C5535 Boot-Image Programmer

**Contents**

 [hide]

* [1Bootloader Features](https://processors.wiki.ti.com/index.php/C5535_Boot-Image_Programmer#Bootloader_Features)
* [2How to program a boot image onto C5535 eZdsp](https://processors.wiki.ti.com/index.php/C5535_Boot-Image_Programmer#How_to_program_a_boot_image_onto_C5535_eZdsp)
* [3Common Mistakes in Boot Image User Code](https://processors.wiki.ti.com/index.php/C5535_Boot-Image_Programmer#Common_Mistakes_in_Boot_Image_User_Code)
* [4Software Tools Checklist](https://processors.wiki.ti.com/index.php/C5535_Boot-Image_Programmer#Software_Tools_Checklist)

Bootloader Features[[edit](https://processors.wiki.ti.com/index.php?title=C5535_Boot-Image_Programmer&action=edit&section=1)]

The bootloader is a program that runs from ROM at power-on and loads a boot-image from a peripheral to the internal RAM of the C5535 DSP. It is always invoked after power-on-reset. Once the transfer is completed, the bootloader switches control to the user code so that it is automatically executed.

In order to ensure that the code cannot be accessed and read outside of the system, the code residing externally may be encrypted. The bootloader is responsible for bootloading the code from an external device (I2C EEPROM, SPI EEPROM, SPI serial flash, MMC/SD, USB, and UART), decrypting it if necessary, and writing it into DSP memory (on-chip or off-chip).

The C5535 has a fixed order in which it checks for a valid boot-image on each supported boot device.

The device order is 16-bit SPI EEPROM, 24-bit SPI serial flash, I2C EEPROM, and SD/SDHC/eMMC/moviNAND. The first device with a valid boot image will be used to load and execute user code. If none of these devices has a valid boot-image, the bootloader goes into an endless loop checking for data received on the UART or USB. If a valid boot image is received, it is used to load and execute user code. If no valid boot-image is received, the bootloader continues to monitor these two devices.

The C5535 eZdsp only contains a SPI serial Flash memory and a microSD card slot, which support both encrypted and non-encrypted boot images.

How to program a boot image onto C5535 eZdsp[[edit](https://processors.wiki.ti.com/index.php?title=C5535_Boot-Image_Programmer&action=edit&section=2)]

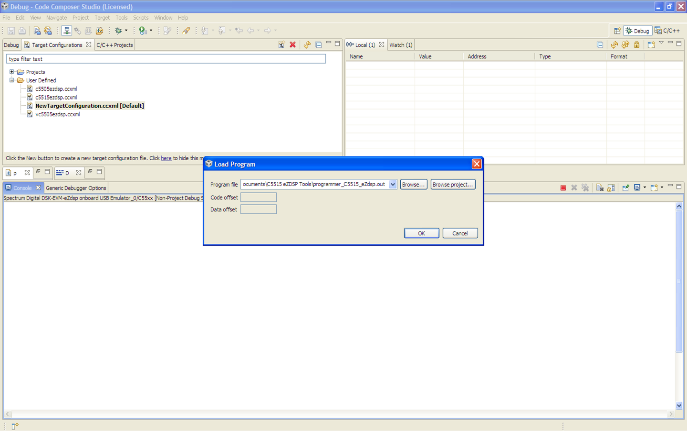
[](https://processors.wiki.ti.com/index.php/File:Figure_5.png)

Figure 5

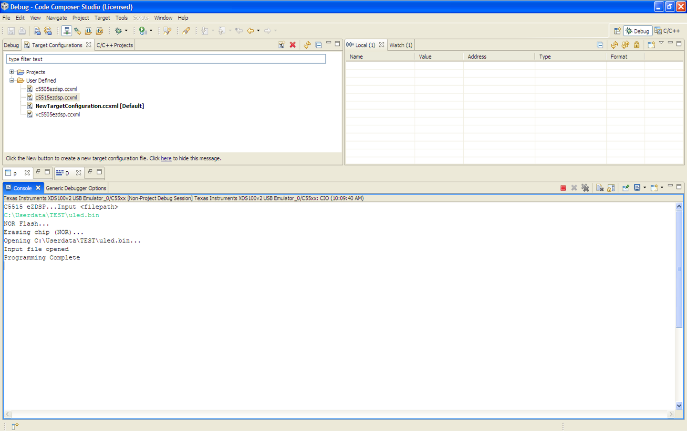
[](https://processors.wiki.ti.com/index.php/File:Figure_6.png)

Figure 6

* For information on how to create a boot-image for your device, [click here](https://processors.wiki.ti.com/index.php?title=C5000_Creating_Boot-Images&action=edit&redlink=1)

Once a boot image (\*.bin) is generated, customers can burn the boot image into the SPI Flash memory of the C5535 eZdsp through a utility called programmer that runs on C5535 using an emulator with [Code Composer Studio™](http://focus.ti.com/docs/toolsw/folders/print/ccstudio.html) software.

First, load the programmer\_c5535\_eZdsp.out from [File:Programmer C5535 eZdsp.zip](https://processors.wiki.ti.com/index.php/File:Programmer_C5535_eZdsp.zip) using the Code Composer Studio as shown in Figure 5 by selecting “Load Program…” under the Target menu in CCS. You can also download the programmer from the Spectrum Digital's [C5535 eZdsp Support Page](http://support.spectrumdigital.com/boards/ezdsp5535).

Next, run the program, when the user will be asked to input the filepath and filename for the boot image:

C5535 eZDSP...Input <filepath>

Figure 6 show how to program lednonencrypted.bin to the C5535 eZdsp using programmer\_c5535\_eZdsp.out.

NOTE: There cannot be any white spaces in the file directory path or the program will not run correctly. Programmer will not be able to find the input file as it considers the white space the end of the input.

Common Mistakes in Boot Image User Code[[edit](https://processors.wiki.ti.com/index.php?title=C5535_Boot-Image_Programmer&action=edit&section=3)]

A frequent issue is that the user code works fine in CCS environment but fails to work after converted to boot image. This is most likely because the user code relies on a GEL file in the CCS environment to initialize and setup PLL. This is easily corrected by not using GEL in CCS environment, but instead including all the necessary initialization steps in the user code. It is a good practice to re-initialize everything you need from within your bootloaded user code.

Another common mistake is allocating program code to SARAM31 memory. The bootloader writes to SARAM31 (byte address 0x4E000 – 0x4FFFF) thus any user code residing in SARAM31 will be corrupted. Do not use SARAM31 if you intend to convert this code to a boot image. Once the bootloader has finished loading the program into RAM, SARAM31 can be used.

Software Tools Checklist[[edit](https://processors.wiki.ti.com/index.php?title=C5535_Boot-Image_Programmer&action=edit&section=4)]

These are the required tools to test boot from flash or EEPROM:

1. hex55.exe to convert .out COFF file to binary.
2. programmer.out to download to flash and EEPROM.
3. CCS to run programmer.out.
4. cleaner.bin file to erase previous boot images from peripherals.