Application Note

DA1468x Booting from Serial Interfaces

AN-B-046

Abstract

The DA1468x can boot from external serial devices to enable development of the application code or to connect to an external (micro)controller. 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 DA1468x.

Note: This Application Note applies to the DA1468x-01 and later.

DA1468x Booting from Serial Interfaces

Contents

Abstract ..... Error! Bookmark not defined.
Contents 2
Figures 2
Tables 2
1 Terms and definitions 3
2 References 3
3 Introduction 4
4 Booting 5
4.1 Booting sequence 5
4.2 Serial booting sequence 5
5 Booting protocols 7
5.1 DA1468x initial boot sequence 7
5.2 DA1468x connected to SPI Master 7
5.3 DA1468x connected to UART 9
5.4 DA1468x connected to SPI Slave 10
5.5 DA1468x connected to I2C Slave 11
5.6 Configuration of a specific serial interface 12
6 DA1468x connected to QSPI flash 15
7 Timing details for DA1468x 16
Revision history 17

Figures

Figure 1: SPI transmitting device ID 7
Figure 2: UART transmitting device ID 9
Figure 3: I2C Slave address scanning 11
Figure 4: Flash image without header Flash image including header. 15
Figure 5: QSPI flash header 15
Figure 6: DA1468x: Scan timing for booting from external serial devices 16

Tables

Table 1: Pin assignment and booting sequence from external devices 6
Table 2: SPI Master boot protocol 7
Table 3: SPI Master data communication 8
Table 4: UART baud rates on different pins while booting 9
Table 5: UART boot protocol 9
Table 6: SysRAM word alignment 10
Table 7: SPI Slave boot protocol 10
Table 8: SPI read and dummy byte cases 11
Table 9: I2C boot protocol 12
Table 10: OTP header 12
Table 11: Chip Configuration Section 13
Table 12: Serial Pin functions 13

© 2022 Renesas Electronics

1 Terms and definitions

OTP	One Time Programmable (memory)
SW	Software
URX	UART Receive port
UTX	UART Transmit port

2 References

1. DA14680, Datasheet Dialog Semiconductor
2. DA14681, Datasheet Dialog Semiconductor
3. DA14682, Datasheet Dialog Semiconductor
4. DA14683, Datasheet Dialog Semiconductor

 © 2022 Renesas Electronics

3 Introduction

The DA1468x 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 DA1468x. (Ref: 1,2,3,4)

DevMode will be entered when the OTP header address 0x7F8E9D0-0x7F8E9D7 contains a value 0x00 (64-bit), when read by the CPU. This implies that the ‘Product Ready’ flag in the OTP is not programmed and the DA1468x should switch to the DevMode, so that application code can be downloaded from an external device into the internal SRAM (SysRAM). When the OTP header contains the specific value 0xAA (64-bit) at these locations, the DA1468x will enter ‘Normal Mode’ and proceed with booting from a non-volatile memory (NVM) which can either be the OTP or the internal (DA14680/682) or external (DA14681/683) QSPI FLASH.

© 2022 Renesas Electronics

4 Booting

4.1 Booting sequence

The booting sequence of DA1468x has been optimised and can be accelerated by programming the OTP header to boot from a predefined serial interface (SPI, UART or I2C). A picture of the Boot sequence is available in the datasheet of the DA1468x. (Ref: 1,2,3,4)

4.2 Serial booting sequence

The BootROM code starts with the RC16 oscillator active but untrimmed, which provides an average frequency of 10.5 MHz in typical conditions. The BootROM code starts the watchdog timer, which will fire after about 5 seconds if not re-initialised.

Next, the OTP controller is initialised and two important configuration flags are read and evaluated: the sequence of the RAM cells and whether JTAG should be enabled or not. If the JTAG is disabled during boot-up, the application code can enable it again when necessary.

Following that, trim and calibration values are read from the OTP and stored into the respective retention registers. Note that the TCS Section of the OTP header (see Table 10) contains all register addresses and values that are being measured during production testing or any other values that are required to be retained. These values are all stored into their respective registers using a ‘while’ loop, which reads and evaluates all TCS slots. From that point onwards, the trimmed RC16 oscillator provides a clock frequency very close to 16 MHz.

The TCS values are protected using inverted redundancy. When a voltage dropout occurs while reading or writing the value, an incorrect redundancy check will re-initiate the copy action. When copying is still unsuccessful after 5 attempts, a hardware reset will be triggered.

After a very short delay of approximately 200 us, the XTAL16M oscillator is enabled.

The “Product Ready” flag defines whether the system should follow the ‘NVM’ or the ‘SERIAL’ booting paths of the flow chart. In the NVM case, the system is supposed to start executing code from a Non-Volatile memory (NVM), which can either be the OTP or the QSPI FLASH in any of the functional modes.

In the case of a non-‘Product Ready’ device, the system clock is switched to the XTAL16M after a delay of 4 ms. This is required for making sure the oscillator is settled. From this point there are two options in the BootROM code:

- Boot from a specific serial interface (UART, SPI or I2C); this provides the ability to directly download code from a specific serial interface. This is to be used in cases where an external MCU will boot the DA1468x. The configuration of the serial interface in terms of pin location, controller and speed is to be found in the OTP header as explained in chapter 5.5. The booter will try to identify the external device by running twice the protocol as described in chapter 5. If no device is identified, then it switches to the booting from **any** serial interface branch.
- Boot from any serial interface; to allow 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 DA1468x side expecting to communicate with an external slave device and SPI can also be slave expecting to communicate with an external master. All serial interfaces will be exercised twice using the protocols described in chapter 5. If no connection with any of the serial interfaces is achieved, the bootROM software checks if there is a valid image in the QSPI flash. If there is a valid image in the QSPI flash, the software execution continues by running the (application) code in the QSPI flash. If there is no valid image in the QSPI flash, the booting sequence terminates with the JTAG on (unless JTAG access is disabled in the OTP header). The sequence of the steps and the assignment of the pins is as explained in
- Table 1.

 © 2022 Renesas Electronics

DA1468x Booting from Serial Interfaces

Note: Since the DA14680 has pins 0-5 on Port 0 not externally connected (since they are connected internally to the QSPI flash), there will be no activity on these pins during boot-up and a limited number of serial interfaces will be used.

Table 1: Pin assignment and booting sequence from external devices

Pin	Step	Boot from SPI Master	Step	Boot from UART	Step	Boot from SPI Slave	Step	Boot from I2C Slave	Step	RUN from QSPI flash
P0_0					13/14	SPI_CLK			23	QSPI_CLK
P0_1	1/2	MISO	3/8	TX		MOSI	15/19	SCL		QSPI_D0
P0_2		MOSI		RX		MISO		SDA		QSPI_D1
P0_3		SPI_CLK	4/9	RX			16/20	SDA		QSPI_D2
P0_4		SPI_CS								QSPI_D3
P0_5				TX		SPI_CS		SCL		QSPI_CS
P1_0			5/10	TX			17/21	SCL		
P1_5				RX				SDA		
P1_2			6/11	TX						
P1_4				RX						
P1_3			7/12	TX			18/22	SCL		
P2_3				RX				SDA		

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 DA1468x’s SPI controller to Slave mode (try to boot from an external SPI Master device) and assign the SPI’s CLK to P0_3, CS to P0_4, MOSI to P0_2 and MISO to P0_1. If there is no response the first time, this step will be exercised a second time (step 2).

If nothing is found to be connected on these pins (the protocol is described in chapter 5), then the boot code continues with the next step (step 3) which looks for an external UART assigning P0_1 to UART TX and P0_2 to UART RX. The next steps cycle through the UARTS twice and finally cycle through the I2C twice.

After step 22, when every serial port has been probed 2 times, no response is received from any of the serial interfaces, the QSPI controller is activated and the QSPI flash is probed if there’s a valid image programmed. If a valid image is found, the boot code will branch and execute this code. If no valid image is found, the DA1468x will enter a loop, waiting for a debugger to connect through SWDIO/SWCLK.

© 2022 Renesas Electronics

5 Booting protocols

5.1 DA1468x initial boot sequence

The boot code initially configures the DA1468x SPI controller with the following parameters:

- 8 bit mode
- Master role
- Mode 0: SPI clock (generated by DA1468x) starts low and SPI phase is zero.
- The SPI clock frequency is set at 4 MHz.

The SPI interface starts transmitting the device ID twice: DA1468xxxxDA1468xxxx (where xxxx is replaced by the actual device type).

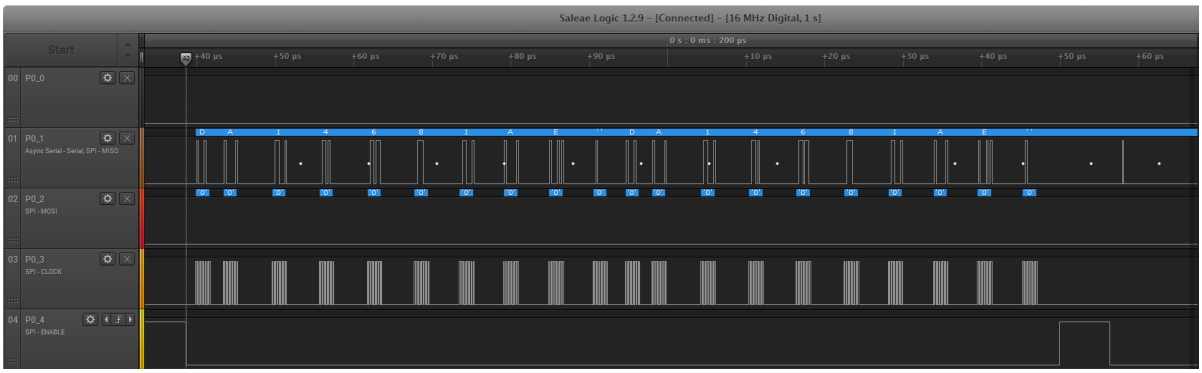

Figure 1: SPI transmitting device ID

5.2 DA1468x connected to SPI Master

After the transmission is finished the bootcode configures the DA1468x 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 protocol required for establishing a successful communication and downloading the SW into the SysRAM (DataRAM) is depicted in Table 2.

Note: The master SPI device generates the SPI clock for the DA1468x. 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 nr.	DA1468x MOSI	DA1468x 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	-

© 2022 Renesas Electronics

Byte nr.	DA1468x MOSI	DA1468x MISO
6	Mode byte	Length ACK:0x02 Length NACK:0x20
7	Empty: 0x00	-
8	Data bytes	Code/Mode ACK:0x02 Code/Mode NACK:0x20

The external SPI master device starts by sending the Preamble bytes (0x70 and 0x50) followed by a zero byte. The DA1468x will confirm 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 DA1468x 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 DA1468x 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 DA1468x SPI controller to compute the CRC and answer with ACK/NACK.

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] = 0x3).
2. Apply a SW reset, so the system starts executing code at the remapped address which depends on the sequence of the DataRAM settings ((SYS_CTRL_REG[REMAP_RAMS] = 0x0/1/2/3)

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

Table 3: SPI Master data communication

Slot nr.	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

© 2022 Renesas Electronics

5.3 DA1468x 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_1	P0_2	115.2
P0_5	P0_3	57.6
P1_0	P1_5	57.6
P1_2	P1_4	57.6
P1_3	P2_3	57.6

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

- Bits: 8
- No parity

Initially every UART configuration starts transmitting the device ID twice: DA1468xxxx(\r\n)DA1468xxxx(\r\n) (where xxxx is replaced by the actual device type).

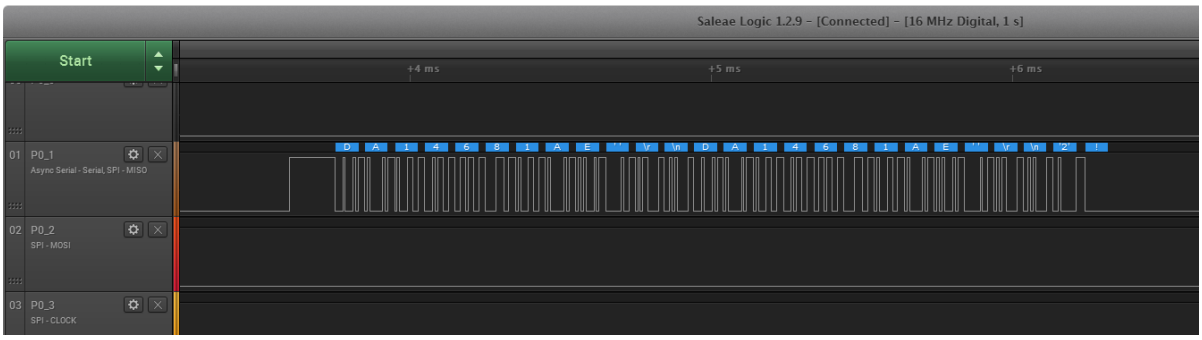

Figure 2: UART transmitting device ID

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

Table 5: UART boot protocol

Byte nr.	DA1458x UTX	DA1458x URX
0	STX = 0x02	
1		SOH = 0x01
2		LEN_LSB
3		LEN_MSB
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 DA1468x UART TX pin transmitting the device ID twice, followed by 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

 © 2022 Renesas Electronics

downloaded (first byte is the least significant, second the most significant). The DA1468x 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 0x7FC0000 as shown in Table 6.

Table 6: SysRAM word alignment

Address	Byte 3 (MSB)	Byte 2	Byte 1	Byte 0 (LSB)
0x7FC0000	Code byte 3	Code byte 2	Code byte 1	Code byte 0
0x7FC0004	…		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] = 0x3).
2. Apply a SW reset, so the system starts executing code at the remapped address which depends on the sequence of the DataRAM settings ((SYS_CTRL_REG[REMAP_RAMS] = 0x0/1/2/3)

5.4 DA1468x connected to SPI Slave

The boot code configures the SPI with the following parameters:

- 8 bit 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 (DataRAM) is shown in Table 7.

Table 7: SPI Slave boot protocol

Byte nr.	DA1468x MOSI	DA1468x MISO
0	Read command	-
1	Address byte 0 = 0x00	-
2	Address byte 1 = 0x00	-
3 to N	Dummy bytes = 0x00	-
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.

© 2022 Renesas Electronics

Table 8: SPI read and dummy byte cases

Case nr.	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 (0x7FC0000). The byte alignment is according to Table 6.

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] = 0x3).
2. Apply a SW reset, so the system starts executing code at the remapped address which depends on the sequence of the DataRAM settings ((SYS_CTRL_REG[REMAP_RAMS] = 0x0/1/2/3)

5.5 DA1468x connected to I2C Slave

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

- I2C slave address = 0x50 0x57(7-bit address)
- I2C speed to full speed mode (400 Kbit/s)

The boot code initially scans to find an I2C slave device at addresses 0x50 up to 0x57.

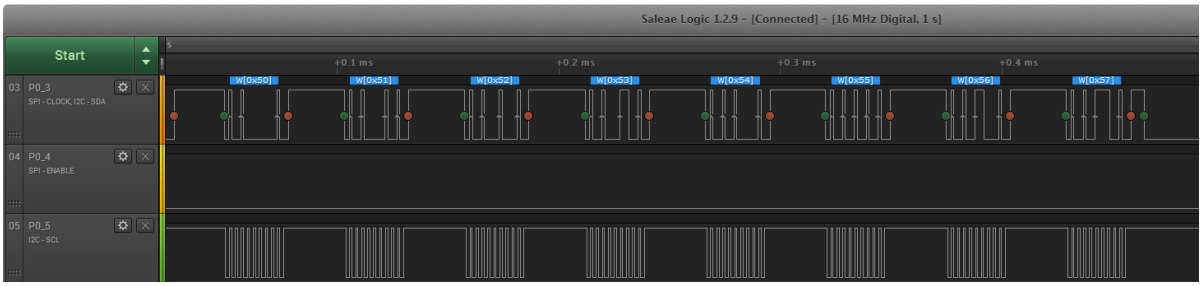

Figure 3: I2C Slave address scanning

Following a successful slave address identification, a specific protocol is executed for downloading the SW into the SysRAM (DataRAM) as shown in Table 9. If unsuccessful on all probed addresses, using I2C pin pair (P0_1, P0_2), the next pair (P0_5, P0_3) will be checked at full speed (400 Kbit/s). If still unsuccessful the last 2 I2C pairs (see

Table 1) will be checked at standard speed (100 Kbit/s).

© 2022 Renesas Electronics

Table 9: I2C boot protocol

Byte nr.	DA1468x SDA	Action (DA1468x I2C master)
0	0x50	Read command
1	0x51	Read command
2	0x52	Read command
3	0x53	Read command
4	0x54	Read command
5	0x55	Read command
6	0x56	Read command
7	0x57	Read command
N (depends on which I2C address responds)	Code length MS byte	Read command
N+1	Code length LS byte	Read command
N+2	CRC over Code only	Read command
N+5 to N+31	Dummy	Read command
N+32 to Length+N+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, the SYS_CTRL_REG register is programmed to:

1. Remap to SysRAM (SYS_CTRL_REG[REMAP_ADR0] = 0x3).
2. Apply a SW reset, so the system starts executing code at the remapped address which depends on the sequence of the DataRAM settings ((SYS_CTRL_REG[REMAP_RAMS] = 0x0/1/2/3)

Upon completion of the serial boot process, all related pads are set to input and pulled down except Pin P0_5 which is configured as output, pulled high (VDDIO).

Note for all serial interfaces: Since there are only 2 code length bytes, the maximum download size is 64 kB.

5.6 Configuration of a specific serial interface

The DA1468x contains a 64 kB One Time Programmable (OTP) memory, which is used for storing application code and for retaining the system's configuration data in a special OTP space called the "OTP header".

The OTP header occupies the last 712 words (64 bits wide) in the OTP memory space. It is partitioned into four sections that contain vital information for the system, as illustrated in the following table:

Table 10: OTP header

Address	Size (Bytes)	Section Name
0x07F8E9C0	184	Chip Configuration Section (CCS)
0x07F8EA78	384	Trim and Calibration Section (TCS)
0x07F8EBF8	3072	Elliptic Curve Contents Section (ECS)
0x07F8F7F8	2048	QSPI FLASH Initialization Section (QFIS)

© 2022 Renesas Electronics

To configure the booting from a specific serial interface (SPI, UART or I2C), running at a specific speed, the serial port/pin combination(s), serial interface and speed can be programmed in the OTP header at address 0x7F8EA30 as outlined in Table 11 and Table 12.

Table 11: Chip Configuration Section

Address	Size(B)	Field Name	Description
0x07F8EA30	8	Serial Configuration Mapping	B0[7:4]: Serial signal 1, port number B0[3:0]: Serial signal 1, bit number B1[7:4]: Serial signal 2, port number B1[3:0]: Serial signal 2, bit number B2[7:4]: Serial signal 3, port number B2[3:0]: Serial signal 3, bit number B3[7:4]: Serial signal 4, port number B3[3:0]: Serial signal 4, bit number B4: Booting Method 0xAA: booting from a specific serial port (B5) and at a specific location (B0 to B3) 0x00: normal booting sequence B5: Serial Interface: 0x0: None 0x1: UART 0x2: UART2 0x3: SPI 0x4: SPI2 0x5: I2C 0x6: I2C2 B6: if UART/UART2 is selected: 0x0:115 KBaud, 0x1: 57.6 KBaud, 0x2: 38.4 KBaud, 0x3: 19.2 KBaud, 0x4: 9.6 KBaud SPI is not applicable since it is a slave interface if I2C/I2C2: 0x0: Standard Mode (100 kbps) 0x1: Fast Mode (400 kbps) B7: RESERVED
0x07F8EA50	8	UART STX timing	Defines the delay for booting from UART in units of 10 ms each.

Table 12: Serial Pin functions

Serial Signal number	UART	SPI	I2C
Serial Signal 1	Tx	SCL	SCL
Serial Signal 2	Rx	CS	SDA
Serial Signal 3	-	MISO	-

© 2022 Renesas Electronics

Serial Signal 4	-	MOSI	-

© 2022 Renesas Electronics

6 DA1468x connected to QSPI flash

If nothing is detected on any of the serial interfaces, the boot code reads the reset sequence from OTP (if programmed, it resets the QSPI flash and configures the QSPI controller in single SPI mode. After that the flash header is read.
The flash header consists of 8 bytes added on top of the binary image of the flash, shifting the first 248 bytes down 8 bytes:

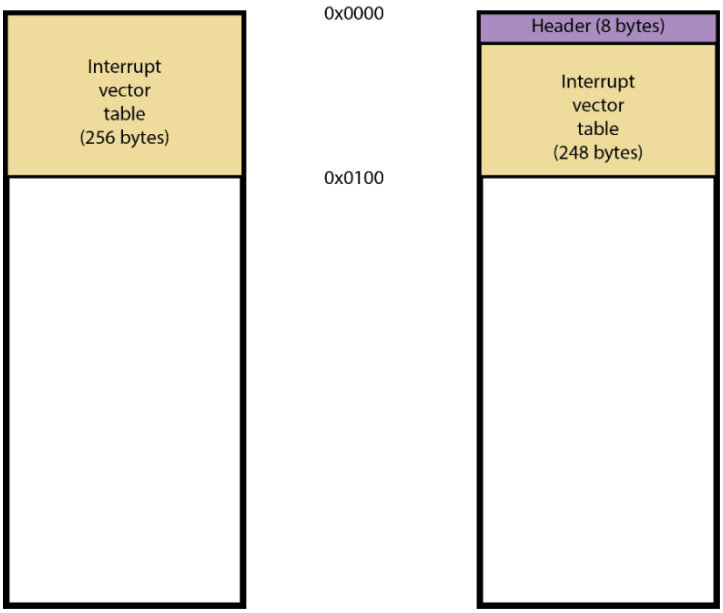

Figure 4: Flash image without header **Flash image including header**

The first 2 bytes (= magic word) are used to determine if the QSPI flash is programmed with a valid application. The next 2 bytes are currently not used and the last 4 bytes represent the length of the application image minus 0x80000000. The image length bytes are only relevant in ‘mirror’ mode, which is currently not supported, and not used in cached mode.

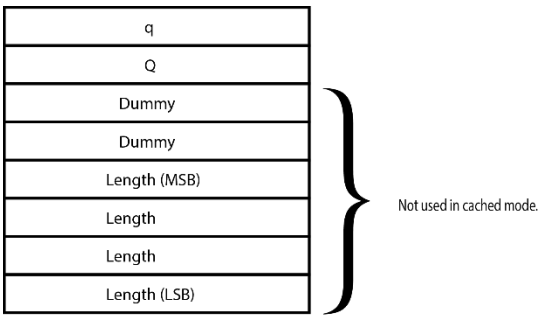

Figure 5: QSPI flash header

If the ‘magic’ word is written in the header (ASCII for “qQ”), the ROMbooter will start the flash download process. The boot code initializes the cache controller in quad mode and copies the Interrupt Vector Table (IVT) to the beginning of the dataRAM. Address 0 is remapped to the QSPI flash and the booting sequence ends with a software reset, which starts running the application in the QSPI flash.

If however the ‘magic’ word is not found at the beginning of the QSPI flash, the watchdog is disabled and system control is handed over to the JTAG/SWD interface.

© 2022 Renesas Electronics

7 Timing details for DA1468x

The time required for the execution of the BootROM code which checks and enables booting from an external serial device is approximately 77 ms. This is illustrated in Figure 6 which displays the power-up sequence of the DA1468x boot pins which are involved in the in the 23 steps as explained in

Table 1. The various serial interfaces and steps are clearly marked.

Note that no actual device is connected meaning that the whole sequence is executed up to the last step.

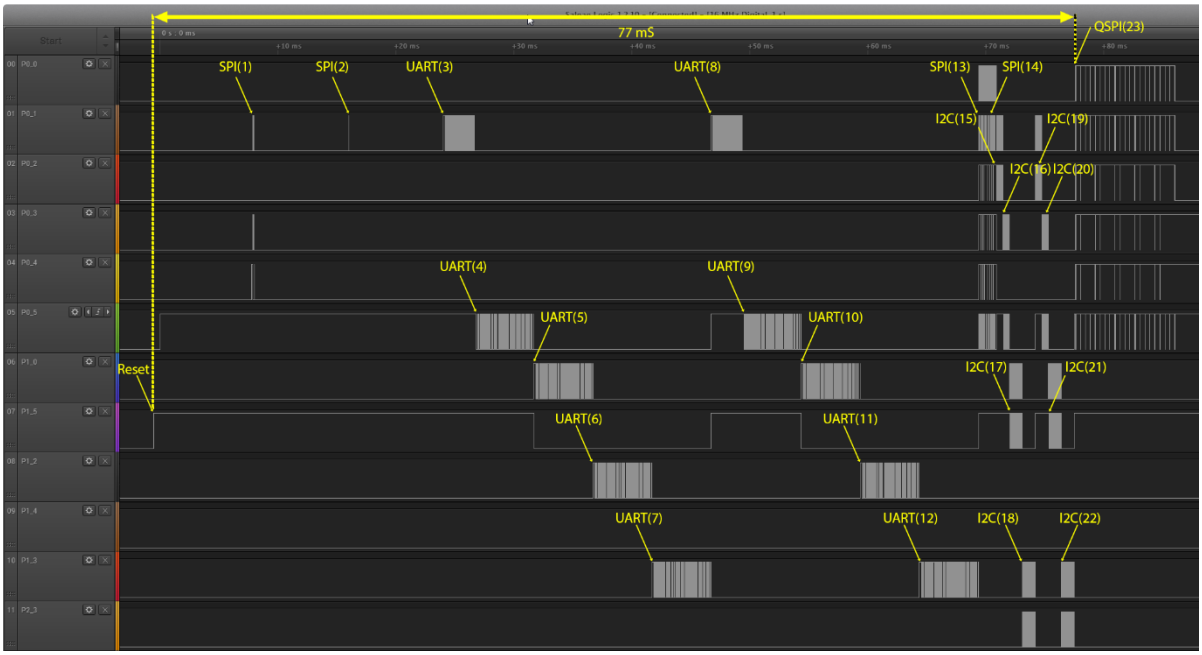

Figure 6: DA1468x: Scan timing for booting from external serial devices

© 2022 Renesas Electronics

Revision history

Revision	Date	Description
1.4	24-Jan-2022	Updated logo, disclaimer, copyright.
1.3	29-June-2020	Included reference to DA14682/683 datasheet. Removed the picture of the Boot sequence and referring to the datasheet.A
1.2	02-Nov-2017	Minor changes
1.1	27-Okt-2017	Added QSPI flash chapter
1.0	22-Jul-2016	Initial version.

© 2022 Renesas Electronics

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.

© 2022 Renesas Electronics