



Estimating and Optimizing Booting Time for Blackfin® Processors

Contributed by Jiang Wu

Rev 1 – December 4, 2006

Introduction

This EE-Note describes how to estimate, measure, and optimize the booting time of a Blackfin® processor-based system. A Booting Time Estimation Tool (BTET) is included to help with the analysis of booting time. Though all measurements and testing described in this EE- Note were performed on an ADSP-BF533 EZ-KIT Lite® board, and all the descriptions are specific to ADSP-BF533 Blackfin processors, the methodology can be readily applied to other target Blackfin systems as well.

Booting Time

The booting time of an ADSP-BF533 Blackfin processor-based system is defined as the time between the deassertion of the processor's /RESET pin and the execution of the first line of application code. Its composition depends on the selected booting mode of the processor. There are four boot modes supported by the processor^[1]:

- Execution from external memory
- Boot from flash
- Boot from SPI slave mode
- Boot from SPI serial EEPROM master mode

In this EE-Note, the two most common boot modes will be the focus: boot from flash and boot from SPI serial EEPROM master mode. In this note, we will only discuss the case where the

processor boots into L1 memory, though some applications may boot into SDRAM.

For these two boot modes, the booting time consists of the time to:

- Run the built-in Boot ROM
- Load data/code from external memory space (flash memory or SPI serial EEPROM) to the processor's L1 memory
- Zero-fill the non-initialized data
- Execute initialization code, if there is any

Initialization code is usually application-specific, so it is assumed that there is no initialization code in the application.

Booting Process

Upon the deassertion of /RESET, the processor starts the boot process by executing the on-chip boot ROM, which determines the boot mode (flash or SPI serial EEPROM), sets up the proper DMA channels (read from external memory to L1 memory), and then puts the processor in an idle state awaiting for the completion of a DMA read transaction. After parsing the block of bytes read by the DMA, the boot ROM initiates another DMA transaction to load more bytes until all the bytes have been read. Note that the booting routine is executed from the boot ROM, which is slower than L1 memory.

Figure 1 graphically shows the boot process.

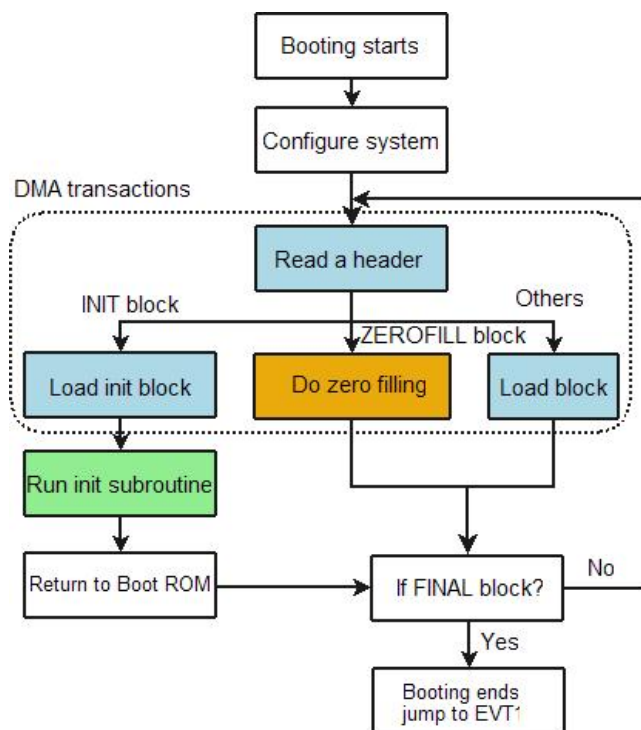


Figure 1. Booting Process Using On-Chip Boot ROM

The content of the flash/SPI serial EEPROM is the loader image (.ldr), which is created by the VisualDSP++® loader utility (elfloader.exe), based on one or more Blackfin executable (.dxe) files. It is organized as header/block pairs, as shown in Figure 2. A header is ten bytes long and provides the information about the block that follows. The boot ROM extracts this block of information by parsing headers and then determines what to do with the corresponding blocks.

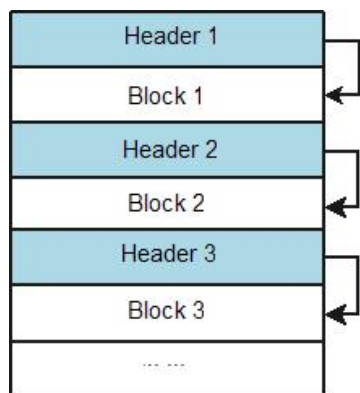


Figure 2. Structure of Loader File in Flash/EEPROM

A header includes:

- ADDRESS (4 bytes) — the address to which the block will be loaded to within the internal memory.
- COUNT (4 bytes) — the number of bytes of the block.
- FLAG (2 bytes) — block type and control commands.

Among the 16 bits of the FLAG word, only three are used for boot time estimation:

Bit 1: ZEROFILL — Indicates that the block is a data buffer with zeros and no actual block follows. It simply instructs the on-chip boot ROM to zero COUNT bytes starting from ADDRESS in memory. No data transfer from flash/SPI serial EEPROM is needed for this block.

Bit 3: INIT — Indicates that the block is an initialization block, which the boot ROM will call after loading into the internal memory (the initialization code must have an RTS at the end).

Bit 15: FINAL — Indicates that the following block is the last one. After processing a FINAL block, the boot ROM jumps to the reset vector address stored in the EVT1 register.

The on-chip boot ROM implements zero-filling using memory DMA transfers. The source is a single boot ROM memory location set to zero, and the destination is the data block specified by the header. For a complete description of the header format, refer to *ADSP-BF533 Blackfin Booting Process (EE-240)*^[2].

Measurement of Boot Time

To measure the boot time of a Blackfin processor-based system, a code snippet, shown in Listing 1, must be compiled into the loader file and programmed/burned to the flash/SPI serial EEPROM of the target system by the Flash Programmer tool provided with VisualDSP++. The system must be configured as “boot from flash” or “boot from SPI serial EEPROM” by

setting the `BMODE[1:0]` pins properly. Then, upon a system reset, the code is brought in by the boot ROM to the beginning of L1 memory (`0xFFA00000`). It first sets up one of the programmable flags (`PF0`) as an output, and then toggles it.

```
#include <defBF533.h>
#define RESERVE_SIZE /* depending on
the size of data file startup.dat */
#define FILL_SIZE 10 /* in KByte */

.section program;

start:
    p0.h = hi(FIO_DIR);
    p0.l = lo(FIO_DIR);
    r0.l = 0x0001;
    w[p0] = r0.l; /* set the flag */
    p0.l = lo(FIO_FLAG_T);
    w[p0] = r0.l; /* toggle PF0 */

wait_loop:
    jump wait_loop;
start.end: nop;

#ifdef FILL_SIZE
.section data1
.byte fill[FILL_SIZE*1024];
#endif
#ifdef RESERVE_SIZE
.byte reserve[] = "startup.dat";
#endif
```

Listing 1. Code Snippet for Measuring Booting Time

On the ADSP-BF533 EZ-KIT Lite board, these two signals (`/RESET` and `PF0`) can be accessed through the extension U-connector on the bottom of the board. The booting time will be the time gap between the rising edge of `/RESET` and the toggling edge of `PF0`. The execution time of the code snippet is negligible in comparison to the loading time and boot ROM execution time.

The `reserve` and `fill` buffers (defined in the code snippet of [Listing 1](#)) are used to change the size of the flash/SPI serial EEPROM loading and zero-filling, therefore facilitating the simulation of the impact of code sizes on booting time.

The `/RESET` pin pulse and flag pin toggle can be recorded by an oscilloscope's single trigger mode — `/RESET` as trigger source with positive edge, as shown in [Figure 3](#).

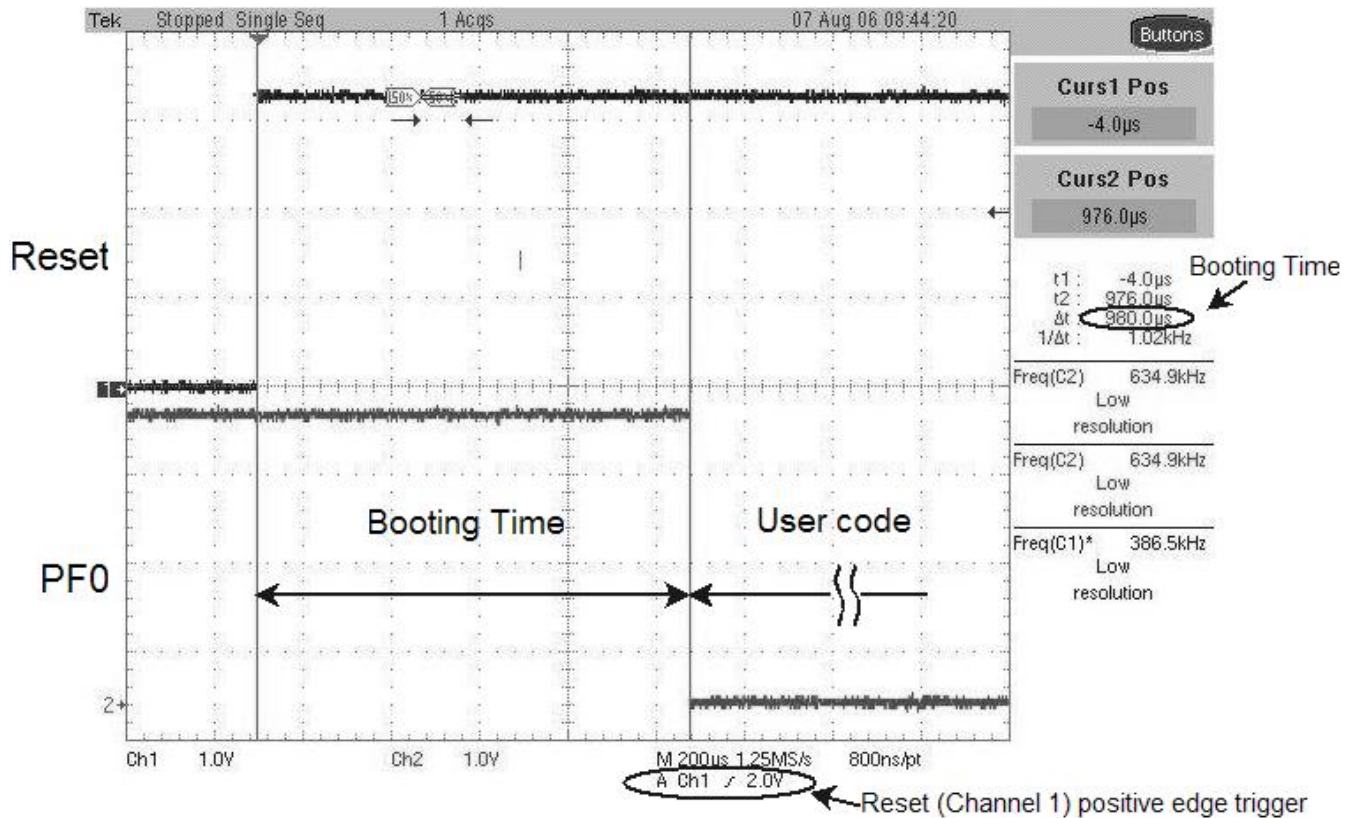


Figure 3. Oscilloscope Waveforms of /RESET and PF0 for Measuring Booting Time

Estimating Boot Time

It is possible to estimate the boot time of an ADSP-BF533 Blackfin processor, given the crystal frequency and the .ldr file. Among the four components of the boot time, boot ROM execution time (T_{ROM}) is relatively constant. Although it may vary with regard to loader file contents, the deviation is usually small enough to be ignored. The execution time of initialization code (T_{Init}) entirely depends on the specific application, and it can be estimated separately.

The data/code loading time can be estimated by Equation 1, where T_{SCLK} is the period of the system clock, $N_{Loading}$ is the number of bytes to be loaded, and P_{Mem} is the number of system clocks needed to read one byte from external memory, whether flash or SPI serial EEPROM.

$$T_{Loading} = T_{SCLK} \times N_{Loading} \times P_{Mem}$$

Equation 1. Estimation of Loading Time

The zero-filling time can be estimated by Equation 2, where T_{CCLK} is the period of the core clock, $N_{Filling}$ is the number of bytes to be zero-filled, and $P_{Filling}$ is the number of core clocks needed for zero-filling one byte.

$$T_{Filling} = T_{SCLK} \times N_{Filling} \times P_{Filling}$$

Equation 2. Estimation of Zero-Filling Time

The boot time ($T_{Booting}$) can be estimated by the sum of the above four times, as shown in Equation 3.

$$T_{Booting} = T_{ROM} + T_{Init} + T_{Loading} + T_{Filling}$$

Equation 3. Estimation of Boot Time

Table 1 summarizes the values of the needed parameters for ADSP-BF533 processors. The loading size ($N_{Loading}$) and zero-filling size ($N_{Filling}$) can be extracted by parsing .ldr files. A detailed description of loader file format can be

found in *ADSP-BF533 Blackfin Booting Process (EE-240)*. The core clock (T_{CLK}) and system clock (T_{SCLK}) can be obtained easily by the crystal clock frequency and the processor's default PLL settings, which can be found in the processor's *Hardware Reference*^[3].

	Flash	SPI Serial EEPROM
T_{ROM}	3360 core clocks	90000 core clocks
P_{Mem}	22	266
$P_{Filling}$	1	1
T_{Init}	Application Specific	Application Specific

Table 1. Estimation Parameter Summary

Optimization of Boot Time

From the above section, it can be seen that all the components of boot time are proportionally related to the processor's core/system clock, given the crystal clock and loader file. However, the default clock setting of the processor is designed to accommodate the slowest flash/SPI serial EEPROM devices. Furthermore, the core does not run at its maximum frequency by default. One solution to expedite the booting process is to add a short piece of initialization code which changes the processor's settings to make it run the boot process at the maximum speed. The changeable settings include:

- PLL's multiplier select $MSEL$ in the PLL_CTL register
- Clock divider selects ($CSEL$ and $SSEL$) in the PLL_DIV register
- SPI baud rate setting in the SPI_BAUD register
- Asynchronous memory control settings in the $EBIU_AMBCTLx$ registers

The booting process with initialization code is shown in Figure 4. Upon reset, the boot ROM first loads the initialization code at the default

speed and calls it. The default speed for a Blackfin processor is $10 * CLKIN$ for the core clock and $2 * CLKIN$ for the $SCLK$. After the return from initialization subroutine, the boot ROM continues the booting process for the remaining code/data in external memory at a higher speed with the changed settings. However, the improved booting time comes with an added latency — the loading and running time of the initialization code, which includes a PLL relock sequence. In addition, the changed settings must not exceed the maximum ratings of the processor and the flash/SPI serial EEPROM devices. The limitations are:

- Maximum core clock and system clock for Blackfin processors
- Minimum setup/access/hold times of flash memory
- Maximum baud rate for the SPI serial EEPROM

As a result, a constrained optimization problem must be solved in order to achieve the shortest boot time. The BTET, introduced in the following section, can be used to find the solution.

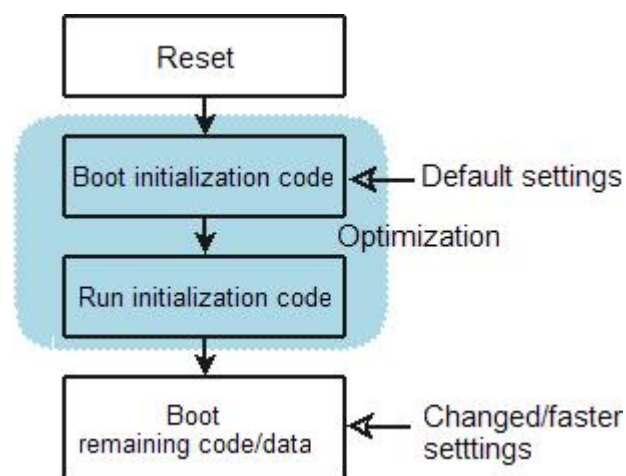


Figure 4. Booting Process with Initialization Code

Booting Time Estimation Tool (BTET)


The booting time estimation tool (`BTET.exe`) is a console-based software tool developed to parse loader files to estimate and optimize booting time. Taking a loader file as input, it can provide the default booting time and give recommendations on processor setting changes needed to achieve the shortest boot time. Both flash and SPI serial EEPROM boot modes are supported. The tool is fully controlled by a user-defined configuration file. The user can apply this tool to the application's loader file to estimate the time for the system to boot up, and then use the recommended settings to obtain a faster startup.

BTET Configuration File

As shown in [Appendix A – BTET Configuration File: `BTET.cfg`](#), the configuration file (`BTET.cfg`) consists of two parts — user-defined variables, and system-defined variables. Each group is divided into several sections. Users should change only the user-defined variables, according to the specific application, and leave the system-defined variables untouched. The variables are explained as follows.

- Mode section
 - FLASH — Determines whether to do optimization for flash
 - SPI — Determines whether to do optimization for SPI serial EEPROM. If neither is specified, only analyze the loader file.
- Crystal section
 - Tcrystal — Period of crystal clock (`CLKIN`) in μs
- SystemLimit section
 - MIN_CCLK — Minimum period of core clock in μs

- MIN_SCLK — Minimum period of system clock in μs
- FlashLimit section
 - FLASH_SETUP — Minimum setup time of flash in μs
 - FLASH_ACCESS — Minimum read/write access time of flash in μs
 - FLASH_HOLD — Minimum hold time of flash in μs
- SPIBaudLimit section
 - SPIBaudLimit — Maximum SPI baud in MHz

 The configuration file (`BTET.cfg`) must be in the same directory as `BTET.exe`.

Running BTET

BTET can be run in Console page of the Output window in two ways:

- It can take the loader file's name as its parameter in the command line, as shown in [Listing 2](#).

```
BTET my_loader_file.ldr
```

Listing 2. BTET Command-Line Example

- If the command line does not provide the loader file name, the BTET will prompt you to type it in, as shown in [Listing 3](#).

```
WARNING: loader file not specified,
type in its name
>>>
```

Listing 3. BTET Prompt

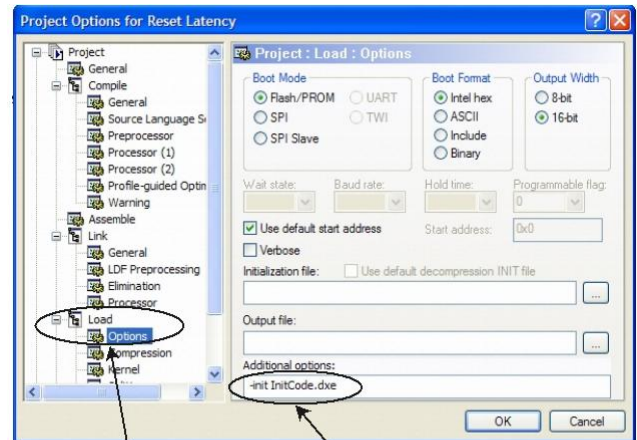
BTET Output

BTET outputs its analysis and optimization results to both the console and to the log file (`BTET.log`). The explanation of each part of the output is illustrated in [Appendix B – Output of BTET](#).

Adding Initialization Code

Sample initialization code accompanies this document in the associated .ZIP file^[4]. If the advice given by BTET is taken, the macros in the sample code can be changed to the recommended settings or customized initialization code can be used.

The initialization code must be compiled into a .dxe file. Then, in VisualDSP++, change the project's load options by adding `-init InitCode.dxe` in the Additional options field of the Project Options dialog box (Figure 5), which you open by choosing Project Options from the Project menu.



Under Project->Project Options..menu

Add initialization option,
InitCode.dxe is the name of the initialization code

Figure 5. Adding Init Code to Project

Appendix A – BTET Configuration File: BTET.cfg

```
# Configuration for Booting Time Estimation Tool
# Variables needing to be defined in configuration file
# if no configuration file is found, the hard-wired values are used
# USER DEFINED VARIABLES #
[Mode]
FLASH = 1           # do FLASH optimization
SPI = 1             # do SPI optimization
                   # if neither is specified, only analyze the loader file

[Crystal]
Tcrystal = 0.03    # in us, 1.0/33 us for 33MHz

[SystemLimit]
MIN_CCLK = 0.002   # in us, 1/500.0 us
MIN_SCLK = 0.011  # in us, 1/90.0 us

[FlashLimit]
FLASH_SETUP = 0.022      # in us, 22/1000.0 us
FLASH_ACCESS = 0.036     # in us, 36/1000.0 us
FLASH_HOLD = 0.011      # in us, 11/1000.0 us

[SPIBaudLimit]
SPIBaudLimit = 1.5      # in MHz

# SYSTEM DEFINED VARIABLES #
# ! DO NOT CHANGE      #
[InitCode]
PLLCount = 512
InitCodeSize = 202
InitCodeRun = 5076

[KernelNature]
FILL_PER_BYTE = 5
BOOT_FLASH_RUN = 3360
BOOT_SPI_RUN = 90000

[DefaultSystemSetting]
MSEL = 10
CSEL = 1
SSEL = 5

[DefaultFlashSetting]
FLASH_SETUP = 3
FLASH_ACCESS = 15
FLASH_HOLD = 4

[DefaultSPIBaudSetting]
SPIBaudSetting = 133;

[Console]
Quiet = 0
LogFileName = "BTET.log"
```


References

- [1] *ADSP-BF533 Blackfin Embedded Processor Data Sheet*. Rev C, May 2006. Analog Devices, Inc.
- [2] *ADSP-BF533 Blackfin Booting Process (EE-240)*. Rev 3, January 2005. Analog Devices, Inc.
- [3] *ADSP-BF533 Blackfin Processor Hardware Reference*. Rev 3.2, July 2006. Analog Devices, Inc.
- [4] *Associated .ZIP File (EE-308)*. Rev 1, November, 2006. Analog Devices, Inc.

Document History

Revision	Description
<i>Rev 1 – December 4, 2006 by Jiang Wu</i>	Initial release.