ON Semiconductor

Is Now

To learn more about onsemi™, please visit our website at
www.onsemi.com

AND8400/D

Bootloading BelaSigna® 300 Using the I²C Interface

ON Semiconductor®

http://onsemi.com

APPLICATION NOTE

INTRODUCTION

This application note describes how to bootload BelaSigna 300 through its I²C interface when it does not have an EEPROM attached (i.e. "bootstrapping"). This situation can occur when a Bluetooth® or a baseband chip, or any I²C-master capable chipset, is connected to BelaSigna 300 through the I²C port.

Since no EEPROM is attached to BelaSigna 300, the external device must have dedicated memory space in its non-volatile memory to store the BelaSigna 300 application. It can either be internal Flash, as is the case with some Bluetooth devices or external Flash / NAND Flash memories in Bluetooth or mobile phone applications.

This application note will provide some background information which is essential in understanding the bootloading process on BelaSigna 300 using the I²C interface. These sections deal with configuring the transfer mode for I²C, writing to the memory and verifying the CRC to make sure that the download was successful. For more information regarding the I²C protocol on BelaSigna 300, please refer to the *Communications Protocol Manual for BelaSigna 300*.

BACKGROUND INFORMATION

Transfer Mode during I²C Bootloading on BelaSigna 300

The Debug Port memory access commands contain a Transfer Mode byte that specifies the memory space, data width, and transfer size for the read or write operation. The Transfer Mode byte is described in Table 1.

Table 1. TRANSFER MODE

Bit(s)	Description	Values
7:5	Reserved	000
4	Transfer Size	0 - Multiple words 1 - Single word
3:2	Memory Space	00 – Reserved 01 - X memory 10 - Y memory 11 - P memory
1:0	Data Width	00 - 8-bit 01 - 16-bit 10 - 24-bit 11 - 32-bit

Data is transferred with the most significant bytes first; for a 32-bit transfer, the sequence is:

1. Bits 31 to 24
2. Bits 23 to 16
3. Bits 15 to 8
4. Bits 7 to 0

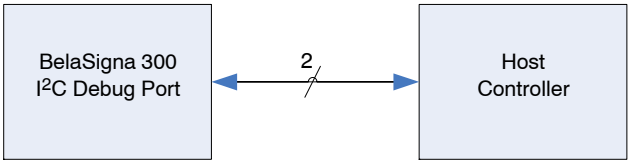
The data width is specified independently from the memory space. If the transfer data width is smaller than the actual memory width, the least significant bits are transferred. For example, for a 32-bit memory with an 8-bit transfer mode set, only bits 7 to 0 are transferred. When reading memory, the data is truncated from the actual memory width to the desired transfer width. When writing memory, the data is zero-extended from the transfer width to the actual memory width.

The debug port supports transferring either single words or multiple words. When the debug host is reading multiple words, the debug port queues up the next word to be sent, possibly triggering a side effect. For memory reads that cause side effects, the single word transfer can be used. In this mode, the debug port performs only a single access for the requested word.

When memory is being read in single word mode, subsequent bytes read from the debug port are all zero. Similarly, when memory is being written in single word mode, extra bytes received are ignored.

Write Memory during I²C Bootloading on BelaSigna 300

Writing memory is initiated through the write memory debug port command. The [TRANSFER_MODE] argument indicates the memory space and data width for the transfer as described in "Transfer Mode". The next two bytes specify the high and low bytes of the starting address.



The write memory transaction occurs in a single I^2C write transfer. After the debug port has acknowledged the memory address, the host controller can begin sending data to be written to memory. Depending on the transfer mode, the host controller sends one, two, three, or four bytes per word. Once the full word has been transmitted, the debug port queues the word for writing to memory and acknowledges the last byte. In this way the host controller can send data to the debug port continuously without the debug port having to stretch the clock. The debug port automatically increments the address by one after each write.

Command Byte	Syntax	Transmit State	Security Mode	CFX Run Mode
0x57 ('W')	[0x57] [TRANSFER_MODE] [ADDR 15:8] [ADDR 7:0] [DATA.0] [DATA.1] ... [DATA.n]	Status byte (2 bytes)	Unrestricted (Required)	Stopped (Required)

The debug port performs no special processing on the address. The address automatically wraps around when it reaches the end of the address range (i.e., when the address reaches 0xFFFF, the next word read is 0x0000).

Below is an example transaction for writing two words to X Memory (0xABCDEF to 0x0010 and 0x123456 to 0x0011) using the 24-bit transfer mode. Normal text indicates data sent from the debug host. Bold text indicates responses from the debug port.

[S] [ADDR][W]**[A]** [0x57]**[A]** [0x6]**[A]** [0x00]**[A]** [0x10]**[A]**
[0xAB]**[A]**[0xCD]**[A]** [0xEF]**[A]** [0x12]**[A]** [0x34]**[A]** [0x56]**[A]** [P]

Where:

[S] - I2C Start Condition

[ADDR] - 7-bit debug port I2C Address

[W] - Read/Write bit: 0 (Write) for commands

[R] - Read/Write bit: 1 (Read) for responses

[A] - Acknowledgement: ACK or NAK from debug port

[P] - I2C Stop Condition

Cyclic Redundancy Check (CRC) during I^2C Bootloading on BelaSigna 300

BelaSigna 300 uses a standard cyclic redundancy code (CRC) algorithm to ensure data integrity for the file system and all debug port communications. To put the debug port in the *CRC* transmit state so that the debug host can read the debug port checksum, we need to execute *[0x4D] Read and Reset CRC* command. This command stores the current CRC value to be sent to the debug host and resets the CRC value to 0xFFFF.

Command Byte	Syntax	Transmit State	Security Mode	CFX Run Mode
0x4D ('M')	[0x4D]	CRC	Any	Any

Use the *Read and Reset CRC* command to read the current CRC value and reset the CRC to 0xFFFF. The CRC includes all transferred bytes up to and including the *Read and Reset CRC* command. If the debug host reads the CRC using the next I^2C read transfer, the CRC bytes read by the host controller will be included in the next CRC calculation.

The debug port maintains a CRC of all bytes transmitted and received. The CRC is updated automatically after a complete byte has been transferred in either direction. This includes all bytes that are not acknowledged, invalid commands, or commands attempted in the incorrect security or run mode. If a byte is not completely transferred, it is not included in the CRC. The CRC does not include the address byte of I^2C transactions.

The debug port uses CRC-CCITT, which has the parameters described in Table 2.

Table 2. CRC-CCITT ALGORITHM PARAMETERS

CRC Parameter	Parameter Value
Order	16
Polynomial	$x^{16} + x^{12} + x^5 + 1$
Polynomial (hex)	0x1021
Initial Value (hex)	0xFFFF
Final XOR Value (hex)	0x0000

USING I²C PROTOCOL TO DOWNLOAD AND RUN AN APPLICATION

The following description shows how to connect to BelaSigna 300, initialize proper communications, set up restricted mode appropriately, download and run an application on the DSP.

Downloading Object Code

When developing software for BelaSigna 300, various file formats can be generated and used depending on the situation. Various options are available to extract the data directly from the executable created by the IDE. ON Semiconductor can provide a utility that converts the .o file format into a C-Header file (called *download_data.h*) as described later in this document. If you wish to use this utility, you will need to convert the executable created by the IDE into the .o format. This is accomplished with the utility absdump.exe, also included with the IDE. This conversion can be performed by manually running absdump.exe, or can be configured to occur automatically as part of your project build in a custom build step. To enable this automatic conversion, edit the build settings for your project, select the 'Custom Build' editor page and click the 'Create Default Custom Build File' button to generate a default custom build script (*custom_build.xml*). Edit this script and un-comment the lines indicated in both the 'pre-build' and post-build' steps to automatically delete and re-generate a .o file.

The CTK libraries can be used to develop PC software in Python or other languages like C++. However, these are only useful when you are using a PC and the Communication Accelerator Adaptor (CAA) to communicate with BelaSigna 300's I²C port. When developing embedded microcontroller software to communicate with BelaSigna 300, we cannot target these CTK libraries to the host controller. In this case, low-level communication functions must be implemented using the host controller's I²C master. Support can be provided by ON Semiconductor to implement the I²C protocol at this low level. Please contact us for more information regarding available support.

Downloading an Example

The following example assumes the use of the ON Semiconductor supplied C conversion utility applied on a .o file. It shows the structure of the generated C-header file, as well as the associated I²C commands needed for the transfer to BelaSigna 300.

In *download_data.h*:

```
struct DataBlock {
    unsigned short byteCount;
    unsigned short crc;
    unsigned char *formattedData;
} downloadBlocks[3] = {
  { 0x024c, 0x2679, downloadData0 },
  { 0x006c, 0x4c81, downloadData1 },
  { 0x0007, 0x481a, downloadData2 },
};
#define DOWNLOAD_BLOCK_COUNT 3
```

The above code presents a summary of the data that are parsed from the .o file. Every contiguous data block is declared as a unique block of data, in this example we can see three blocks of data. These blocks can be identified by their Byte Count, CRC Value and Formatted Data. The last element points to actual data. As an example, we have chosen the second data block (downloadData1) which represents the Interrupt Vector Table for this application. As seen from the block below, the Interrupt Vector Table is copied using the write memory command to P memory, address *0xFFE0*. Please refer to the second data block shown below:

```
unsigned char downloadData1[] = {
  CMD_WRITE_MEMORY, 0x0f, 0xff, 0xe0,
  0x3c, 0xd8, 0x04, 0x00,
  0x3b, 0x20, 0x10, 0x65,
  0x3b, 0x20, 0x10, 0x6a,
  0x3b, 0x20, 0x10, 0x6a,
  0x3b, 0x20, 0x10, 0x6a,
  0x3b, 0x20, 0x10, 0x6a,
  0x3b, 0x20, 0x10, 0x6a,
  0x3b, 0x20, 0x10, 0x6a,
  0x3b, 0x20, 0x10, 0x6a,
  0x3b, 0x20, 0x10, 0x6a,
  0x3b, 0x20, 0x10, 0x6a,
  0x3b, 0x20, 0x10, 0x6a,
  0x3b, 0x20, 0x10, 0x6a,
  0x3b, 0x20, 0x10, 0x6a,
  0x3b, 0x20, 0x10, 0x6a,
  0x3b, 0x20, 0x10, 0x6a,
  0x3b, 0x20, 0x10, 0x6a,
  0x3b, 0x20, 0x10, 0x6a,
  0x3b, 0x20, 0x10, 0x6a,
  0x3b, 0x20, 0x10, 0x6a,
  0x3b, 0x20, 0x10, 0x6a,
  0x3b, 0x20, 0x10, 0x67,
  0x3b, 0x20, 0x10, 0x6a,
  0x3b, 0x20, 0x10, 0x6a,
  0x3b, 0x20, 0x10, 0x6a,
  0x3b, 0x20, 0x10, 0x6a,
  0x3b, 0x20, 0x10, 0x6a,
};
```

The first line of each block of data contains the Write Memory command with its arguments. The syntax of the write command is given below. For more information about the Write Memory command and Transfer Modes please refer to the *Communication Protocol Manual for BelaSigna 300*.

[WRITE_MEM] [TRANSFER_MODE] [ADDR 15:8] [ADDR 7:0]

All the data blocks defined in the C-header file (*download_data.h*) are to be downloaded by sending the write memory command above.

Running the Application

Running the application on BelaSigna 300 is a two step process:

1. Change the Program Counter (PC) to point to the address 0x1000.
2. Run the core.

Please refer to the *Table 4* for more details.

A Complete Bootloading Example

Table 4 outlines the I^2C commands which are needed to be called to initialize BelaSigna 300, download and run the application.

Table 4: BELASIGNA 300 DOWNLOAD STEPS

#	Command	Master ↔ Slave	Description
1	Send Status Request 'Write to address 96' (0xC0) + 'S' (0x53)	→	The status word consists of two bytes. The status response is repeated indefinitely as long as the master continues to acknowledge the response bytes. This allows the master to poll for a change in the status response using a single I2C read transfer.
2	Read Status Response (2 bytes) 'Read from address 96' (0xC1)	→	For more information regarding the status response on BelaSigna 300, refer to the *Debug Port Status Response* table in the *Debug Port Protocol* chapter of the *Communication Protocols Manual for BelaSigna 300.*
3	Keep looping and Read Status until bit 11 of status response is 0 indicating security mode is unrestricted.	→	For more information regarding the status response on BelaSigna 300, refer to the *Debug Port Status Response* table in the *Debug Port Protocol* chapter of the *Communication Protocols Manual for BelaSigna 300*.
4	Stop Core 'Write' (0xC0) + 'P' (0x50)	→	Puts the debug port in Stopped mode, stopping the CFX DSP core.
5	Reset the loop counter (optional) by executing the following command four times 'Write' (0xC0) + 'O' (0x4F) + 0x3C + 0xD8 + 0x09 + 0x00	→	In the unlikely event the device was stopped in a hardware loop; wind down the loop counter by manually executing four ENDLOOP instructions. For more information, refer to *Run Control Commands* in the *Debug Port Protocol* chapter of the *Communication Protocols Manual for BelaSigna 300.*
6	Reset the status (SR) register 'Write' (0xC0) + 'F' (0x46) + '50' (0x32) + 0x00 + 0x00 + 0x00	→	For more information, refer to the *Normal Register Indexes* table in the *Debug Port Protocol* chapter of the *Communication Protocols Manual for BelaSigna 300.*
7	Download the program (follow these steps exactly to ensure the calculated CRC matches). For each memory block: 1. Read and Reset CRC 'Write' (0xC0) + 'M' (0x4D) 2. Write memory block 'Write' (0xC0) + downloadBlocks[n] 3. Read and Reset CRC 'Write' (0xC0) + 'M' (0x4D) 4. Read CRC value (2 bytes) 'Read' (0xC1) Compare the read CRC value with the calculated value in downloadBlocks[n] (in *download_data.h*)	→	The 'Write Memory' (0x57) command as well as the transfer mode precedes the data in downloadBlocks[n] (in *download_data.h*). For more information regarding the transfer mode, refer to the *Transfer Mode* table in the *Debug Port Protocol* chapter of the *Communication Protocols Manual for BelaSigna 300*.
8	Change program counter (PC) to 0x1000 'Write' (0xC0) + 'O' (0x4F) + 0x3B + 0x20 + 0x10 + 0x00	→	Single−Step the GOTO.0D instruction using the *Execute Instruction* command. For more information, refer to *Run Control Commands* in the *Debug Port Protocol* chapter of the *Communication Protocols Manual for BelaSigna 300.*
9	Start Core 'Write' (0xC0) + 'G' (0x47)	→	Puts the debug port into Running mode, allowing the CFX DSP core to run freely.

COMPANY OR PRODUCT INQUIRIES

For more information about ON Semiconductor, our technology and our products, visit our website at: http://www.onsemi.com.

PUBLICATION ORDERING INFORMATION

LITERATURE FULFILLMENT:
Literature Distribution Center for ON Semiconductor
P.O. Box 5163, Denver, Colorado 80217 USA
Phone: 303−675−2175 or 800−344−3860 Toll Free USA/Canada
Fax: 303−675−2176 or 800−344−3867 Toll Free USA/Canada
Email: orderlit@onsemi.com

N. American Technical Support: 800−282−9855 Toll Free USA/Canada
Europe, Middle East and Africa Technical Support:
Phone: 421 33 790 2910
Japan Customer Focus Center
Phone: 81−3−5773−3850

ON Semiconductor Website: **www.onsemi.com**

Order Literature: http://www.onsemi.com/orderlit

For additional information, please contact your local Sales Representative