Application note

DA14580/581/583 Booting from serial interfaces

AN-B-001

Abstract

The DA1458x can boot from external serial devices when the OTP memory is not programmed, to enable development of the application code. At power-up the system enters Development Mode, where the boot code decides which interface to boot from. This document describes the booting sequence for all supported serial interfaces and provides the developer with the necessary information for realizing the protocol required for establishing communication between an external device and the DA1458x.

Contents

Figures

Tables

1 Terms and definitions

CPU	Central processing unit
CS	Chip Select
I2C	Inter-Integrated Circuit
JTAG	Joint Test Action Group
MISO	Master Input Slave Output
MOSI	Master Output Slave Input
OTP	One Time Programmable (memory)
RAM	Random Access Memory
ROM	Read-Only Memory
SPI	Serial Peripheral Interface
SCK	SPI Serial Clock
SDA	Serial Data Line
SCL	Serial Clock Line
SW	Software
UART	Universal Asynchronous Receiver/Transmitter
URX	UART Receive port
UTX	UART Transmit port

2 References

For each supported product list below DA14580 and A14581, please refer to the following documents available from https://support.dialog-semiconductor.com/connectivity

[1] DA14580, Datasheet Dialog Semiconductor

[2] DA14581, Datasheet Dialog Semiconductor

3 Introduction

The DA1458x operates in two modes, namely the 'Normal Mode' and the 'Development/Calibration Mode' hereafter addressed as 'DevMode'. The decision which mode the chip enters after power-up, is taken by the boot code residing in the ROM. A complete flow chart of the booting code is illustrated in the datasheet of the DA1458x. [1] [2]

DevMode will be entered when the OTP header contains a value zero at the first two addresses, when read by the CPU. This implies that the OTP is not programmed and the DA1458x should switch to the DevMode, so that users get access to download code from external devices into the internal SRAM (SysRAM). However, when the OTP contains specific values (magic numbers) in these locations the DA1458x will enter Normal Mode and proceed with mirroring of the OTP contents into the SysRAM in the booting sequence, as described in the datasheet. [1] [2]

To allow for maximum flexibility, a predefined number of pins are examined and utilised at boot time to communicate with external devices using the three serial interfaces available on chip: UART, SPI and I2C. SPI and I2C can be masters on the DA1458x side expecting to communicate with an external slave device and SPI can also be slave expecting to communicate with an external master.

4 Booting pins

During booting, port P0 is used to check for the presence of external devices. The assignment of the pins as well as the sequence of the interface activation by the boot code is presented in Table 1 below.

Table 1: Pin assignment and booting sequence from external devices

<table>
<tr><th>Pin</th><th>Step</th><th>Booting from external SPI Master</th><th>Step</th><th>Booting from external SPI Master</th><th>Step</th><th>UART</th><th>Step</th><th>Booting from external SPI Slave</th><th>Step</th><th>I2C</th></tr>
<tr><td>P0_0</td><td rowspan="8">1</td><td>SCK</td><td rowspan="8">2</td><td>SCK</td><td rowspan="2">3</td><td>TX</td><td rowspan="8">7</td><td>SCK</td><td rowspan="2">8</td><td>SCL</td></tr>
<tr><td>P0_1</td><td></td><td>CS</td><td>RX</td><td></td><td>SDA</td></tr>
<tr><td>P0_2</td><td></td><td>MISO</td><td rowspan="2">4</td><td>TX</td><td></td><td rowspan="2">9</td><td>SCL</td></tr>
<tr><td>P0_3</td><td>CS</td><td>MOSI</td><td>RX</td><td>CS</td><td>SDA</td></tr>
<tr><td>P0_4</td><td></td><td></td><td rowspan="2">5</td><td>TX</td><td></td><td rowspan="2">10</td><td>SCL</td></tr>
<tr><td>P0_5</td><td>MOSI</td><td></td><td>RX</td><td>MISO</td><td>SDA</td></tr>
<tr><td>P0_6</td><td>MISO</td><td></td><td rowspan="2">6</td><td>TX</td><td>MOSI</td><td rowspan="2">11</td><td>SCL</td></tr>
<tr><td>P0_7</td><td></td><td></td><td>RX</td><td></td><td>SDA</td></tr>
</table>

The mapping of the serial interface to each pin is shown in Table 1. Each interface column is preceded by a 'Step' column, which corresponds to the sequence step in the booting procedure.

The first step of the boot code is to configure the DA1458x's SPI controller to Slave mode (try to boot from an external SPI Master device) and assign the SPI's SCK to P0_0, CS to P0_3, MOSI to P0_5 and MISO to P0_6.

If either there is no SPI master connected to these pins or the SPI master does not communicate according to the protocol described in section 3.0, then the boot code continues with the step 2. In Step 2 the boot code will check for communication with an external SPI master just like in case of step 1 using the pins P0_0, P0_1, P0_2 and P0_3 this time.

The booting sequence is completed at step 11. If at step 11 also there is no response received at P0_6 (SCL) and P0_7 (SDA), then the booting sequence will retry starting from step 7 (SPI Master and UART will not be activated again).

Steps 7 to 11 are executed a total of 5 times. If no successful communication has been established by then, the DA1458x boot code will end in an endless (while) loop with the Serial Wire interface (JTAG) activated.

5 Booting sequence

The booting sequence of DA14581 has been optimised for fast startup by removing the 100 ms waiting time.

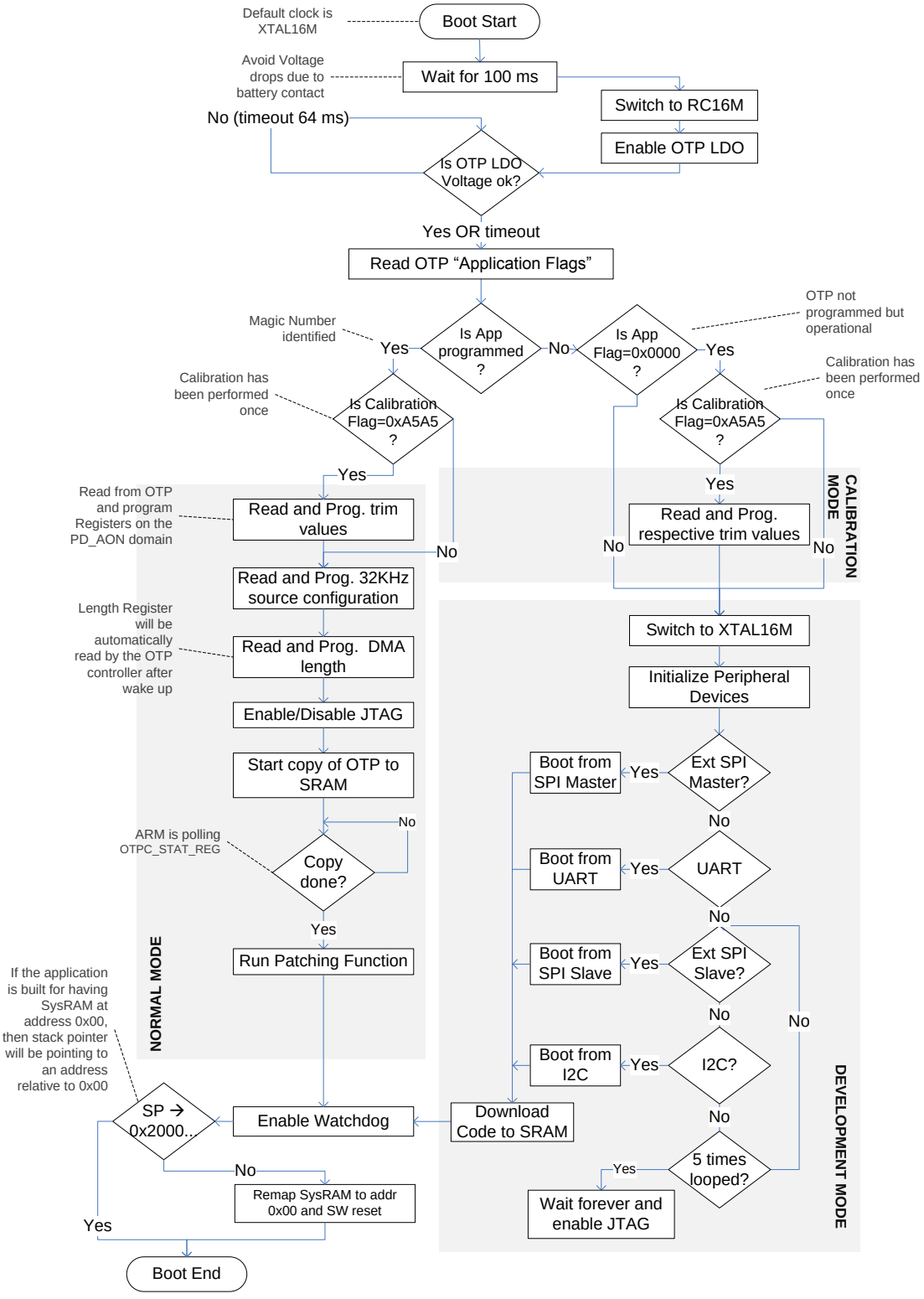

Figure 1: DA14580 booting sequence

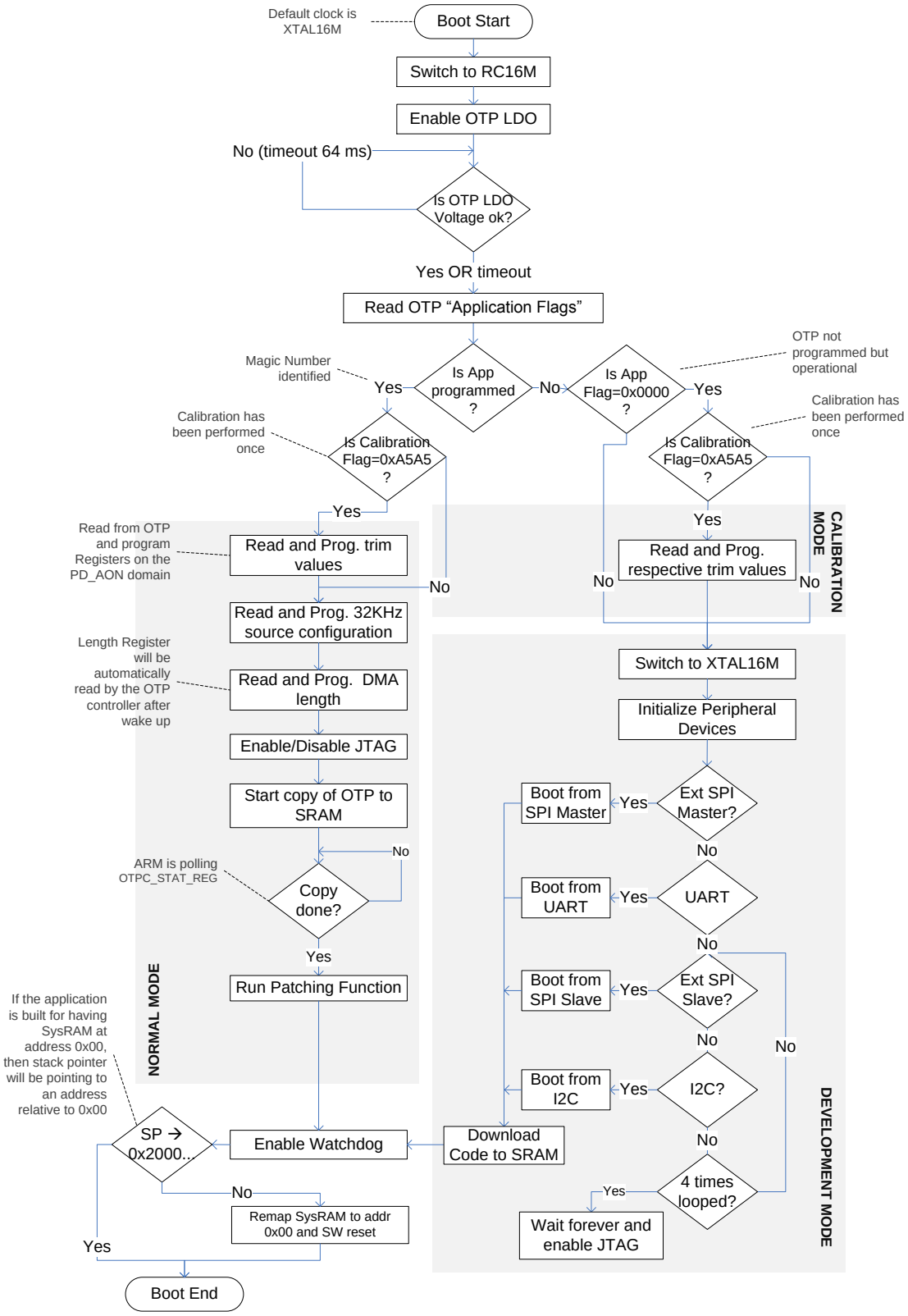

Figure 2: DA14581 booting sequence

6 Booting protocols

6.1 DA1458x connected to SPI Master

The boot code acts as the slave and initially configures the DA1458x SPI controller with the following parameters:

- 8 bit mode
- Slave role
- Mode 0: SPI clock is initially expected to be low and SPI phase is zero.

The communication protocol required for downloading the SW into the SysRAM is depicted in Table 2.

Note: The master SPI device generates the SPI clock for the DA1458x. In case of a continuous SPI clock, the frequency of this clock must not be higher than 500 kHz. For an SPI clock frequency higher than 500 kHz, the SPI clock should be paused after each byte.

Table 2: SPI Master boot protocol

Byte number	DA1458x MOSI	DA1458x MISO
0	Preamble: 0x70	-
1	Preamble: 0x50	-
2	Empty: 0x00	-
3	Length LS byte	Preamble ACK: 0x02 Preamble NACK:0x20
4	Length MS byte	-
5	CRC byte	-
6	Mode byte	Length ACK:0x02 Length NACK:0x20
7	Empty: 0x00	-
8	Data bytes	Code/Mode ACK:0x02 Code/Mode NACK:0x20

The communication starts with the external SPI master device sending the Preamble bytes (0x70 and 0x50) followed by a zero byte. The DA1458x confirms the reception of the Preamble with 0x02 (Acknowledged) or 0x20 (Not Acknowledged) in case something went wrong. Bytes 3 and 4 define the length of the payload to follow. The least significant byte is sent first. The length is a number which represents the amount of data in 32-bit words.

Next, the SPI master must send the calculated CRC of the payload. The CRC is calculated by XORing every successive byte with the previous value. Initial CRC value is 0xFF.

Byte 6 defines the mode of operation directed by the SPI master (8, 16 or 32-bit modes) while the DA1458x SPI slave answers with ACK/NACK regarding the reception of the length bytes. The mode is encoded as follows:

- 0x00 = 8-bit mode
- 0x01 = 16-bit mode
- 0x02 = 32-bit mode
- 0x03 = Reserved

Byte 8 is the last control byte, where DA1458x replies with ACK/NACK regarding the reception of the CRC and the mode, while the external SPI master starts sending the first byte of the payload (least significant byte of the first word).

The data section is presented in Table 3, taking into consideration the instructed mode. The stream of data is followed by 2 extra empty slots to provide the required time to the DA1458x SPI controller to compute the CRC and answer with ACK/NACK.

Upon completion of the SPI master process, all related pads are set to input and pulled down.

Table 3: SPI Master data communication

Slot number	MOSI (8-bit mode)	MOSI (16-bit mode)	MOSI (32-bit mode)	MISO
0	byte 0	byte 1, byte 0	byte 3, byte 2, byte 1, byte 0	-
1	byte 1	byte 3, byte 2	byte 7, byte 6, byte 5, byte 4	-
…				
4*Len-1 or 2*Len-1 or Len-1	byte (4*Len–1)	16-bit word (2*Len-1)	32-bit word (Len-1)	-
	all 0x00	all 0x00	all 0x00	all 0xAA
	all 0x00	all 0x00	all 0x00	ACK: 0x02 NACK: 0x20

6.2 DA1458x connected to UART

The boot code enters this mode configuring the UART controller with different baud rate parameters depending on the pin mapping as shown in Table 4.

Table 4: UART baud rates on different pins while booting

UTX	URX	Baud rate (kbit/s)
P0_0	P0_1	57.6
P0_2	P0_3	115.2
P0_4	P0_5	57.6
P0_6	P0_7	9.6

The rest of the UART parameters are common for all pin mapping, i.e.:

- Data Bits: 8
- Parity type: None
- Flow Control: None

The protocol required for establishing a successful communication and downloading the SW into the SysRAM is shown in Table 5.

Table 5: UART boot protocol

Byte number	DA1458x UTX	DA1458x URX
0	STX = 0x02	
1		SOH = 0x01
2		LEN_LSB
3		LEN_MSB

Byte number	DA1458x UTX	DA1458x URX
4	ACK = 0x06 or NACK = 0x15	
5 to N		SW code bytes
N+1	CRC (XOR over the SW code)	
N+2		ACK = 0x06

The protocol starts with the DA1458x UART TX pin transmitting 0x02 (Start TX, STX). The external device is expected to answer with a 0x01 (Start of Header, SOH) byte followed by 2 more bytes (LEN_LSB, LEN_MSB) which define the length of the code to be downloaded (first byte is the least significant, second the most significant). The DA1458x answers with 0x06 (ACK) if 3 bytes have been received and SOH has been identified or with 0x15 (NACK) if anything went wrong.

At this point the connection has been successfully established and the SW code will start being downloaded. The next N bytes are received and placed into the SysRAM, starting at address 0x20000000 as shown in Table 6.

Table 6: SysRAM word alignment

Address	Byte 3 (MSB)	Byte 2	Byte 1	Byte 0 (LSB)
0x20000000	Code byte 3	Code byte 2	Code byte 1	Code byte 0
0x20000004	…		Code byte 5	Code byte 4

Following the completion of the required code bytes, the boot code will calculate the CRC and send it over the URX. The booting sequence ends when reading the value 0x06 (ACK) at the URX line. CRC is calculated by XORing every successive byte with the previous value. Initial CRC value is 0x00.

During the final step of the boot code, the SYS_CTRL_REG register is programmed to:

1. Remap to SysRAM (SYS_CTRL_REG[REMAP_ADR0] = 10).
2. Apply a SW reset, so the system starts executing code at the remapped address (SYS_CTRL_REG[SW_RESET] = 10).

6.3 DA1458x connected to SPI Slave

The boot code acts as the master and configures the DA1458x SPI controller with the following parameters:

- Data 8 bits mode
- Master role
- Mode 3: SPI clock is initially high and SPI phase is shifted by 90 degrees.
- The SPI clock frequency is set at 2 MHz.

The protocol required for establishing a successful communication and downloading the SW into the SysRAM is shown in the following table where N is the number of dummy bytes as it defined in the Table 8 :

Table 7: SPI Slave boot protocol

Byte number	DA1458x MOSI	DA1458x MISO
0	Read command	-
1	Address byte 0 = 0x00	-
2	Address byte 1 = 0x00	-
3 to N	Dummy bytes = 0x00	-

Byte number	DA1458x MOSI	DA1458x MISO
N+1	-	‘p’ = 0x70
N+2	-	‘P’ = 0x50
N+3 to N+6	-	Dummy bytes
N+7	-	Code length MS byte
N+8	-	Code length LS byte
N+9 ...	-	Code bytes

The sequence as described in Table 7 is repeated for four different cases regarding the read command and the dummy byte parameters, as indicated in Table 8.

Table 8: SPI read and dummy byte cases

Case number	Read command opcode	Number of dummy bytes
0	0x03	0
1	0x03	1
2	0x0B	2
3	0xE8	5

As soon as the length has been received (2 bytes), the actual downloading of the code into the SysRAM starts. The start address is the base address of the SysRAM. The byte alignment is according to Table 6.

During the final step of the boot code a SW reset is given and the system starts executing the downloaded code.

6.4 DA1458x connected to I2C Slave

The boot code initialises the I2C controller in master mode with the following parameters:

- I2C slave address = 0x50 (7-bit address)
- I2C speed to standard mode (100 kbit/s)

The boot code initially scans to find an I2C slave device at addresses 0x50 up to 0x57. Following a successful slave address identification, a specific protocol is executed for downloading the SW into the SysRAM as shown in Table 9. If unsuccessful, a timeout is programmed to expire after 20 ms to get the chip out of the I2C booting mode into the next one.

Table 9: I2C boot protocol

Byte number	DA1458x SDA	Action (DA1458x I2C master)
0	0x70	Read command
1	0x50	Read command
2	Code length MS byte	Read command
3	Code length LS byte	Read command
4	CRC over Code only	Read command
5 to 31	Dummy	Read command
32 to Length+32	Code data	Read command

The boot code will calculate the CRC by XORing every successive byte with the previous value. Initial CRC value is 0x00. The CRC is calculated on multiples of 32 bytes. Padding with zeros is required when the payload size is not a multiple of 32 bytes.

During the final step of the boot code a SW reset is given and the system starts executing the downloaded code.

7 Timing details for DA1458x

The time required for the execution of the BootROM code which checks and enables booting from an external serial device is 76.5 ms (DA14581) or 146 ms (DA14580). This is illustrated in Figure 3 and Figure 4 which displays the power-up sequence of the DA1458x and pins P0_0 to P0_7, which are involved in the 11 steps as explained in Table 1.

Notes:

- The following figures are shown the BootROM execution starts at power up and not by RESET triggering.
- No actual device is connected so that the whole sequence is executed up to the last step.

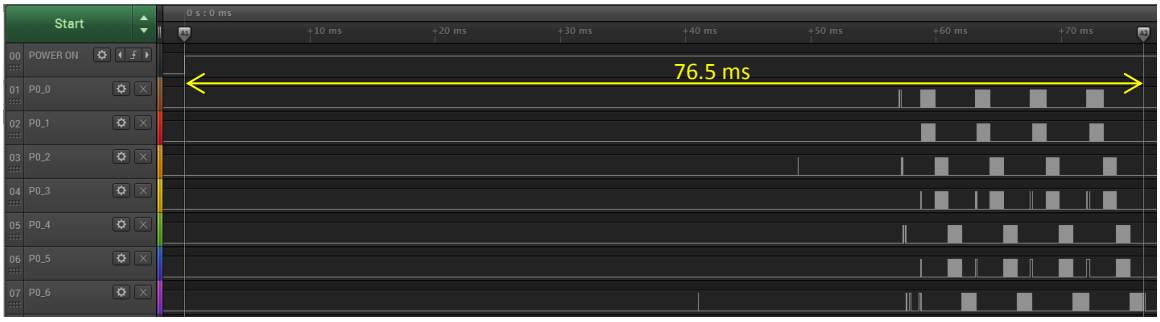

Figure 3: DA14581: Scan timing for booting from external serial devices

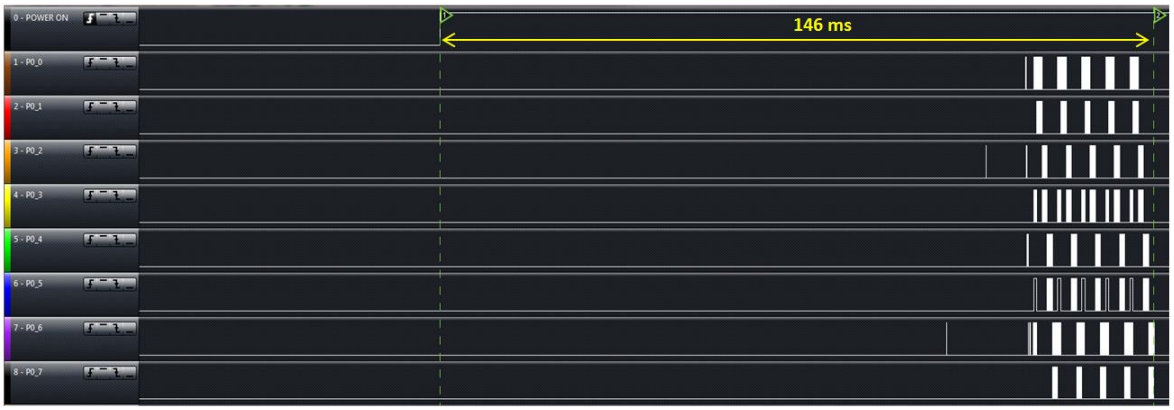

Figure 4: DA14580: Scan timing for booting from external serial devices

The first two booting steps turn the pins into inputs, thus expecting an external SPI master device to provide clock and data. This starts 41 ms (if DA14581) or 103 ms (if DA14580) after the actual power-up of the chip (as illustrated in Figure 5 and Figure 6), which is required for internal settling and executing the initial BootROM instructions up to the point of getting into the DevMode.

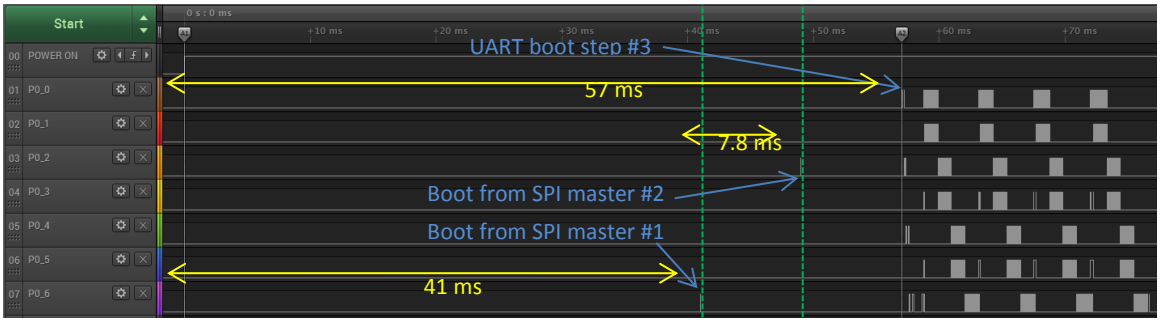

Figure 5: DA14581: Boot timing from SPI Master

Figure 6: DA14580: Boot timing from SPI Master

The next serial interface to be examined for a potential booting sequence is the UART. This spans over booting steps 3 to 6, where the DA1458x is transmitting on the TX line and waits for an answer on its RX line. The whole sequence time may vary from 2 ms up to 240 ms, depending on whether the RX line is high or low.

When RX is high, it is acknowledged to be a valid UART state and thus the DA1458x will wait for a response for 208 µs. This is displayed in Figure 7 in step 3: P0_0 is the TX signal while P0_1 is the RX which is high during transmission. This triggers the DA1458x to wait for the timer expiration after 208 µs, concluding that there is indeed no external device mounted on these pins and thus it should move to the next step.

In the cases of steps 4 to 6 the RX line is low, so the timer is not triggered and the DA1458x just continues on the next step without timer expiration.

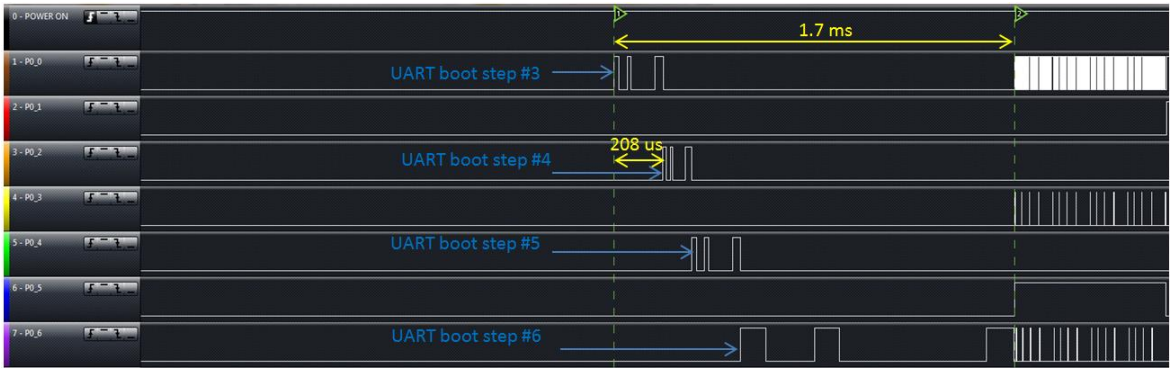

Figure 7: Boot timing from UART

A better overview of the booting from UART sequence and timing is given in Figure 8, where steps 4 to 6 are zoomed in. Note that the baud rate is different for each step with steps 3 and 5 expecting communication at 57.6 kbit/s, while steps 4 and 6 use 115.2 kbit/s and 9.6 kbit/s respectively.

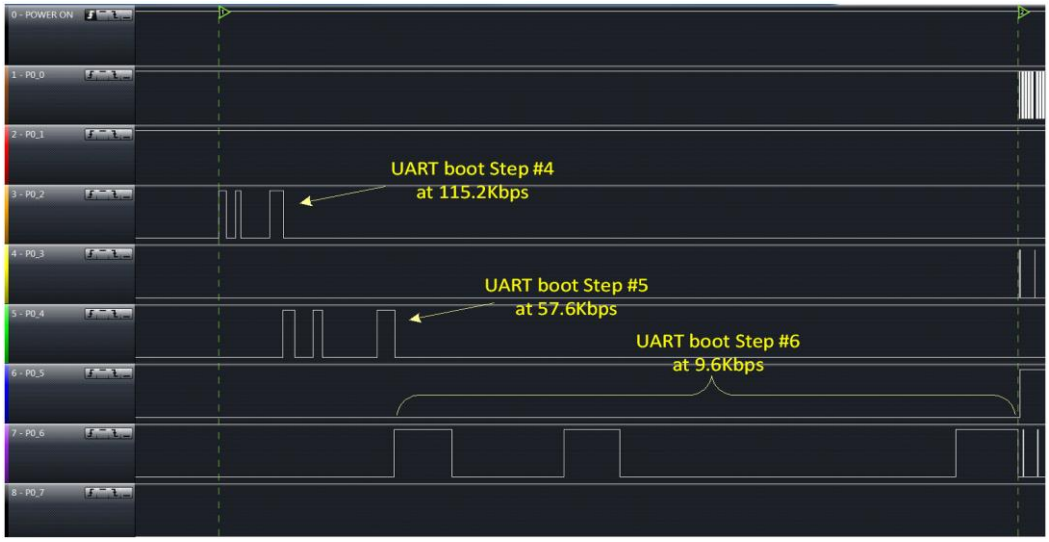

Figure 8: Boot timing from UART: zoomed steps 4 to 6

Following the UART booting, the DA1458x searches for an external SPI slave device, as explained in step 7. The clock output (at P0_0) of the DA1458x is toggling at 2 MHz while the whole process takes 640 us as displayed in Figure 9. Note here that booting from an external SPI slave is repeated 4 times before moving to the next step.

Figure 9 also shows the I2C booting process. It consists of 4 different steps that run the SCL (clock) line of the I2C at 100 kHz and require 1 ms for identifying that no external device is connected on these pins.

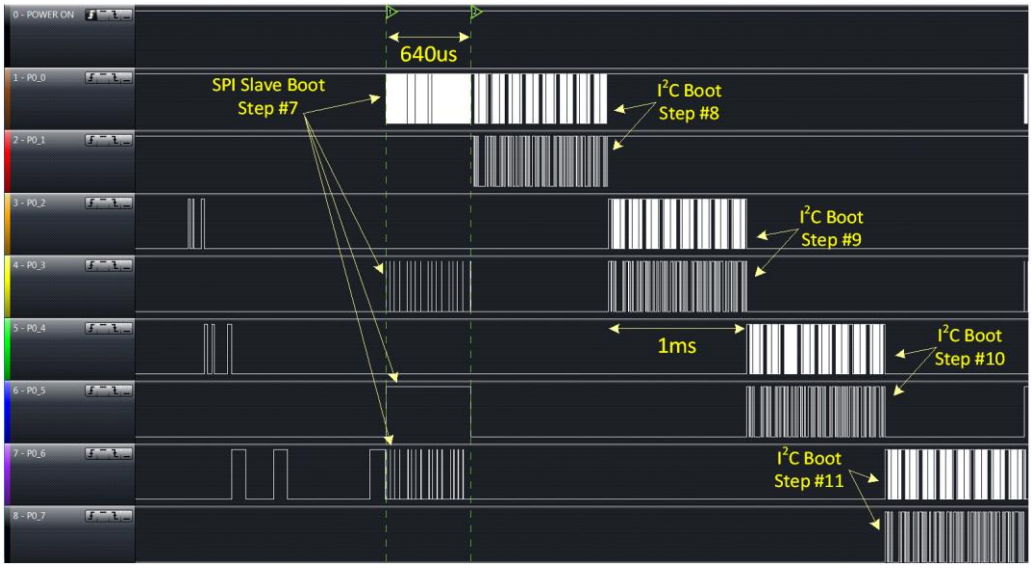

Figure 9: SPI Slave and I2C boot timing

The steps of Figure 9 are repeated up to 5 times when no device is successfully identified on any of the pins or interfaces. Each iteration lasts 4.8 ms. The timing overview is presented in Figure 10.

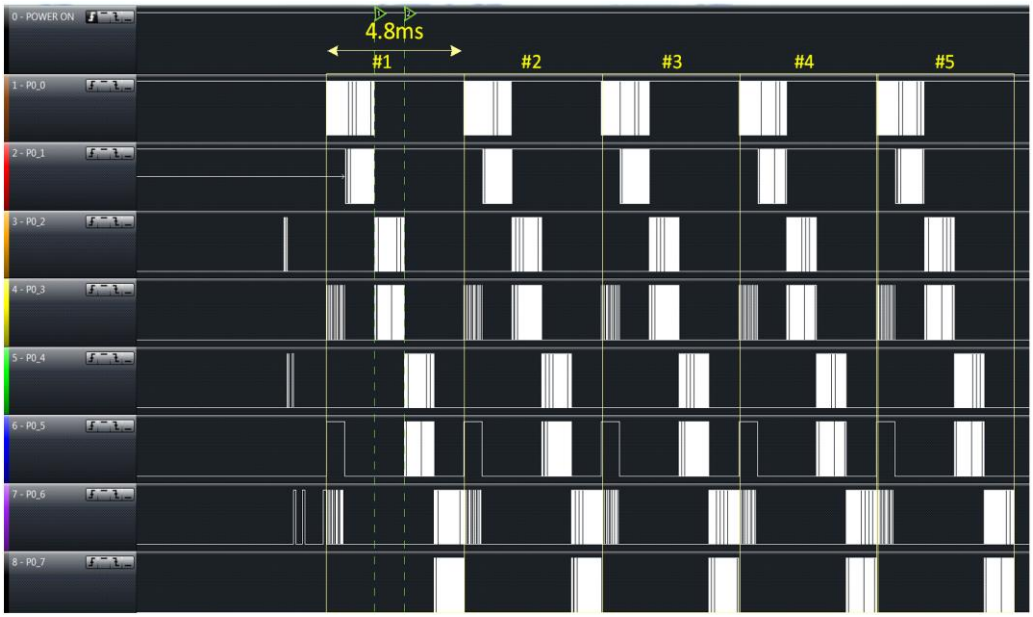

Figure 10: Overview of 5 SPI/I2C boot iterations

Revision history

Revision	Date	Description
1.0	26-Apr-2013	Initial version.
1.1	17-Sep-2013	Added SPI Master and UART baud rates.
1.2	03-Dec-2013	Corrected Table 1 regarding the SPI Master pin assignments.
1.3	27-Dec-2013	Added notes about the CRC calculation of the I2C booting sequence.
1.4	22-Sep-2014	Changes implemented: ● Added timing information for scanning serial interfaces while in Development Mode. ● Corrected the number of internal boot from SPI slave iterations. ● Updated Figure 1, Figure 2 and Figure 3. ● Converted to new template.
2.0	21-Oct-2014	Changes implemented: ● Title changed to ‘DA1458x Booting from serial interfaces’. ● Added Section 5 (Booting sequence). ● Section 6.1: added SPI clock note. ● Section 7: added timing diagrams for DA14581 (Figures 3 and 5).
2.1	30-Mar-2018	Changes implemented: ● Minor text revisions

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.

Disclaimer

Information in this document is believed to be accurate and reliable. However, Dialog Semiconductor does not give any representations or warranties, expressed or implied, as to the accuracy or completeness of such information. Dialog Semiconductor furthermore takes no responsibility whatsoever for the content in this document if provided by any information source outside of Dialog Semiconductor.

Dialog Semiconductor reserves the right to change without notice the information published in this document, including without limitation the specification and the design of the related semiconductor products, software and applications.

Applications, software, and semiconductor products described in this document are for illustrative purposes only. Dialog Semiconductor makes no representation or warranty that such applications, software and semiconductor products will be suitable for the specified use without further testing or modification. Unless otherwise agreed in writing, such testing or modification is the sole responsibility of the customer and Dialog Semiconductor excludes all liability in this respect.

Customer notes that nothing in this document may be construed as a license for customer to use the Dialog Semiconductor products, software and applications referred to in this document. Such license must be separately sought by customer with Dialog Semiconductor.

All use of Dialog Semiconductor products, software and applications referred to in this document are subject to Dialog Semiconductor's Standard Terms and Conditions of Sale, unless otherwise stated.

RoHS Compliance

Dialog Semiconductor complies to European Directive 2001/95/EC and from 2 January 2013 onwards to European Directive 2011/65/EU concerning Restriction of Hazardous Substances (RoHS/RoHS2).
Dialog Semiconductor's statement on RoHS can be found on the customer portal https://support.diasemi.com/. RoHS certificates from our suppliers are available on request.

Contacting Dialog Semiconductor

United Kingdom (Headquarters)
Dialog Semiconductor PLC
Phone: +44 1793 757700

Germany
Dialog Semiconductor GmbH
Phone: +49 7021 805-0

The Netherlands
Dialog Semiconductor B.V.
Phone: +31 73 640 8822

Email:
enquiry@diasemi.com

North America
Dialog Semiconductor Inc.
Phone: +1 408 845 8500

Japan
Dialog Semiconductor K. K.
Phone: +81 3 5425 4567

Taiwan
Dialog Semiconductor Taiwan
Phone: +886 281 786 222

Web site:
www.dialog-semiconductor.com

Singapore
Dialog Semiconductor Singapore
Phone: +65 64 849929

China
Dialog Semiconductor China
Phone: +86 21 5178 2561

Korea
Dialog Semiconductor Korea
Phone: +82 2 3469 8291