



Using a secondary bootloader on the EMSK

Overview

embARC is an open software platform designed to help accelerate the development and production of embedded systems based on DesignWare® ARC® processors.

This article provides instructions on how to use a secondary bootloader to auto-load a “boot.bin” file containing a desired application to memory from an SD card after power on or reset.

NOTE: This article assumes the reader is already familiar with embARC. If this is your first project with embARC, please start by first reading our “Quick start” article to ensure your development environment is properly setup before you begin.

Please visit <https://www.embarc.org/index.html> for more information on embARC.

Development Environment

The development environment used in this article is the following:

Development host operating system:

- **Windows 7**

Development Toolchain for Target Platform:

- **GNU Toolchain for DWC ARC Processors, version 2015.06**

Target platform:

- **ARC EM Starter Kit (EMSK), version 2.1**

NOTE:

Loading of secondary boot loader in SPI Flash is only supported on EMSK version 2.1 or higher.

Running an application in self-boot mode

The EMSK uses a Xilinx Spartan®-6 FPGA part, which can be configured to run different members of the ARC EM Processor family. The EMSK includes a SPI flash pre-programmed with four FPGA configurations of ARC EM cores. EMSK v2.0 and v2.1 support the ARC EM5D and EM7D Processors, which implement the ARCV2DSP Instruction Set Architecture (ISA).

When a “power on” or reset/configure is issued, the FPGA will auto-load one of the pre-installed FPGA configurations from SPI flash.

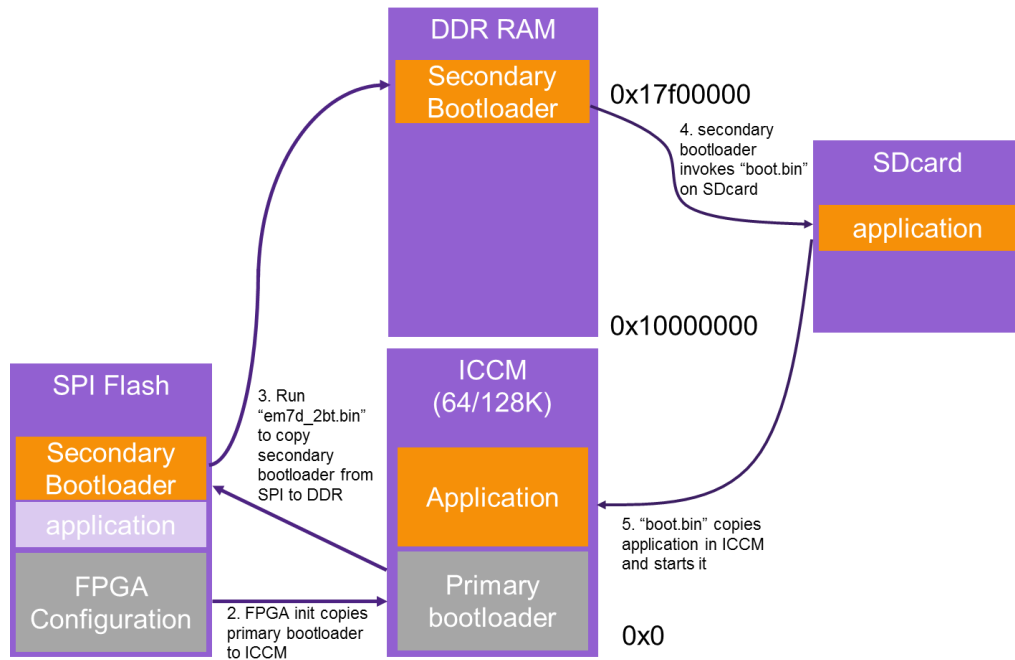
After the FPGA configuration is loaded from the SPI flash, a simple primary bootloader is loaded in ICCM. Through the primary bootloader, an application can be loaded from SPI Flash into ICCM or external DDR memory.

The function of secondary bootloader

Considering that the SPI Flash is used to store FPGA images, the secondary bootloader is designed based on the primary bootloader to load an application from an SD card since it can be read and written easily.

The startup sequence is listed below:

1. Power on or reset event
2. Load FPGA configuration from the SPI flash
3. Run primary bootloader, which loads the secondary bootloader from the SPI Flash into main memory (DRAM)
4. Run secondary bootloader from main memory to load application from the SD card into ICCM memory
5. Run the application from ICCM memory



Building and running secondary bootloader example

- 1) Program the secondary bootloader application into onboard SPI Flash:
 - a. Go to `\embARC\example\emsk\bootloader` in command line.
 - b. Enter `"make TOOLCHAIN=gnu BD_VER=21 CUR_CORE=arcem7d bin"`
 NOTE: The default settings of board (BOARD), board version (BD_VER) and core (CUR_CORE) for the project are "emsk", "11" and "arcem6", respectively so you need to explicitly specify these parameters on the command line for use with EMSK v2.1 and EM7D.
 Enter `"make cfg TOOLCHAIN=gnu BD_VER=21 CUR_CORE=arcem7d"` to show the configuration.
 Also make sure you have selected configuration 1 on the EMSK for arcem7d by setting dipswitch 1 to the ON position (all others OFF).

The image below shows output of successful binary file generation:

```

Compiling      : " ../../../../middleware/ntshell/cmds/cmd_spiw.c
"Compiling     : " ../../../../middleware/ntshell/cmds/cmd_bcr.c
"Compiling     : " ../../../../middleware/ntshell/cmds/cmds.c
"Compiling     : " ../../../../middleware/ntshell/cmds/cmd_ledflash.c
"Compiling     : " ../../../../middleware/ntshell/cmds/cmd_map.c
"Compiling     : " ../../../../middleware/ntshell/cmds/cmd_wifi.c
"Compiling     : " ../../../../middleware/ntshell/cmds/cmd_swape.c
"Compiling     : " ../../../../middleware/ntshell/cmds/cmd_load.c
"Compiling     : " ../../../../middleware/ntshell/port/ntshell_usrcmd.c
"Compiling     : " ../../../../middleware/ntshell/port/ntshell_task.c
"Archiving     : " obj_emsk_11/gnu_arcem6/libmidntshell.a
arc-elf32-ar: creating obj_emsk_11/gnu_arcem6/libmidntshell.a
"Creating Directory : " obj_emsk_11/gnu_arcem6/arc
"Compiling     : " ../../../../arc/arc_timer.c
"Compiling     : " ../../../../arc/arc_cache.c
"Compiling     : " ../../../../arc/arc_exception.c
"Assembling    : " ../../../../arc/arc_startup.s
"Assembling    : " ../../../../arc/arc_exc_asm.s
"Archiving     : " obj_emsk_11/gnu_arcem6/libarc.a
arc-elf32-ar: creating obj_emsk_11/gnu_arcem6/libarc.a
"Archiving     : " obj_emsk_11/gnu_arcem6/libembarc.a
arc-elf32-ar: creating obj_emsk_11/gnu_arcem6/libembarc.a
"Linking      : " obj_emsk_11/gnu_arcem6/emsk_bootloader_gnu_arcem6.elf
"Generating Binary obj_emsk_11/gnu_arcem6/emsk_bootloader_gnu_arcem6.bin"

```

- c. Insert the SD card to your PC, and copy the binary file **obj_emsk_21/gnu_arcem7d/emsk_bootloader_gnu_arcem7d.bin** to the SD Card root directory. Rename it to **"em7d_2bt.bin"** as the secondary bootloader.
- d. Insert the SD Card into the EMSK board SD Card slot.

Go to `\embARC\example\emsk\ntshell` in command line, Enter **"make TOOLCHAIN=gnu BD_VER=21 CUR_CORE=arcem7d"**. Once the application builds successfully, enter **"make run TOOLCHAIN=gnu BD_VER=21 CUR_CORE=arcem7d"** in command line to start ntshell. The response in the terminal window is shown as below.

```

"Archiving     : " obj_emsk_11/gnu_arcem6/libmidntshell.a
arc-elf32-ar: creating obj_emsk_11/gnu_arcem6/libmidntshell.a
"Creating Directory : " obj_emsk_11/gnu_arcem6/arc
"Compiling     : " ../../../../arc/arc_timer.c
"Compiling     : " ../../../../arc/arc_cache.c
"Compiling     : " ../../../../arc/arc_exception.c
"Assembling    : " ../../../../arc/arc_startup.s
"Assembling    : " ../../../../arc/arc_exc_asm.s
"Archiving     : " obj_emsk_11/gnu_arcem6/libarc.a
arc-elf32-ar: creating obj_emsk_11/gnu_arcem6/libarc.a
"Archiving     : " obj_emsk_11/gnu_arcem6/libembarc.a
arc-elf32-ar: creating obj_emsk_11/gnu_arcem6/libembarc.a
"Linking      : " obj_emsk_11/gnu_arcem6/emsk_ntshell_gnu_arcem6.elf

C:\Users\nnsong\workspace\embARC201505\embARC\example\emsk\ntshell>make run
"Download & Run obj_emsk_11/gnu_arcem6/emsk_ntshell_gnu_arcem6.elf"
arc-elf32-gdb -ex "target remote | openocd --pipe -s C:/arc_gnu/share/openocd/scripts -
pts/board/snps_em_sk_v1.cfg" -ex "load" -ex "c" obj_emsk_11/gnu_arcem6/emsk_ntshell_gnu
GNU gdb (ARCompact/ARCV2 ISA elf32 toolchain/2015.06) 7.9.1
Copyright (C) 2015 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software; you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"

```

- e. Use ntshell command "*spirw*" in the terminal window to write the *em7d_2bt.bin* to SPI flash:
 - i. run "*spirw -h*" to show help;
 - ii. run "*spirw -i*" to check SPI flash ID, it should be Device ID = ef4018;
 - iii. run "*spirw -w em7d_2bt.bin 0x17f00000 0x17f00004*" to program SPI flash;

Tips: 0x17f00000 is the starting address when the program is loaded into RAM.
0x017f00004 is the starting address of running program. They must be determined according to `\embARC\example\emsk\bootloader\emsk_bootload.ld`.
 - iv. Check the output message to see if it is programmed successfully.

Be patient, this may take some time.

```

COM4:115200baud - Tera Term VT
File Edit Setup Control Window Help
ntshell command ledsbtn(operate the LED through button and switch) was registered!
ntshell command load(load .bin file from SD card to ram at specified address) was registered!
ntshell command go(run the program at the specified address) was registered!
NTShell Task StartUp
COM1>spirw -h
usage: spirw <-opt>
--Option Available--
-h/-?          show help
-i read Device ID
-r filename      read image from spi to file on host, address and size have been taken from header in spi flash
-w file ram_addr start_addr  write binary [file] to spi flash, the image will be loaded to ram_addr and run from start_addr
*****
SPI Flash Command Done
COM1>spirw -i
Device ID = fffffff
*****
SPI Flash Command Done
COM1>spirw -i
Device ID = ef4018
*****
SPI Flash Command Done
COM1>spirw -w em6_2bt.bin 0x17f00000 0x17f00004
** Writing file em6_2bt.bin; to address 0x17f00000 start at 0x17f00004
   Erased 37 sectors from 0x780000 to 0x7a4fff

** Writing SPI flash **
- data
write 0x78001c - 0x78101c check_sum:56f2c
write 0x78101c - 0x78201c check_sum:abc34

```

```

write 0x7a201c - 0x7a301c check_sum:ef4699
write 0x7a301c - 0x7a401c check_sum:f31229
write 0x7a401c - 0x7a430c check_sum:f3de2d
  Written 148236 bytes: header=28 and image=148208 image from 0x780000 to 0x7a430b

** Boot image header **
head: 0x68656164
cpu: 0x0
start: 0x78001c
size: 0x242f0
ram_addr: 0x17f00000
start: 0x17f00004
checksum: 0xf3de2d
*****
SPI Flash Command Done

```

- 2) Generate *boot.bin* using embARC examples which RAM start address should be 0x10000000. The gpio project is selected as an example:

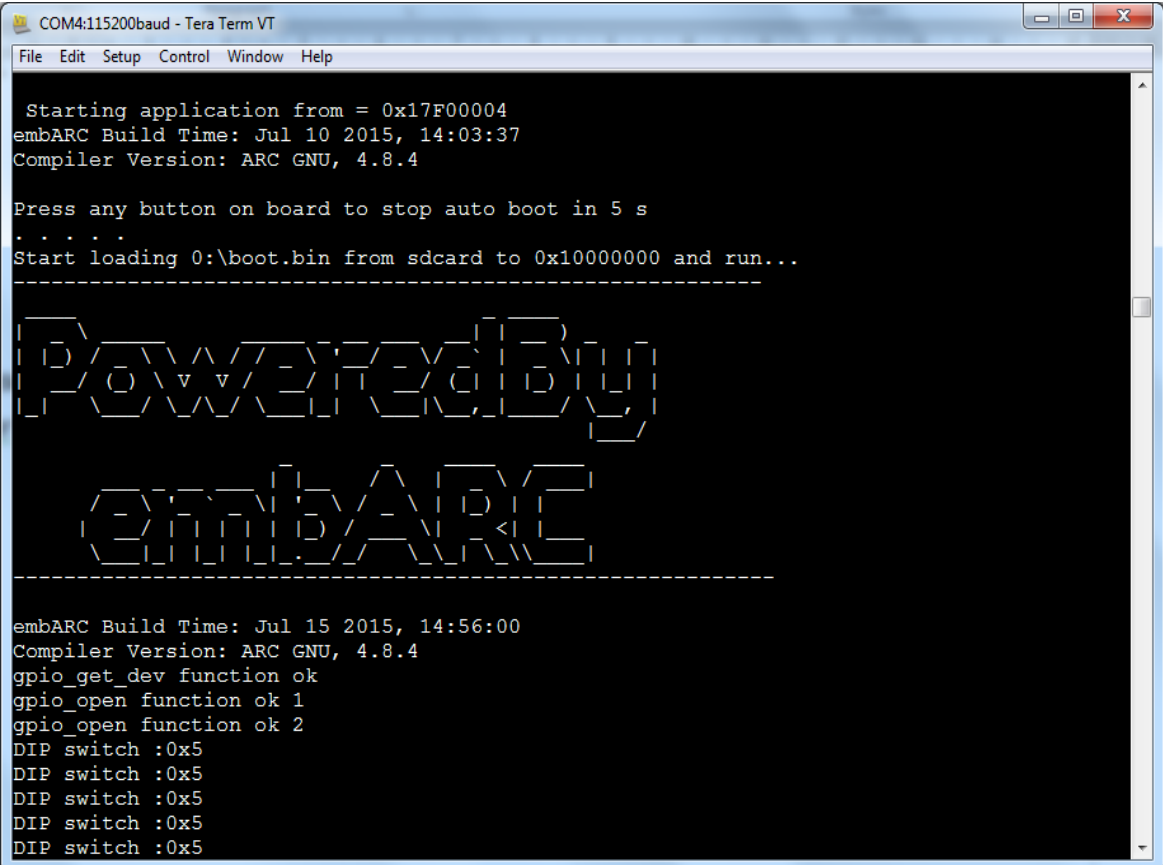
- a. Go to `\embARC\example\emsk\gpio` in command line;
- b. Enter “make TOOLCHAIN=gnu BD_VER=11 CUR_CORE=arcem6 bin”;
- c. Insert SDCard to PC. Copy generated binary file `obj_emsk_11/gnu_arcem6/emsk_gpio_gnu_arcem6.bin` to SD card root. And rename it to `boot.bin`;

Tips: The secondary bootloader can only identify `boot.bin` in the SD card root. So `boot.bin` should be set as the application name and be moved to SD card root.

- 3) Insert SD Card back to EMSK.

Before resetting the EMSK board, make sure Bit 4 of the onboard DIP switch is ON to enable secondary bootloader to run.

Press the configure button to reboot the EMSK. The following console output showing the GPIO application being loaded and run should be seen:



```
COM4:115200baud - Tera Term VT
File Edit Setup Control Window Help
Starting application from = 0x17F00004
embARC Build Time: Jul 10 2015, 14:03:37
Compiler Version: ARC GNU, 4.8.4

Press any button on board to stop auto boot in 5 s
. . . . .
Start loading 0:\boot.bin from sdcard to 0x10000000 and run...
-----
          Powered By
          embARC
-----

embARC Build Time: Jul 15 2015, 14:56:00
Compiler Version: ARC GNU, 4.8.4
gpio_get_dev function ok
gpio_open function ok 1
gpio_open function ok 2
DIP switch :0x5
DIP switch :0x5
DIP switch :0x5
DIP switch :0x5
DIP switch :0x5
```

For any additional support on embARC, please post a question on embARC Forums at <https://forums.embarc.org/>