

CSCE 4114/5114
Embedded Systems:
Board Flashing Instructions

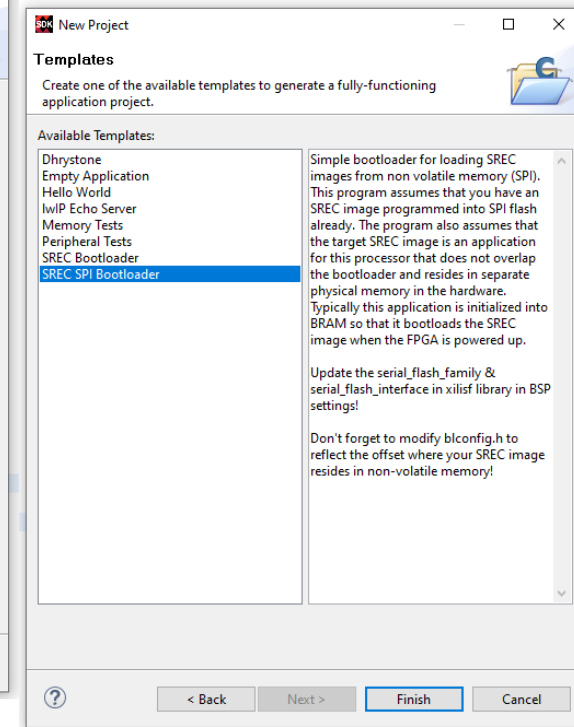
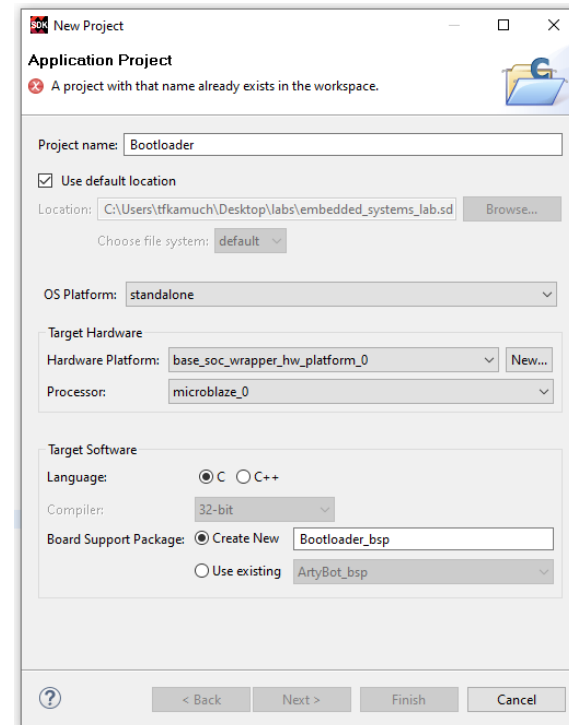
Tendayi Kamucheka

Introduction

- To get your application flashed on your board you will need two things:
 - A bootloader application to launch your application once it is stored on the flash memory
 - Your project application

Bootloader: Creating the Application

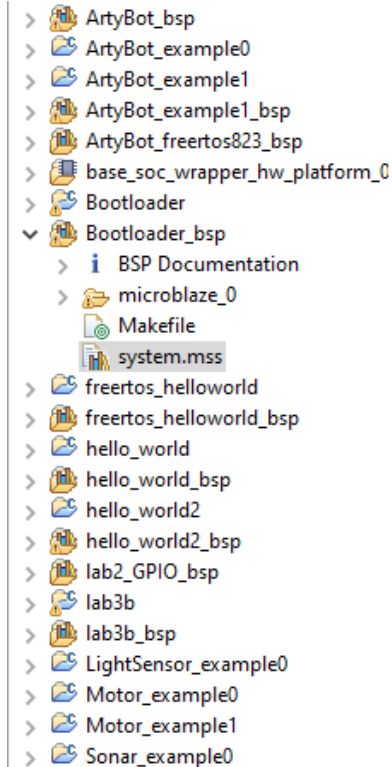
- Start off by creating a new "**Application Project**" in the SDK
- Name the project "*Bootloader*" or something along those lines
 - Make the OS Platform is "**standalone**"
 - For Board Support Package, you'll want to create a new package for the Bootloader.
- On the next page:
 - Select the "**SREC SPI Bootloader**" template
 - Then click on "**Finish**" to create the project



The examples repository has a bootloader ready to go!

Bootloader: Board Support Package

- We need to make a few changes to the board support package before the Bootloader is ready:
 - Namely, we want to add support for flash memory
 - And change the flash memory family to match what's on the board
- Under "**Bootloader_bsp**":
 - Open the "*system.mss*" file
 - Then select "Modify this BSP's Settings"



Bootloader_bsp Board Support Package

[Modify this BSP's Settings](#)

[Re-generate BSP Sources](#)

Target Information

This Board Support Package is compiled to run on the following hardware specification:
Hardware Specification: C:\Users\tfkamuch\Desktop\labs\example0
Target Processor: microblaze_0

Operating System

Board Support Package OS.

Name: standalone

Version: 6.1

Description: Standalone is a simple, low-level software

Documentation: [standalone v6.1](#)

Peripheral Drivers

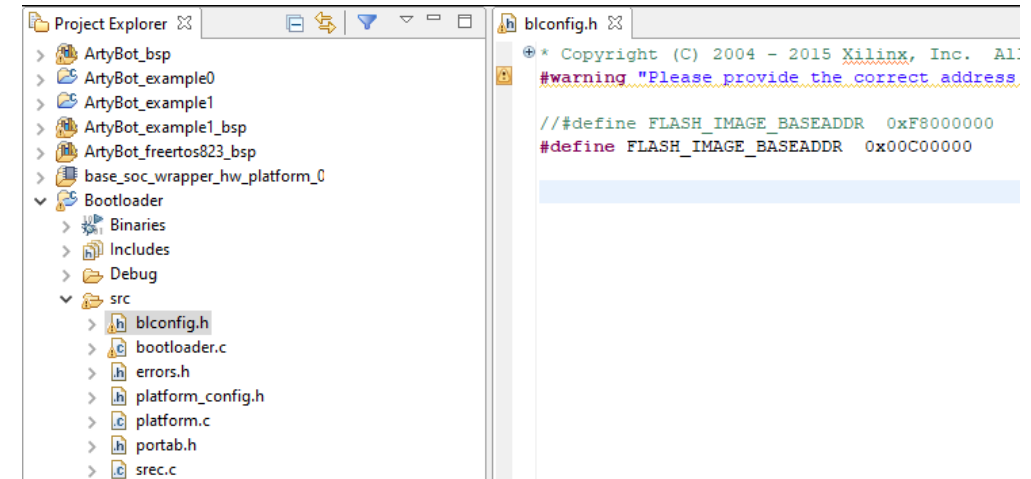
Drivers present in the Board Support Package.

Pmod_DHB1_0 Pmod_DHI
Pmod_Dual_MAXSONAR_0 Pmod_Dual_MAXSONAR_1
axi_ethernetlite_0 emaclite
axi_gpio_0 gpio
axi_gpio_1 gpio

The examples repository has a bootloader ready to go!

Bootloader: Memory Configuration

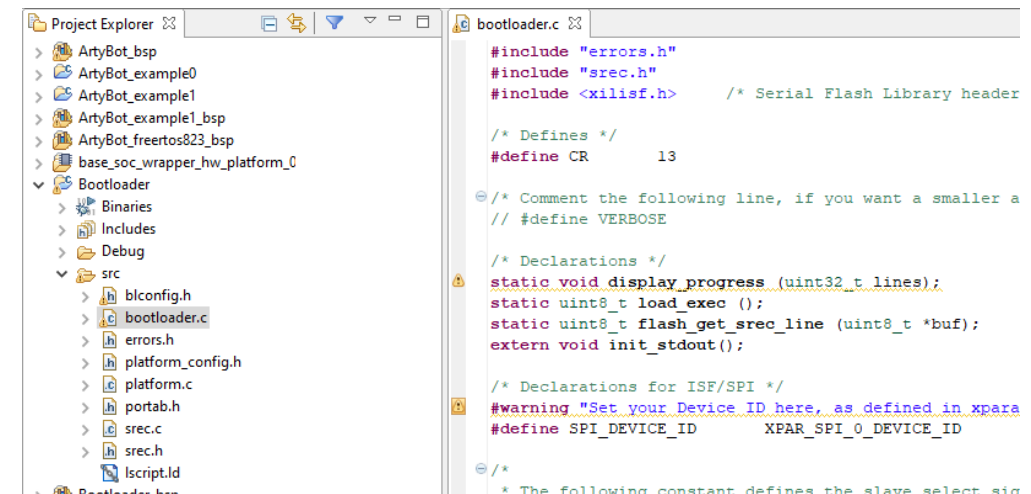
- In the "Bootloader" project, open the "*blconfig.h*" file.
- Next, modify the "**FLASH_IMAGE_BASEADDR**" value to "**0x00C00000**".
- This will be the starting address of your project in the flash memory
- Optionally, if you want to see debug info in Putty:
 - Uncomment the "**#define VERBOSE**" line in "*bootloader.c*"
- The Bootloader is ready!



The screenshot shows the Project Explorer on the left with the 'Bootloader' project selected. The 'src' folder is expanded, showing 'blconfig.h' as the active file. The editor on the right displays the content of 'blconfig.h', which includes a copyright notice for Xilinx, Inc. (2004-2015), a warning to provide the correct address, and two definitions for 'FLASH_IMAGE_BASEADDR'. The first is commented out as '0xF8000000', and the second is active as '0x00C00000'.

```
* Copyright (C) 2004 - 2015 Xilinx, Inc. All rights reserved.
#warning "Please provide the correct address"

// #define FLASH_IMAGE_BASEADDR 0xF8000000
#define FLASH_IMAGE_BASEADDR 0x00C00000
```



The screenshot shows the Project Explorer on the left with the 'Bootloader' project selected. The 'src' folder is expanded, showing 'bootloader.c' as the active file. The editor on the right displays the content of 'bootloader.c', which includes headers for 'errors.h', 'srec.h', and 'xil_printf.h'. It defines 'CR' as 13 and has a commented-out line for '#define VERBOSE'. It also contains declarations for 'display_progress', 'load_exec', 'flash_get_srec_line', and 'init_stdout'. A warning is present to set the device ID, which is defined as 'XPAR_SPI_0_DEVICE_ID'. A comment at the bottom indicates that the following constant defines the slave select signal.

```
#include "errors.h"
#include "srec.h"
#include <xil_printf.h> /* Serial Flash Library header

/* Defines */
#define CR 13

/* Comment the following line, if you want a smaller a
// #define VERBOSE

/* Declarations */
static void display_progress (uint32_t lines);
static uint8_t load_exec ();
static uint8_t flash_get_srec_line (uint8_t *buf);
extern void init_stdout();

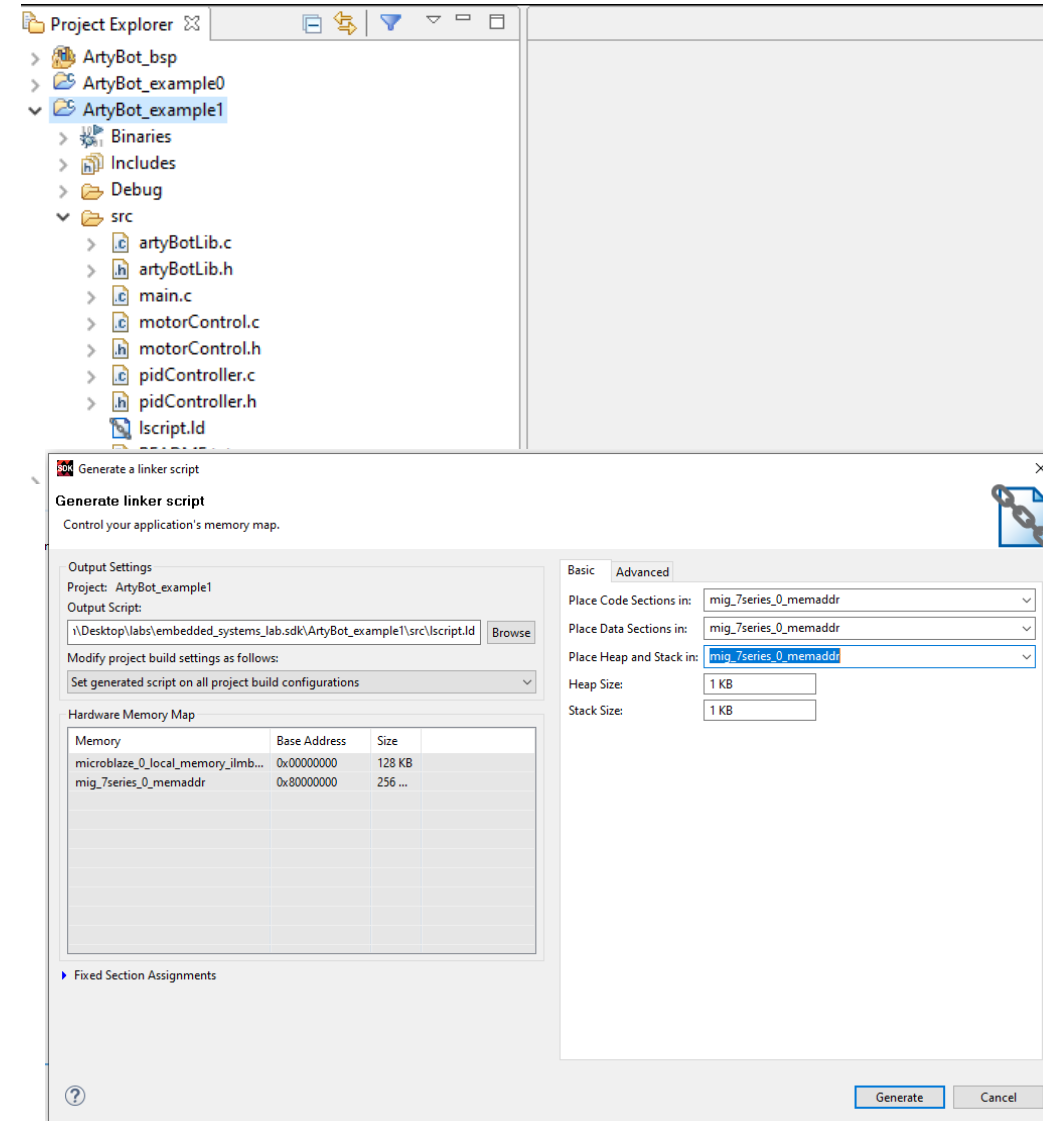
/* Declarations for ISF/SPI */
#warning "Set your Device ID here, as defined in xpara
#define SPI_DEVICE_ID XPAR_SPI_0_DEVICE_ID

/*
 * The following constant defines the slave select signal
```

The examples repository has a bootloader ready to go!

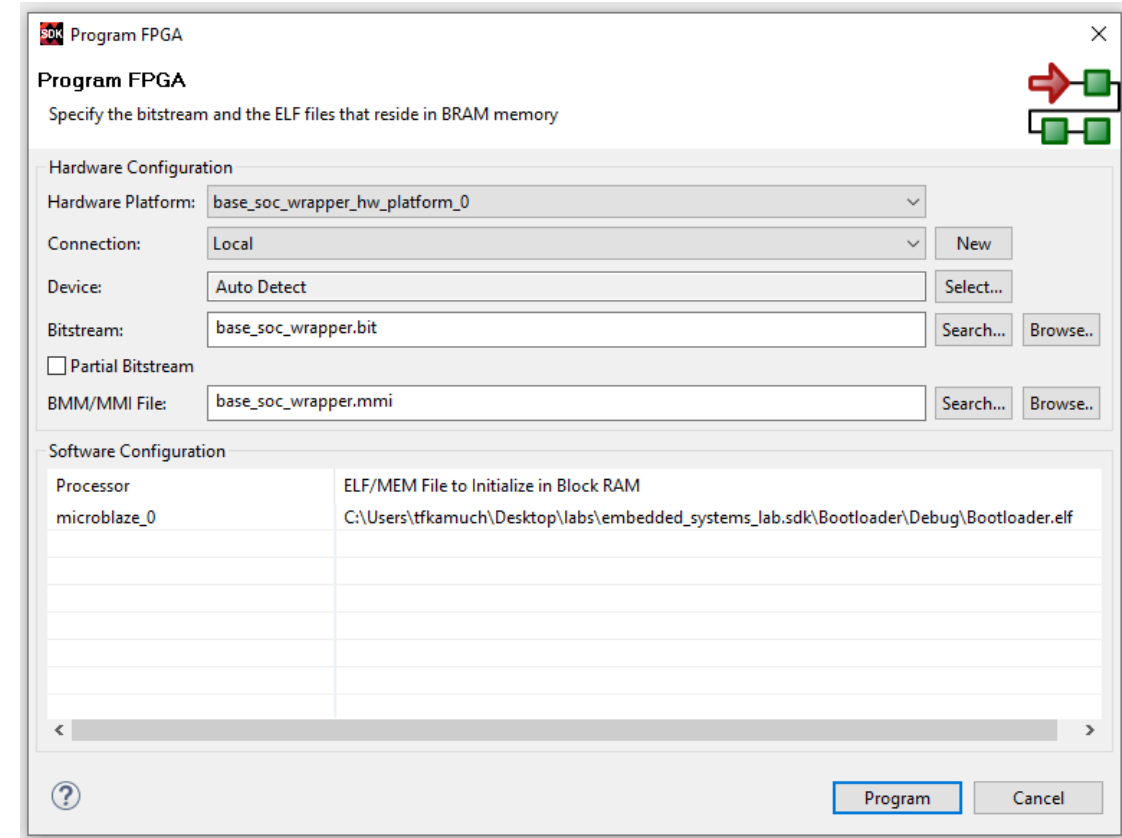
Creating Your Project: Linker Script

- We want to make sure our compiled application are run from the DDR3 ram, so we need to change the linker script
- Start by right-clicking your project's folder and selecting "**Generate Linker Script**"
- On the right side of the generate window, change all the memory locations to "*mig_7series_0_memaddr*"
- Now, move on to developing your amazing application



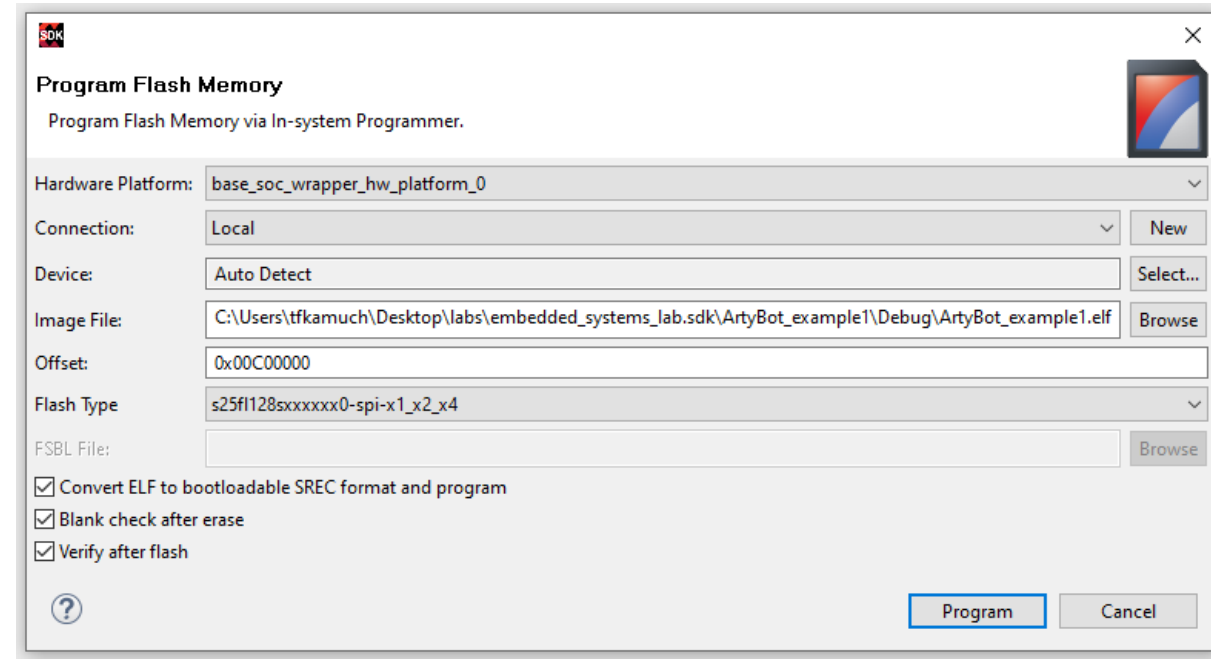
SPI Flashing: Generating Bitstream

- First, we generate a bitstream that we'll later use to flash the SPI memory.
- Open the "**Program FPGA**" utility
- Then change the "**ELF/MEM File to Initialize Block RAM**" option to "*Bootloader.elf*"
- Refer to example image for path to elf file
- Now, click "**Program**"



SPI Flashing: Your project

- Now let's get your project on the board
- Open the "**Program Flash Memory**" utility
- Set the "**Image File**" to your project's compiled elf file
- Set the "**Offset**" to "**0x00C00000**"
- For flash type, refer to example image
- Check all the checkboxes in the bottom section
- Now, click "**Program**" to begin flashing the board



SPI Flashing: Bootloader

- Almost there! Now we need to flash Bootloader
- Open the "**Program Flash Memory**" utility again
- This time, change "**Image File**" to "*base_soc_wrapper_hw_platform_0\download.bit*"
- Set the "**Offset**" to "**0x0**"
- Finally, click "**Program**"
- We're done! Your bot should be ready to drive solo.

