User Manual

DA1458x Production Line Tool Reference CLI

UM-B-008

Abstract

This document describes the reference command line interface (CLI) for the production test tool of DA14580/581/583.

Contents

Tables

1 Terms and definitions

CLI	Command Line Interface
GUI	Graphical User Interface
HCI	Host Controller Interface
OTP	One Time Programmable
XTAL16M	16 MHz Crystal
XTAL32K	32 KHz Crystal
RC16M	16 MHz RC Oscillator
RC32K	32 KHz RC Oscillator

2 References

1. DA14580, Data sheet, Dialog Semiconductor
2. SPECIFICATION OF THE BLUETOOTH SYSTEM, version 4.0, Bluetooth SIG, 2010
3. Connection Manager, Help Document, Dialog Semiconductor
4. DA14581, Data sheet, Dialog Semiconductor

3 Introduction

This document describes the command line interface (CLI) for the production test tool of DA14580/581/583.

The tool is a Microsoft Windows command line program that enables communication over UART with a DA14580/581/583 device running the production test firmware.

The production test firmware is a special firmware that supports:

- The Bluetooth SIG standardised receiver and transmitter test HCI commands [2].
- Additional custom test HCI commands.

All test commands are also supported by the Connection Manager GUI application [3].

4 Getting started

4.1 Precompiled binaries

Precompiled binaries are provided for both the production test firmware and tool in the SDK:

- **binaries\da1458x\prod_test\prod_test.hex for DA14580**
  binaries\da1458x\prod_test\prod_test_581.hex for DA14581
  binaries\da1458x\prod_test\prod_test_583.hex for DA14583
- **binaries\host\windows\prod_test_cmds\prodtest.exe**

4.2 Building the tool

For SDK3.0.10.1 and earlier the tools are placed under: **tools\prod_test\prod_test_cmds** folder in the SDK.

For SDK5.0.2.1 and later this is under: **utilities\prod_test\prod_test_cmds**

The next lines refer the SDK5.0.2 path using the **utilities** folder but it is the same in SDK3.0.10 in the **tools** folder

Open **utilities\prod_test\prod_test_cmds\prodtest.sln** in “Microsoft Visual C++ 2010 Express”.

Make sure that the active solution configuration is the Release Configuration.

Build the project by selecting “Build –> Build Solution” in the menu.

The executable **prodtest.exe** is generated under the **utilities\prod_test\prod_test_cmds\Release** folder.

4.3 Building the firmware

The required firmware is included in the SDK under the folder:

SDK5.0.2 and later: **projects\target_apps\prod_test\prod_test**

SDK3.0.10.1 and earlier: **dk_apps\keil_projects\prod_test\prod_test**

Steps for building the firmware

1. Open **the \prod_test\prod_test\ folder.**
2. Open the appropriate project depending on chip type and Keil µVision IDE version:
   a. Keil 4:
      DA14580: .\Keil_4\prod_test.uvproj
      DA14581: .\Keil_4\prod_test_581.uvproj
      DA14583: .\Keil_4\prod_test_583.uvproj
   b. Keil 5:

DA14580: .\Keil_5\prod_test.uvprojx
DA14581: .\Keil_5\prod_test_581.uvprojx
DA14583: .\Keil_5\prod_test_583.uvprojx

3. Select menu “Project -> Rebuild all target files” to build the project.
4. Get the generated hex file from:

a. Keil 4:

DA14580: **.\Keil_4\out\prod_test.hex**
DA14581: **.\Keil_4\out_581\prod_test_581.hex**
DA14582: **.\Keil_4\out_583\prod_test_583.hex**

b. Keil 5:

DA14580: **.\Keil_5\out\prod_test.hex**
DA14581: **.\Keil_5\out_581\prod_test_581.hex**
DA14583: **.\Keil_5\out_583\prod_test_583.hex**

5 General description

The general syntax of the tool is:

prodtest <switches> <command> <command-arg-1> … <command-arg-2>

A command always returns a status code and optionally a list of return values. The return status code and values are written to the standard output in a simple format: <*value_name*> = <*value*>.

The return status code is also returned as an exit code.

A zero status code represents the successful execution of a command. A non-zero status code encodes the type of failure.

All other output of the tool is written to stderr (except the help message when –h option is used).

6 Command line switches

6.1 Switch –h

Description: print out all the commands of prodtest on a command line terminal

Usage: prodtest -h

6.2 Switch –p

Description: select the COM port number on a PC. All commands require this switch.

Usage: prodtest –p <COM_PORT_NUMBER> ….

6.3 Switch –v

Description: display the tool’s version

Usage: prodtest –v

7 Commands

7.1 cont_pkt_tx

Description: this is the Bluetooth SIG standardized [2] HCI_LE_Transmitter_Test command. It continuously transmits packets until the **stoptest** command is executed.

Syntax: prodtest –p <COM_PORT_NUMBER> cont_pkt_tx <FREQUENCY> <DATA_LENGTH> <PAYLOAD_TYPE>

Parameters:

- *FREQUENCY* is an even integer between 2402 and 2480. E.g. integer 2480 corresponds to 2.480 GHz.
- *DATA_LENGTH* is the payload length in bytes (a number between 1 and 37).
- *PAYLOAD_TYPE* must have one of the following values:
  0: Pseudo-Random bit sequence 9
  1: Pattern of alternating bits ‘11110000’
  2: Pattern of alternating bits ‘10101010’
  3: Pseudo-Random bit sequence 15
  4: Pattern of All ‘1’ bits
  5: Pattern of All ‘0’ bits
  6: Pattern of alternating bits ‘00001111’
  7: Pattern of alternating bits ‘0101’

Example:

```
prodtest -p 14 cont_pkt_tx  2402  35  6
```

Output example:

```
status = 0
```

7.2 pkt_tx

Description: transmit the specified number of packets.

Syntax: prodtest –p <COM_PORT_NUMBER> pkt_tx <FREQUENCY> <DATA_LENGTH> <PAYLOAD_TYPE> <NUMBER_OF_PACKETS>

Parameters:

- *FREQUENCY* is an even integer between 2402 and 2480. E.g. 2480 corresponds 2.480 GHz.
- *DATA_LENGTH* is the payload length in bytes (a number between 1 and 37).
- *PAYLOAD_TYPE* must have one of the following values:
  0 : Pseudo-Random bit sequence 9
  1 : Pattern of alternating bits ‘11110000
  2 : Pattern of alternating bits ‘10101010’
  3 : Pseudo-Random bit sequence 15
  4 : Pattern of All ‘1’ bits
  5 : Pattern of All ‘0’ bits
  6 : Pattern of alternating bits ‘00001111
  7 : Pattern of alternating bits ‘0101’
- *NUMBER_OF_PACKETS* is an integer between 1 and 65535.

Example: the following command sends 1000 packets

```
prodtest -p 14 pkt_tx  2402  35  6 1000
```

Output example:

status = 0

7.3 start_pkt_rx

Description: This is the Bluetooth SIG standardized [2] HCI_LE_Receiver_Test command. It continuously receives packets until the **stoptest** command is executed.

Syntax: prodtest –p <COM_PORT_NUMBER> start_pkt_rx <FREQUENCY>

Parameters:

- *FREQUENCY* is an even integer between 2402 and 2480. E.g. 2480 corresponds 2.480 GHz.

Example:

prodtest –p 14 start_pkt_rx 2402

Output example:

status = 0

7.4 start_pkt_rx_stats

Description: starts packet RX with additional statistics. It continuously receives packets until the stop_pkt_rx_stats command is executed.

Syntax: prodtest –p <COM_PORT_NUMBER> start_pkt_rx_stats <FREQUENCY>

Parameters:

- *FREQUENCY* is an even integer between 2402 and 2480. E.g. 2480 corresponds 2.480 GHz.

Example:

prodtest –p 14 start_pkt_rx_stats 2402

Output example:

status = 0

7.5 stop_pkt_rx_stats

Description: ends packet RX with additional statistics and reports the following statistics:

- number of packets received correctly (nb_packets_received_correctly)
- number of packets with sync error (nb_packets_with_syncerror)
- number of packets with CRC error (nb_packets_received_with_crcerr)
- RSSI in dBm (rssi)

Syntax: prodtest –p <COM_PORT_NUMBER> stop_pkt_rx_stats

Example:

prodtest –p 14 stop_pkt_rx_stats

Output example:

status = 0

nb_packets_received_correctly = 8529

nb_packets_with_syncerror = 0

nb_packets_received_with_crcerr = 1

rssi = -37.30

7.6 stoptest

Description: This is the Bluetooth SIG standardized [2] HCI_LE_Test_End command. It is used:

- after a cont_pkt_tx command to end the standard packet TX test mode.
- after a start_pkt_rx command to end the standard packet RX mode and report the number of received packets.

Syntax: prodtest –p <COM_PORT_NUMBER> stoptest

Example:

prodtest –p 14 stoptest

Output example (the number of packets is always zero when executed after a cont_pkt_tx):

status = 0

number_of_packets = 0

Output example (when executed after a start_pkt_rx):

status = 0

number_of_packets = 4360

7.7 unmodulated TX

Description: starts a Continuous Wave (CW) or unmodulated TX test.

Syntax: prodtest –p <COM_PORT_NUMBER> unmodulated TX <FREQUENCY>

Parameters:

- *FREQUENCY* is an even integer between 2402 and 2480. E.g. 2480 corresponds 2.480 GHz.

Example:

prodtest -p 14 unmodulated TX 2404

Output example:

status = 0

7.8 unmodulated RX

Description: starts the unmodulated RX test.

Syntax: prodtest –p <COM_PORT_NUMBER> unmodulated RX <FREQUENCY>

Parameters:

- *FREQUENCY* is an even integer between 2402 and 2480. E.g. 2480 corresponds 2.480 GHz.

Example:

prodtest -p 14 unmodulated RX 2404

Output example:

status = 0

7.9 unmodulated OFF

Description: stops unmodulated TX or RX test.

Syntax: prodtest –p <COM_PORT_NUMBER> unmodulated OFF

Example:

prodtest -p 14 unmodulated OFF

Output example:

status = 0

7.10 start_cont_tx

Description: starts the continuous TX test mode. It transmits continuously a modulated signal until the **stop_cont_tx** command is executed.

Syntax: prodtest –p <COM_PORT_NUMBER> start_cont_tx <FREQUENCY> <PAYLOAD_TYPE>

Parameters:

- *FREQUENCY* is an even integer between 2402 and 2480. E.g. 2480 corresponds 2.480 GHz.
- *PAYLOAD_TYPE* must have one of the following values:
  0 : Pseudo-Random bit sequence 9
  1 : Pattern of alternating bits ‘11110000’
  2 : Pattern of alternating bits ‘10101010’
  3 : Pseudo-Random bit sequence 15
  4 : Pattern of All ‘1’ bits
  5 : Pattern of All ‘0’ bits
  6 : Pattern of alternating bits ‘00001111’
  7 : Pattern of alternating bits ‘0101’

Example:

prodtest –p 14 start_cont_tx 2404 4

Output example:

status = 0

7.11 stop_cont_tx

Description: stops the continuous TX test mode.

Syntax: prodtest –p <COM_PORT_NUMBER> stop_cont_tx

Example:

prodtest –p 14 stop_cont_tx

Output example:

status = 0

7.12 reset

Description: this is the Bluetooth SIG standardized [2] HCI_Reset command.

Syntax: prodtest –p <COM_PORT_NUMBER> reset

Example:

prodtest –p 14 reset

Output example:

status = 0

7.13 xtrim rd

Description: reads the value of the XTAL16M trimming register as specified in DA14580 datasheet [1] section 5: CLK_FREQ_TRIM_REG (0x50000002).

The value is reported in decimal format.

Syntax: prodtest -p <COM port number> xtrim rd

Example:

prodtest –p 14 xtrim rd

Output example:

status = 0

trim_value = 500

7.14 xtrim wr

Description: writes a value into the XTAL16M trimming register. The new value is written immediately and it is already effective when the prodtest program finishes executing this command.

Syntax: prodtest -p <COM port number> xtrim wr <trim_value>

Parameters:

- *trim_value* is the unsigned 16-bit decimal value to be written into the XTAL16M trimming register.

Example:

prodtest –p 14 xtrim wr 500

Output example:

status = 0

7.15 xtrim en

Description: enables XTAL16M output on GPIO P0_5. The command takes effect immediately.
Syntax: prodtest -p <COM port number> xtrim en

Example:

prodtest –p 14 xtrim en

Output example:

status = 0

Note 1 This command can also be used to enable the 32kHz clock on a GPIO 0_6. To do this two additional registers need to be set using the **write_reg16** command (see section 7.31)

Enable 32kHz clock in CLK_32K_REG, set bit 0 and bit 7 to ‘1’

Set GPIO P0_6 to output mode. In P06_MODE_REG, set bit 8 and 9 to ‘1’

Example:

prodtest –p 14 write_reg16 50000020 079D

prodtest –p 14 write_reg16 50003012 0300

Output example:

status = 0

7.16 xtrim dis

Description: disables XTAL16M output on GPIO P0_5. The command takes effect immediately.

Note 2 This command also disables:

XTAL32K output on GPIO P0_6

RC16M output on P0_7

RC32K output on P1_0

Syntax: prodtest -p <COM port number> xtrim dis

Example:

prodtest –p 14 xtrim dis

Output example:

status = 0

7.17 xtrim inc

Description: increments the XTAL16M trimming register. The register's new value is already effective when the prodtest program finishes executing this command.

Syntax: prodtest -p <COM port number> xtrim inc <delta>

Parameters:

- *delta* is the unsigned 16-bit decimal value to be added to the XTAL16M trimming register.

Example:

prodtest –p 14 xtrim inc 5

Output example:

status = 0

7.18 xtrim dec

Description: decrements the XTAL16M trimming register. The register's new value is already effective when the prodtest program finishes executing this command.

Syntax: prodtest -p <COM port number> xtrim dec <delta>

Parameters:

- *delta* is the unsigned 16-bit decimal value to be subtracted from the XTAL16M trimming register.

Example:

prodtest –p 14 xtrim dec 5

Output example:

status = 0

7.19 xtrim cal

Description: runs the XTAL16M trimming register calibration procedure which uses an externally applied 500ms high square pulse as a reference for trim value calibration. The procedure executes the following steps:

1. Calibrates the XTAL16M trimming register.
2. Programs the calibrated value in OTP (at address 0x7F8C).
3. Enables the "XTAL16M trim value is valid" bit in OTP (at address 0x7F78).

If a square pulse has not been applied on the given GPIO then the calibration procedure will fail with a status code equal to 27.

Syntax: prodtest -p <COM port number> xtrim cal <Px_y>

Parameters:

- *Px_y* is the GPIO that the external square pulse is applied on.

Example:

prodtest –p 14 xtrim cal P1_0

Output example:

status = 0

7.20 xtrim caltest

Description: behaves identically to “xtrim cal” with the exception that it does not modify the OTP.

Syntax: prodtest -p <COM port number> xtrim caltest <Px_y>

Parameters:

- *Px_y* is the GPIO that the external square pulse is applied on.

Example:

prodtest –p 14 xtrim caltest P1_0

Output example:

status = 0

7.21 sleep

Description: puts the device to sleep for a specified number of minutes and seconds.

Syntax: prodtest –p <COM_PORT_NUMBER> sleep <mode> <minutes> <seconds>

Parameters:

- *mode* is one of the available sleep modes defined in section 3.5 of the datasheet [1]:
  - **none** = Sleep Mode (no power gating, ARM CPU is idle waiting for an interrupt)
  - **extended** = Extended Sleep Mode
  - **deep** = Deep Sleep Mode
- *minutes* is a number between 0 and 255.
- *seconds* is a number between 0 and 255.

If both minutes and seconds are set to zero then the device sleeps forever.
The active peripherals in each sleep mode are shown in table 1.

Table 1: Power domain states for each sleep mode

Power mode	PD_SYS (AHB, OTP, ROM, Watchdog, SW timer, GPIO multiplexing)	PD_PER (UARTs, SPI, I2C, Keyboard controller, ADC)	PD_DBG	PD_RAD (Radio and BLE core)	PD_RRx (Retention RAM x)	PD_SR (System RAM)	Analog (BandGap, DCDC converter XTAL16M, RC oscillators, ADC, LDOs)
Sleep Mode	ON	ON	ON	ON	ON	ON	ON
Extended Sleep Mode	OFF	Programmed to OFF	OFF	Programmed to OFF	Programmed to ON	ON	OFF
Deep Sleep Mode	OFF	Programmed to OFF	OFF	Programmed to OFF	Programmed to ON	OFF	OFF

Known limitations:

- UART communication is lost when the device wakes up from extended sleep mode.
- The device will not wake up from deep sleep mode.

Example 1:

prodtest –p 14 sleep none 1 0

Example 2:

prodtest –p 14 sleep extended 0 30

Example 3:

prodtest –p 14 sleep deep 0 0

Output example:

status = 0

7.22 otp wr_xtrim

Description: writes the XTAL16M trim value into the OTP header.

Syntax: prodtest -p <COM port number> otp wr_xtrim <decimal trim value>

Parameters:

- *decimal trim value* is the unsigned 16-bit decimal XTAL16M trim value to be written into the OTP header.

Example:

prodtest –p 14 otp wr_xtrim 500

Output example:

status = 0

7.23 otp rd_xtrim

Description: reads the XTAL16M trim value from the OTP header. The output value is in decimal format.

Syntax: prodtest -p <COM port number> otp rd_xtrim

Example:

prodtest –p 14 otp rd_xtrim

Output example:

status = 0

otp_xtrim_value = 500

7.24 otp wr_bdaddr

Description: writes the Bluetooth Device address into the OTP header.

Syntax: prodtest -p <COM port number> otp wr_bdaddr <BD address>

Parameters:

- *BD address* is the Bluetooth Device address to be written into the OTP header. The BD address must be formatted as XX:XX:XX:XX:XX:XX where X is a hexadecimal digit.

Example:

prodtest –p 14 otp wr_bdaddr 44:49:41:4c:4f:47

Output example:

status = 0

7.25 otp re_xtrim

Description: reads the XTAL16M calibration flag from the OTP header (bit 4 of the calibration flag in OTP header).

Syntax: prodtest -p <COM port number> otp re_xtrim

Example:

prodtest –p 14 otp re_xtrim

Output example:

status = 0

otp_xtrim_enabled = 1

7.26 otp we_xtrim

Description: enables the XTAL16M calibration flag from the OTP header by modifying the bit 4 of the calibration flag in OTP header from 0 to 1.

Syntax: prodtest –p <COM port number> otp we_xtrim

Example:

prodtest –p 14 otp we_xtrim

Output example:

status = 0

7.27 otp_read

Description: reads consecutive words from OTP memory.

Syntax: prodtest -p <COM port number> otp_read <otp address> <word count>

Parameters:

- *otp address* is the start address of the OTP memory region to be read. This is a word address hence it must be divisible by 4. It must be lower than 0x8000. It must be formatted as a 16 bit hexadecimal number.
- *Word count* is the number of words to be read. The accepted range is from 1 to 60. The word count must satisfy the formula: *<otp address> + 4 * <word count> <= 0x8000.* It must be formatted as a decimal number.

Example:

prodtest -p 16 otp_read 7FDC 5

Output example:

status = 0

[7FDC] = 1234A5A5

[7FE0] = A5A51234

[7FE4] = 14121117

[7FE8] = 24222127

[7FEC] = 00000007

7.28 otp_write

Description: writes consecutive words into OTP memory.

Syntax: prodtest -p <COM port number> otp_write <otp address> <word #1> … <word #n>

Parameters:

- *otp address* is the start address of the OTP memory region to be written. This is a word address hence it must be divisible by 4. It must be lower than 0x8000. It must be formatted as a 16 bit hexadecimal number.
- *Word #i* is the word value to be written in address: *<otp address> + 4 * i*. It must be formatted as a 32 bit hexadecimal number. The number of word value arguments must be at most 60 and must satisfy the formula: *<otp address> + 4 * n <= 0x8000.*

Example:

prodtest -p 16 otp_write 7FDC 1234A5A5 A5A51234

Output example:

status = 0

7.29 read_reg16

Description: reads the value of a 16-bit register. The output value is printed in hex.

Syntax: prodtest -p <COM port number> read_reg16 <register address>

Parameters:

- *register address* is the address of the 16-bit register to be read. The register address must be half-word aligned hence it must be divisible by 2. It must be formatted as a 32-bit hexadecimal number.

Example: the following command reads TRIM_CTRL_REG (0x50000016) [1].

prodtest -p 16 read_reg16 50000016

Output example:

status = 0

value = 00A2

7.30 read_reg32

Description: reads the value of a 32-bit register. The output value is printed in hex.

Syntax: prodtest -p <COM port number> read_reg32 <register address>

Parameters:

- *register address* is the address of the 32-bit register to be read. The register address must be word aligned hence it must be divisible by 4. It must be formatted as a 32-bit hexadecimal number.

Example: the following command reads PATCH_VALID_REG (0x40008400) [1].

prodtest -p 16 read_reg32 40008400

Output example:

status = 0

value = 00000001

7.31 write_reg16

Description: writes into a 16-bit register.

Syntax: prodtest -p <COM port number> write_reg16 <register address> <register value>

Parameters:

- *register address* is the address of the 16-bit register to be read. The register address must be half-word aligned hence it must be divisible by 2. It must be formatted as a 32-bit hexadecimal number.

- *register value* is the value to be written into the register. It must be formatted as a 16-bit hexadecimal number.

Example: the following command writes 0x00B3 in TRIM_CTRL_REG (0x50000016) [1].

```
prodtest -p 16 write_reg16 50000016 00B3
```

Output example:

```
status = 0
```

7.32 write_reg32

Description: writes into a 32-bit register.

Syntax: prodtest -p <COM port number> write_reg32 <register address> <register value>

Parameters:

- *register address* is the address of the 32-bit register to be read. The register address must be word aligned hence it must be divisible by 4. It must be formatted as a 32-bit hexadecimal number.
- *register value* is the value to be written into the register. It must be formatted as a 32-bit hexadecimal number.

Example: the following command writes 0x0000000F in PATCH_VALID_REG (0x40008400) [1].

```
prodtest -p 16 write_reg32 40008400 0000000F
```

Output example:

```
status = 0
```

8 Return status codes

Following table summarizes the return codes of the productions test tool

Table 2: Return codes

Status code	Description
0	SC_NO_ERROR
1	SC_MISSING_COMMAND
2	SC_INVALID_COMMAND
3	SC_WRONG_NUMBER_OF_ARGUMENTS
4	SC_INVALID_COM_PORT_NUMBER
5	SC_INVALID_FREQUENCY_ARG
6	SC_INVALID_DATA_LENGTH_ARG
7	SC_INVALID_PAYLOAD_TYPE_ARG
8	SC_COM_PORT_INIT_ERROR
9	SC_RX_TIMEOUT
10	SC_UNEXPECTED_EVENT
11	SC_INVALID_NUMBER_OF_PACKETS_ARG
12	SC_INVALID_UNMODULATED_CMD_MODE_ARG
13	SC_COM_PORT_NOT_SPECIFIED
14	SC_INVALID_SLEEP_CMD_MODE_ARG
15	SC_INVALID_SLEEP_CMD_MINUTES_ARG
16	SC_INVALID_SLEEP_CMD_SECONDS_ARG
17	SC_INVALID_XTAL_TRIMMING_CMD_OPERATION_ARG
18	SC_INVALID_XTAL_TRIMMING_CMD_TRIM_VALUE_ARG
19	SC_INVALID_OTP_CMD_OPERATION_ARG
20	SC_INVALID_OTP_CMD_TRIM_VALUE_ARG
21	SC_INVALID_OTP_CMD_BDADDR_ARG
22	SC_INVALID_OTP_ADDRESS_ARG
23	SC_INVALID_NUMBER_OF_OTP_WORDS_ARG
24	SC_INVALID_WORD_VALUE_ARG
25	SC_INVALID_GPIO_ARG
26	SC_XTAL_TRIMMING_CAL_OUT_OF_RANGE_ERROR
27	SC_XTAL_TRIMMING_CAL_FREQ_NOT_CONNECTED
28	SC_INVALID_REGISTER_ADDRESS_ARG
29	SC_INVALID_REGISTER_VALUE_ARG
1000 – 1063	SC_HCI_STANDARD_ERROR_CODE_BASE (1000) + standard HCI error code

9 Revision history

Revision	Date	Description
1.0	24-Mar-2014	Initial version for DA14580-01
1.1	17-Apr-2014	Addition of commands for reading/writing XTAL16M trim value and BD address from OTP
1.2	22-Apr-2014	Addition of commands for enabling XTAL16M calibration flag and reading XTAL16M calibration flag
1.3	05-May-2014	Modification of “otp re_xtrim” command’s output.
1.4	16-May-2014	Addition of otp_read, otp_write_commands.
1.5	20-Jun-2014	Addition of read_reg16, read_reg32, write_reg16, write_reg32, “xtrim cal” and “xtrim caltest” commands.
1.6	06-Mar-2015	Updated for DA14581. Added –v switch. Updated paths for SDK v3.0.8. The “xtrim cal” and “xtrim caltest” commands detect if the square wave pulse is absent.
1.7	18-Nov-2015	Updated for the SDK 3.0.10 and SDK5.0.2 release. Added the description how to enable XTAL32 on a GPIO
1.8	17-Jan-2022	Updated logo, disclaimer, copyright.

Status definitions

Status	Definition
DRAFT	The content of this document is under review and subject to formal approval, which may result in modifications or additions.
APPROVED or unmarked	The content of this document has been approved for publication.

RoHS Compliance

Dialog Semiconductor's suppliers certify that its products are in compliance with the requirements of Directive 2011/65/EU of the European Parliament on the restriction of the use of certain hazardous substances in electrical and electronic equipment. RoHS certificates from our suppliers are available on request.