

# FSBL: Introduction Demo Script

## Introduction

Here you will have the opportunity to build a default configuration of a First Stage Boot Loader (FSBL) and explore the basic components of the software.

## FSBL: Introduction

Action with Description	Point of Emphasis and Key Takeaway
<ul style="list-style-type: none"><li>Launch the Vitis™ Unified IDE.</li><li>Close the Welcome tab if it is open.</li><li>Click <b>Open Workspace</b> and select a directory of your choice. Ensure that the directory is empty so that its contents do not interfere with the workspace.</li><li>Select <b>File &gt; New Component &gt; Platform</b> to create a new platform component.</li><li>Set the following options:<ul style="list-style-type: none"><li>Platform name: <b>UED_plat</b></li><li>Click <b>Next</b> and select <b>Hardware Design</b></li><li>XSA file: <b>\$TRAINING_PATH/CustEdIP/UED_zcu104.xsa</b></li><li>OS platform: <b>standalone</b></li><li>Processor: <b>psu_cortexa53_0</b></li><li>Select the <b>Generate boot artifacts</b> option</li></ul></li><li>Click <b>Finish</b>.</li></ul> <p>This will take a minute or so to complete.</p>	<ul style="list-style-type: none"><li>Create a platform component containing the FSBL and PMU firmware.</li></ul>

Action with Description	Point of Emphasis and Key Takeaway
<p>Explore the generated software.</p> <ul style="list-style-type: none"> <li>Expand <b>UED_plat &gt; Sources &gt; zynqmp_fsbl</b>.</li> <li>Open <b>xfsbl_main.c</b> and <b>xfsbl_initialization.c</b>.</li> </ul>	<ul style="list-style-type: none"> <li>The thrust of this exercise is to: <ul style="list-style-type: none"> <li>Determine the reset reason; initialize the APU; determine that the WDT did not create an interrupt during the previous boot attempt.</li> <li>Determine the boot mode according to the boot device and prepare that device for booting.</li> <li>Cycle through the partitions on the boot device and copy data to the indicated device.</li> <li>Pass control to the application or subsequent boot loader.</li> </ul> </li> </ul>
<ul style="list-style-type: none"> <li>Select <b>xfsbl_initialization.c</b>.</li> <li>Advance to (302)  <pre>if(FsblInstancePtr-&gt;ResetReason == XFSBL_PS_ONLY_RESET)</pre> </li> </ul>	<ul style="list-style-type: none"> <li>Determine the reset reason; initialize the APU; determine that the WDT did not create an interrupt during the previous boot attempt.</li> </ul>
<ul style="list-style-type: none"> <li>Advance to (873)  <pre>FsblInstancePtr-&gt;PrimaryBootDevice = BootMode;</pre> </li> </ul>	<ul style="list-style-type: none"> <li>Determine the boot mode according to the boot device and prepare that device for booting.</li> </ul>
<ul style="list-style-type: none"> <li>Select <b>xfsbl_main.c</b>.</li> <li>Advance to (196) <pre>FsblStatus = XFsbl_PartitionLoad(&amp;FsblInstance, PartitionNum);</pre></li> </ul>	<ul style="list-style-type: none"> <li>Cycle through the partitions on the boot device and copy data to the indicated device.</li> </ul>
<ul style="list-style-type: none"> <li>Advance to (254)  <pre>XFsb1_Printf(DEBUG_INFO, "=====  ===== In Stage 4 =====  \n\r");</pre> </li> </ul>	<ul style="list-style-type: none"> <li>Pass control to the application or subsequent boot loader.</li> </ul>

Action with Description	Point of Emphasis and Key Takeaway
<ul style="list-style-type: none"><li>• Open <b>xfsbl_debug.h</b>.</li><li>• Notice the debug definitions.</li></ul>	<ul style="list-style-type: none"><li>• Question: What symbol must be defined to start getting debug messages?</li><li>• Answer:<ul style="list-style-type: none"><li>• <code>DEBUG_PRINT_ALWAYS</code> or</li><li>• <code>DEBUG_GENERAL</code> or</li><li>• <code>DEBUG_INFO</code> or</li><li>• <code>DEBUG_DETAILED</code>.</li></ul></li><li>• Question: Which one do you think provides more information?</li><li>• Answer: <code>DEBUG_DETAILED</code>. Look at line 64.</li></ul>
<ul style="list-style-type: none"><li>• Close the Vitis Unified IDE.</li></ul>	<ul style="list-style-type: none"><li>• Enables a clean exit.</li></ul>

## Summary

Here you saw how the FSBL can be built for the specified hardware, thus supporting the Zynq™ 7000 SoC's A9 processor, the Zynq UltraScale+™ MPSoC's R5 or A53 processor, or even a MicroBlaze™ processor (in a device without the PS).

The FSBL software has a number of hooks that can be enabled to change how much text is displayed during its run as well as places to insert specific user boot code.