DECEMBER 27, 2013 V1.3

AN-B-001
DA14580 Booting from Serial Interfaces

Abstract

The DA14580 can boot from external serial devices when the OTP memory is not programmed, to enable development of the application code. The system enters this mode (Development Mode) at power up, with the boot code deciding what interface to boot from. This Application Note 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 DA14580.

Table of Contents

1.0 Introduction

The DA14580 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 data sheet of the DA14580.

DevMode will be entered if the OTP memory contains a value zero at the first two addresses, when read by the CPU. This implies that the OTP is not programmed and thus, the DA14580 should switch to the DevMode so that users get access to download code from external devices into the internal SRAM (SysRAM). However, if the OTP has specific values programmed (magic numbers) then the DA14580 will enter the Normal Phase and proceed with mirroring of the OTP contents into the SysRAM in the booting sequence as described in the data sheet.

To allow for maximum flexibility, a predefined number of pins are examined and utilized at boot, to communicate with external devices using the three serial interfaces available on chip: UART, SPI and I2C. SPI and I2C are considered to be masters on the DA14580 side expecting to communicate with an external slave device.

1.1 TERMS AND ABBREVIATIONS

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

1.2 REFERENCES

1. DA14580, Data sheet Dialog Semiconductor

1.3 HISTORY

April 26, 2013 Initial version

September 17, 2013 Added SPI Master and UART baud rates

December 03, 2013 Corrected Table 1 regarding the SPI Master pin assignments

December 27, 2013 Added notes about the CRC calculation of the I2C booting sequence.

2.0 Booting Pins

During boot, port P0 is used in checking for external devices. The assignment of the pins as well as the sequence of the interface activation by the boot code is

presented in the following table.

Table 1: Pin assignment and booting sequence from external devices

Pin	STEP	SPI MASTER	STEP	SPI MASTER	STEP	UART	STEP	SPI SLAVE	STEP	I2C
P0_0	1	SCK	2	SCK	3	TX	7	SCK	8	SCL
P0_1				CS		RX				SDA
P0_2				MISO	4	TX			9	SCL
P0_3		CS		MOSI		RX		CS		SDA
P0_4					5	TX			10	SCL
P0_5		MOSI				RX		MISO		SDA
P0_6		MISO			6	TX		MOSI	11	SCL
P0_7						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. So, the first step of the boot code is to configure the DA14580'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 nothing is found to be connected on these pins (the protocol is described in section 3.0), then the boot code continues with the second step which looks for an external SPI master at P0_0, P0_3, P0_2 etc. The booting sequence is finished at step 11, if no response is received to P0_6(SCL) and P0_7(SDA) and starts all over again 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 DA14580 boot code will end in an endless (while) loop with the Serial Wire interface (JTAG) activated.

3.0 Booting Protocols

3.1 SPI MASTER

The boot code initially configures the DA14580 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 a successful communication establishment and the SW downloading into the System RAM is depicted in the following table:

Table 2: SPI Master boot protocol

Byte #	DA14580 MOSI	DA14580 MISO
0	Preamble: 0x70	-
1	Preamble: 0x50	-
2	Empty: 0x00	-
3	Length LS Byte	Pream. ACK: 0x02 Pream. NACK: 0x20
4	Length MS Byte	0x00
5	CRC Byte	0x00
6	Mode Byte	Length ACK: 0x02 Length NACK: 0x20
7	Empty: 0x00	0x00
8	Data Bytes Byte/Word 0	Code/Mode ACK: 0x02 Code/Mode NACK: 0x02

The external SPI master device starts by sending the Preamble bytes (0x70 and 0x50) followed by a zero byte. The DA14580 will confirm the reception of the Preamble with 0x02 (Acknowledged) or 0x20 (Not Acknowledged) in case something went wrong. Bytes 3 and 4 are defining the length of the payload to follow. The least significant byte is sent first. The length is a number which depicts the amount of data in 32-bit words.

Following that, the SPI master must send the calculated CRC value of the payload to be sent. CRC is calculated by XORing every successive byte with the

previous value. Initial CRC value is 0xFF.

Byte 6 is defining the Mode of operation directed by the SPI master (8, 16 or 32-bit modes) while the DA14580 SPI slave is answering with ACK or NACK regarding the length bytes reception. 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 DA14580 replies with ACK/NACK regarding the reception of the CRC and the mode, while the external SPI master starts sending the first byte (least significant byte of first word) of the payload.

The data section is presented in the following table, taking into consideration the instructed mode. The train of data is followed by 2 extra empty slots to provide the required time to the DA14580 SPI controller to compute the CRC and answer with ACK/NACK.

Table 3: SPI Master data communication

Slot #	MOSI 8-bit	MOSI 16-bit	MOSI 32-bit	MISO
0	Byte0	Byte1-Byte0	Byte3-Byte2-Byte1-Byte0	-
1	Byte1	Byte3-Byte2	Byte7-Byte6-Byte5-Byte4	-
...				
4*Len-1 or 2*Len-1 or Len-1	Byte(4*Len)-1	16bit_word(2*Len)-1	32bit_word(Len-1)	-
	All 0x00	All 0x00	All 0x00	All 0xAA
	All 0x00	All 0x00	All 0x00	ACK: 0x02 NACK: 0x20

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

3.2 UART

The boot code enters this mode configuring the UART controller with different baud rate parameters depending on the pins mapping as shown in the following table:

Table 4: UART baud rates on different pins while booting

UTX	URX	Baud rate (kbps)
P0_0	P0_1	57600
P0_2	P0_3	115200
P0_4	P0_5	57600
P0_6	P0_7	9600

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

- Bits: 8
- No parity

The protocol required for a successful communication establishment and the SW downloading into the Sys-RAM is depicted in the following table:

Table 5: UART boot protocol

Byte #	DA14580 UTX	DA14580 URX
0	STX = 0x02	
1		SOH = 0x01
2		LEN_LSB
3		LEN_MSB
4	ACK = 0x06 or NACK = 0x15	
5 ... N		SW Code Bytes
N+1	CRC (XOR over the SW Code)	
N+2		ACK = 0x06

The protocol starts with the DA14580 UART TX pin transmitting 0x02 (Start TX, STX). The external device is expected to answer 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 DA14580 answers with 0x06 (ACK) if 3 bytes have been received and SOH identified or with 0x15 (NACK) if anything went wrong.

At this point 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 follows:

Table 6: SysRAM word alignment

Address	Byte3 (MSB)	Byte2	Byte1	Byte0 (LSB)
0x20000000	Code Byte 3	Code Byte 2	Code Byte 1	Code Byte 0
0x20000004	...		Code Byte 6	Code Byte 5

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 with by 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 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).

3.3 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 a successful communication establishment and the SW downloading into the SysRAM is depicted in the following table:

Table 7: SPI boot protocol

Byte #	DA14580 MOSI	DA14580 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 - 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 4 is repeated for four different cases regarding the Read Command and the Dummy Bytes parameters as indicated in Table 8.

Table 8: SPI read and dummy byte cases

Case #	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, the SYS_CTRL_REG 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).

3.4 I2C

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

Table 9: I2C boot protocol

Byte #	DA14580 SDA	Action (DA14580 I2C master)
0	0x70	Read command
1	0x50	Read command
2	MSByte Code Length	Read command
3	LSByte Code Length	Read command
4	CRC over Code only	Read command
5 ... 31	Dummy	Read command
32 ... 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 if the payload size is not a multiple of 32 Bytes.

During the final step of the boot code, the SYS_CTRL_REG 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).