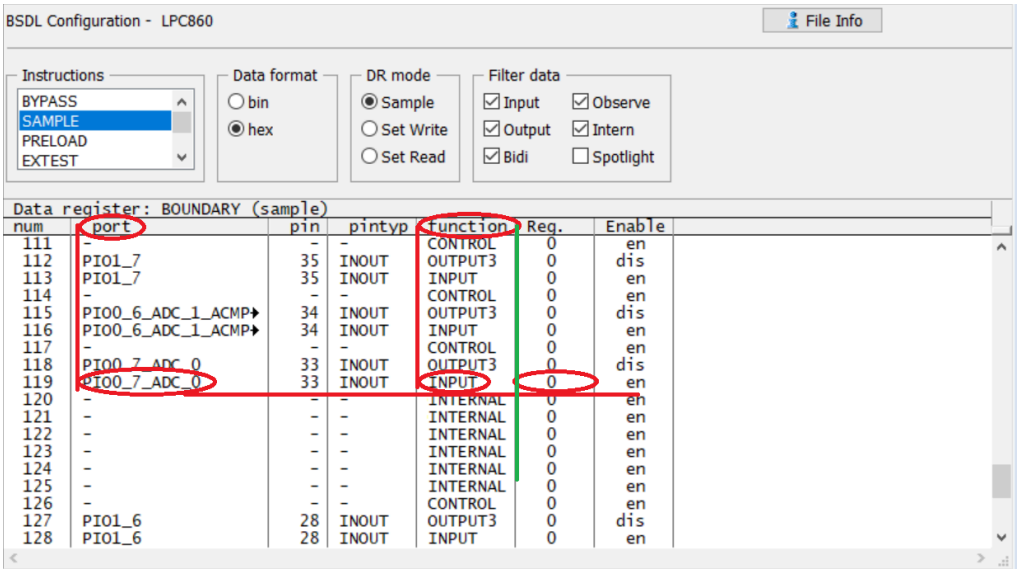

Figure 18. Sample low level on P0_7

Use the above method to traverse all the I/O pins defined in the BSDL file. If all the I/O pins pass the test, the SAMPLE test of BSDL passes.

9. Enter the `BSDL.SET` command on the **TRACE32** command line, and the **BSDL.SET** window appears. In the Instructions field, click **EXTEST** and in the **DR mode** field, choose **Set Write**, as shown in Figure 19.

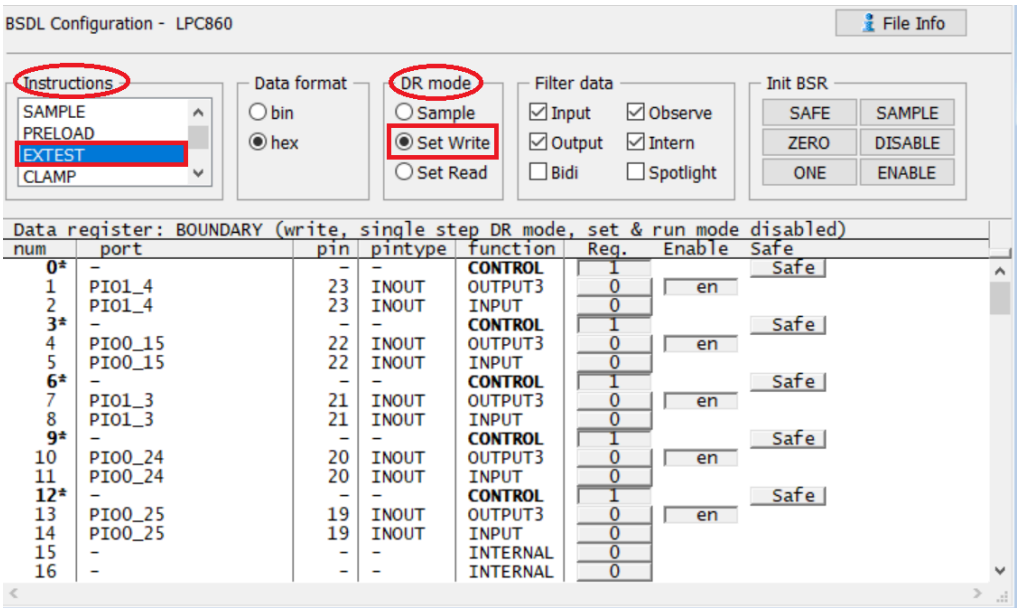

Figure 19. EXTEST settings in BSDL.SET window

Switch to the **BSDL.state** window and check **SetAndRun** and **TwoStepDR**, as shown in Figure 20.

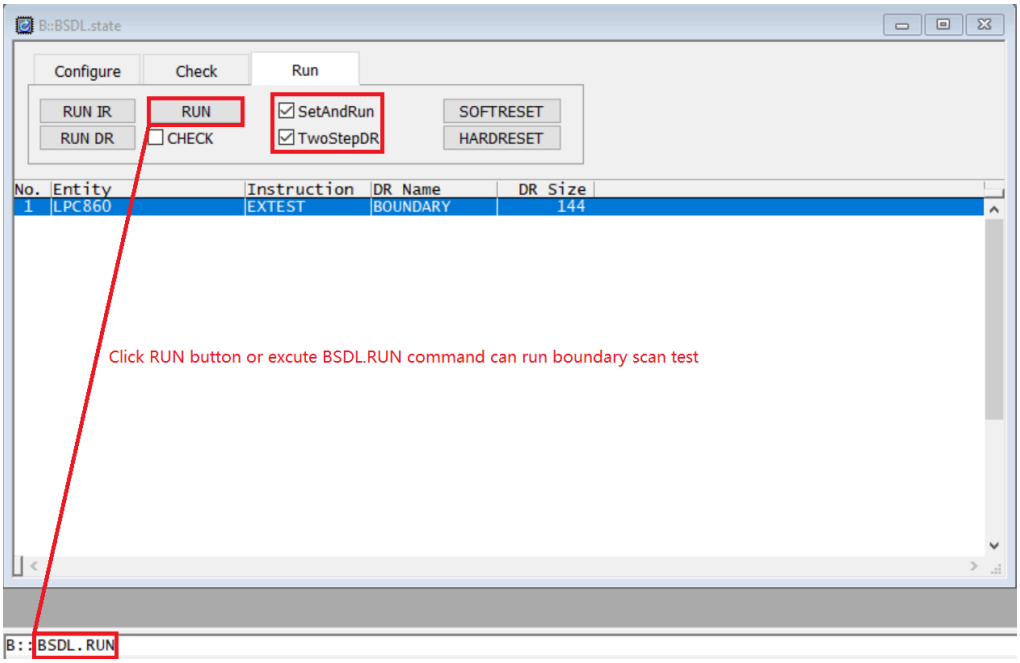

Figure 20. EXTEST settings in BSDL.state window

Switch back to the **BSDL.SET** window. Taking pin `P0_7` as an example to describe how to perform **EXTTEST**. Enable output of pin `P0_7` by clicking the button in the **Enable** column. Toggle the output logic state 0 or 1 of this pin by clicking the button in the **Reg.** column, as shown in Figure 21. Use a multimeter to measure whether the logic state really toggles on this pin or not.

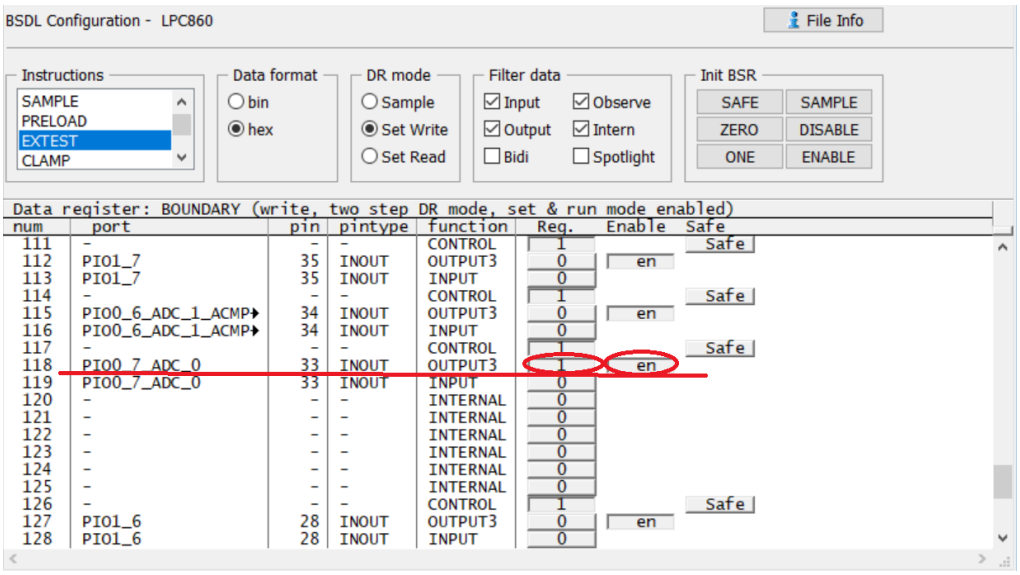

Figure 21. Enabling and toggling signal level

Use the above method to traverse all the I/O pins defined in the BSDL file. If all the I/O pins pass the test, the EXTTEST test of BSDL passes.

10. Use **PRELOAD** test with **EXTTEST**. To enable and preset driving buffers to 1 or 0 of all I/O pins, click **ONE** or **ZERO** and then **ENABLE** in the **Init BSR** field, as shown in Figure 22.

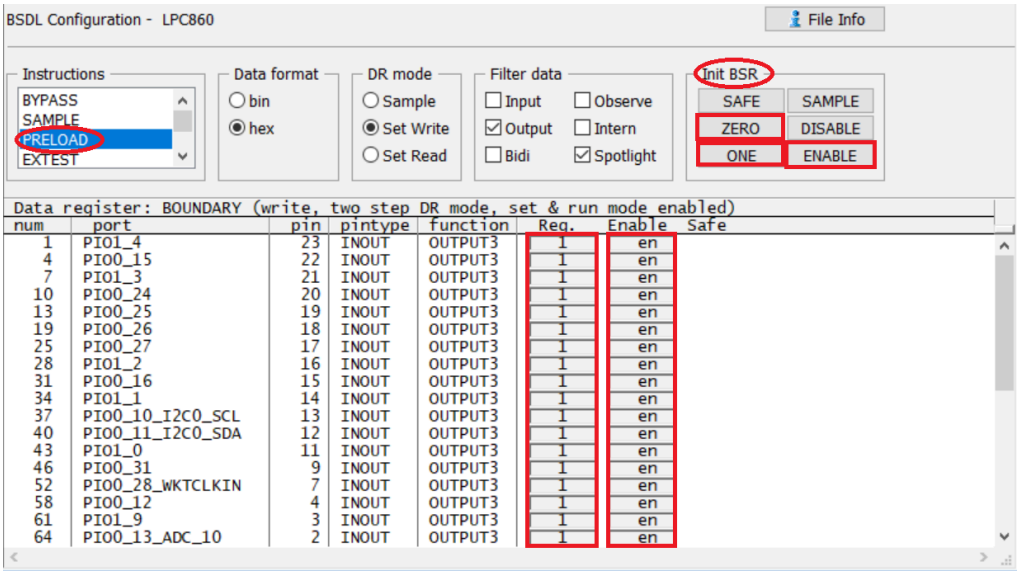

Figure 22. Preload I/O output status

Run the **PRELOAD** test and then **EXTTEST**. Use a multimeter to measure whether the logic state of all I/O pins matches the pre-load values.

Use the above method to traverse all the I/O pins defined in the BSDL file. If all the I/O pins pass the test, the PRELOAD test of BSDL passes.

11. Choose **HIGHZ** in the **Instructions** field in the **BSDL.SET** window and then run the **HIGHZ** test. All I/O pins defined in the BSDL file are in the high-impedance state. Taking 3.3 V logic as an example, if an intermediate level, such as 1.65 V, is applied to a pin in a high-impedance state, use a multimeter to measure the voltage on the pin. The measured voltage is 1.65 V and the pin in the high-impedance state is regarded as open circuits due to its high impedance. It does not cause a significant voltage drop to the external driving source.

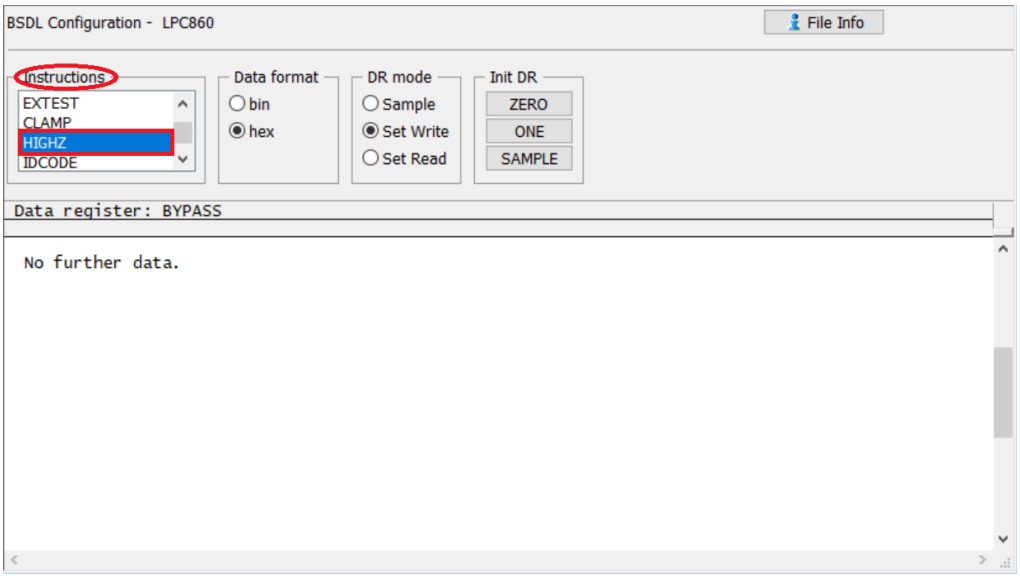

Figure 23. HIGHZ settings

After the HIGHZ test starts, all pins are in a high-impedance state. Then select a pin and apply an intermediate level. Taking 1.64 V as is selected, use a multimeter to measure the pin level. The HIGHZ test result of P0_7 pin is as shown in Figure 24.

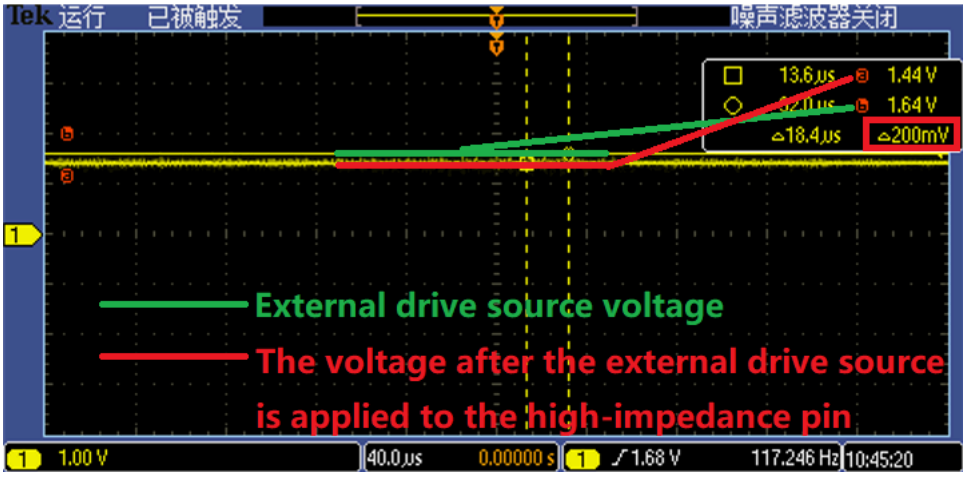

Figure 24. HIGHZ test result

As shown in Figure 24, when the external driving source is applied to the pin `P0_7`, it only causes a 200 mV voltage drop. `P0_7` is indeed in a high impedance state.

Use the above method to traverse all the I/O pins defined in the BSDL file. If all the I/O pins pass the test, the HIGHZ test of BSDL passes.

5 Automated boundary scan test

Through the above introduction to interactive boundary scan test, some interactive testing steps are not conducive for quick test. To improve test efficiency, TRACE32 supports practice script. To perform automated boundary scan test, write the script program.

On the main page of TRACE32, the **File** menu provides three submenu items related to script, **New Script**, **Open Script...**, and **Run Script...**. They are used to create, open, and run script.

A script example used to automate boundary scan test is described as below.

```
/* Copyright 2023 NXP. NXP Confidential. This software is owned or controlled by
 NXP and may only be
* used strictly in accordance with the applicable license terms found at
* https://www.nxp.com/LA_OPT_NXP_SW. The "production use license" in Section 2.3
 in the NXP SOFTWARE
* LICENSE AGREEMENT is expressly granted for this software.
*/

;System setup
SYStem.Mode Down                    ;Disables the debug mode.
SYStem.CPU CortexM0+                ;Tells TRACE32 the exact CPU type
                                    ;used on your target, CPU core of
                                    ;LPC86x is Cortex-M0+.
SYStem.CONFIG.DEBUGPORTTYPE JTAG    ;Specifies which probe cable shall
                                    ;be used, here, JTAG is selected
SYStem.JtagClock 1MHz               ;Selects JTAG frequency (TCK)
;BSDL Settings
BSDL.RESet                          ;Initialize the boundary scan engine
BSDL.ParkState Select-DR-Scan       ;Set PartState as Select-DR-Scan
BSDL.state                          ;Open BSDL.state window
;Configure boundary scan chain
BSDL.FILE LPC865M301JHI48.bsdl      ;your BSDL file name, need locate at the
 same
                                    ;folder with script file
;Check boundary scan chain
BSDL.SOFTRESET
IF !BSDL.CHECK.BYPASS()             ;BYPASS Test
(
  BSDL.BYPASSall
   PRINT %ERROR "Bypass test failed"
   ENDDO
)
IF !BSDL.CHECK.IDCODE()             ;IDCODE Test
(
   BSDL.IDCODEall
   PRINT %ERROR "ID code test failed"
   ENDDO
)
;Perform SAMPLE test
BSDL.SAMPLEall
;Perform EXTTEST
;Pin output settings, you can add other pin output settings
BSDL.SET 1. PORT PIO0_7 0           ;Set PIO0_7 output as 0
BSDL.RUN DR                         ;Only apply data register settings
                                    ;to the boundary scan chain
BSDL.SET 1. IR EXTEST               ;Only apply instruction register
                                    ;settings to the boundary scan chain
BSDL.RUN                            ;BSDL run
;Perform HIGHZ test
BSDL.SET 1. IR HIGHZ                ;Only apply instruction register
                                    ;settings to the boundary scan chain
  BSDL.RUN                          ;BSDL run
```

6 Revision history

Table 4 summarizes the revisions to this document.

Table 4. Revision history

Revision number	Date	Substantive changes
0	08 May 2023	Initial release

7 Legal information

7.1 Definitions

Draft — A draft status on a document indicates that the content is still under internal review and subject to formal approval, which may result in modifications or additions. NXP Semiconductors does not give any representations or warranties as to the accuracy or completeness of information included in a draft version of a document and shall have no liability for the consequences of use of such information.

7.2 Disclaimers

Limited warranty and liability — Information in this document is believed to be accurate and reliable. However, NXP Semiconductors does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information and shall have no liability for the consequences of use of such information. NXP Semiconductors takes no responsibility for the content in this document if provided by an information source outside of NXP Semiconductors.

In no event shall NXP Semiconductors be liable for any indirect, incidental, punitive, special or consequential damages (including - without limitation - lost profits, lost savings, business interruption, costs related to the removal or replacement of any products or rework charges) whether or not such damages are based on tort (including negligence), warranty, breach of contract or any other legal theory.

Notwithstanding any damages that customer might incur for any reason whatsoever, NXP Semiconductors' aggregate and cumulative liability towards customer for the products described herein shall be limited in accordance with the Terms and conditions of commercial sale of NXP Semiconductors.

Right to make changes — NXP Semiconductors reserves the right to make changes to information published in this document, including without limitation specifications and product descriptions, at any time and without notice. This document supersedes and replaces all information supplied prior to the publication hereof.

Suitability for use — NXP Semiconductors products are not designed, authorized or warranted to be suitable for use in life support, life-critical or safety-critical systems or equipment, nor in applications where failure or malfunction of an NXP Semiconductors product can reasonably be expected to result in personal injury, death or severe property or environmental damage. NXP Semiconductors and its suppliers accept no liability for inclusion and/or use of NXP Semiconductors products in such equipment or applications and therefore such inclusion and/or use is at the customer's own risk.

Applications — Applications that are described herein for any of these products are for illustrative purposes only. NXP Semiconductors makes no representation or warranty that such applications will be suitable for the specified use without further testing or modification.

Customers are responsible for the design and operation of their applications and products using NXP Semiconductors products, and NXP Semiconductors accepts no liability for any assistance with applications or customer product design. It is customer's sole responsibility to determine whether the NXP Semiconductors product is suitable and fit for the customer's applications and products planned, as well as for the planned application and use of customer's third party customer(s). Customers should provide appropriate design and operating safeguards to minimize the risks associated with their applications and products.

NXP Semiconductors does not accept any liability related to any default, damage, costs or problem which is based on any weakness or default in the customer's applications or products, or the application or use by customer's third party customer(s). Customer is responsible for doing all necessary testing for the customer's applications and products using NXP Semiconductors products in order to avoid a default of the applications and the products or of the application or use by customer's third party customer(s). NXP does not accept any liability in this respect.

Terms and conditions of commercial sale — NXP Semiconductors products are sold subject to the general terms and conditions of commercial sale, as published at http://www.nxp.com/profile/terms, unless otherwise agreed in a valid written individual agreement. In case an individual agreement is concluded only the terms and conditions of the respective agreement shall apply. NXP Semiconductors hereby expressly objects to applying the customer's general terms and conditions with regard to the purchase of NXP Semiconductors products by customer.

Export control — This document as well as the item(s) described herein may be subject to export control regulations. Export might require a prior authorization from competent authorities.

Suitability for use in non-automotive qualified products — Unless this data sheet expressly states that this specific NXP Semiconductors product is automotive qualified, the product is not suitable for automotive use. It is neither qualified nor tested in accordance with automotive testing or application requirements. NXP Semiconductors accepts no liability for inclusion and/or use of non-automotive qualified products in automotive equipment or applications.

In the event that customer uses the product for design-in and use in automotive applications to automotive specifications and standards, customer (a) shall use the product without NXP Semiconductors' warranty of the product for such automotive applications, use and specifications, and (b) whenever customer uses the product for automotive applications beyond NXP Semiconductors' specifications such use shall be solely at customer's own risk, and (c) customer fully indemnifies NXP Semiconductors for any liability, damages or failed product claims resulting from customer design and use of the product for automotive applications beyond NXP Semiconductors' standard warranty and NXP Semiconductors' product specifications.

Translations — A non-English (translated) version of a document, including the legal information in that document, is for reference only. The English version shall prevail in case of any discrepancy between the translated and English versions.

Security — Customer understands that all NXP products may be subject to unidentified vulnerabilities or may support established security standards or specifications with known limitations. Customer is responsible for the design and operation of its applications and products throughout their lifecycles to reduce the effect of these vulnerabilities on customer's applications and products. Customer's responsibility also extends to other open and/or proprietary technologies supported by NXP products for use in customer's applications. NXP accepts no liability for any vulnerability. Customer should regularly check security updates from NXP and follow up appropriately.

Customer shall select products with security features that best meet rules, regulations, and standards of the intended application and make the ultimate design decisions regarding its products and is solely responsible for compliance with all legal, regulatory, and security related requirements concerning its products, regardless of any information or support that may be provided by NXP.

NXP has a Product Security Incident Response Team (PSIRT) (reachable at PSIRT@nxp.com) that manages the investigation, reporting, and solution release to security vulnerabilities of NXP products.

NXP B.V. - NXP B.V. is not an operating company and it does not distribute or sell products.

7.3 Trademarks

Notice: All referenced brands, product names, service names, and trademarks are the property of their respective owners.

NXP — wordmark and logo are trademarks of NXP B.V.

Contents

Date of release: 8 May 2023
Document identifier: AN13806