Instructions for loading SINGR project onto a uZ board

1. Download a new version of the SINGR DevKit from GitHub:
   1. <https://github.com/RMDInc/SINGR>
   2. Click the green “Clone or Download” button on the right side of the page to get a new version of the project.
   3. Define the “workspace” to keep the project in and move this new version of the DevKit files there.
2. Start SDK at the location of the “workspace”.
3. Import files:
   1. File > Import > General folder > Existing projects into workspace
   2. Click next, then browse to the workspace you defined
   3. Select root directory ->Browse to the folder where the DevKit files are located->Ok
      1. The instruction sets should populate under projects.
      2. Make sure they are selected.
   4. Select “Copy projects into workspace”
   5. Select “Add project to working sets”
   6. Click Finish
   7. SDK will briefly build the workspace.
4. Close the “Welcome” tab and the “Project Explorer” should be visible.
5. In the Project Explorer, right click on Software (ex: Echo3)
   1. Select Generate linker script > generate
6. Change the hardware specification file:
   1. Right click on the originalTrigger\_small folder > change hardware specification > click yes
   2. Click browse and navigate to the “system.hdf” file in the originalTrigger\_small folder > ok
7. Regenerate the “standalone\_bsp\_2” folder and correct sd card library settings
   1. Right click on the “standalone\_bsp\_2” folder > Regenerate BSP Sources
   2. When SDK is finished working, in the Project Explorer, click the dropdown to see the contents of standalone\_bsp\_2.
   3. We want to check/change a few settings in the xilffs\_v3\_1 library. Open dropdown boxes to get there:
      1. Standalone\_bsp\_2 -> ps7\_cortexa9\_0 -> libsrc -> xilffs\_v3\_1 -> src -> include
      2. Look for the file ffconf.h and double click to open it.
   4. We need to make sure the following settings are correct:
      1. \_FS\_MINIMIZE 0
      2. \_USE\_LFN 1
   5. After those settings are confirmed, save the project and SDK should rebuild the workspace.

NOTE: if the BSP is regenerated after this point, it will return to default and these settings will need to be changed again.

1. Right click on the “Echo3” project > Change referenced BSP
   1. Choose the new bsp that we created > Click ok
2. Make the First Stage Bootloader
   1. File > New > Application Project
   2. Type In Project Name (example: MZ\_FSBL)
   3. For the “Board Support Package” select the “Use existing” option which should show standalone\_bsp\_2.
   4. Click Next
   5. Select Zynq FSBL from the list.
   6. Click Finish
   7. SDK will build the first stage bootloader into the project.
3. If the project is going to be run in debug mode follow the next instructions, otherwise go to step 12 for how to build the project in Release mode and program the flash.
4. Set FSBL and Echo3 for debug
   1. Right Click on MZ\_FSBL and click Select Build Configurations > Set Active > Debug
   2. Right Click on Echo3 and click Select Build Configurations -> Set Active -> Debug
   3. Project > Build All
   4. Xilinx Tools > Program FPGA
   5. For Hardware Platform choose originalTrigger\_small
   6. For the bitstream textbox, click Search… and choose design\_1\_wrapper.bit
      1. Click Program. The Blue LED on the microZed board will turn off while the FPGA is being programmed and turn back on when it is finished.
   7. Run > Debug Configurations…
   8. There will be a list of debuggers, from the list choose Xilinx C/C++ application (GDB)
   9. Click the “New” button to create a configuration of that type
   10. Ensure that the hardware platform shown is the correct one (originalTrigger\_small)
   11. For Bitstream File, click Search… and choose design\_1\_wrapper.bit
   12. For Initialization file click Search… and choose the ps7\_init.tcl file.
   13. There is a dropdown box at the bottom of the window, choose Reset Processor
       1. Uncheck Program FPGA
       2. Check Run ps7\_init
       3. Check Run ps7\_post\_config
       4. Uncheck Enable Cross-Triggering
   14. Click the Application tab
   15. Click browse next to Project Name and select Echo3
   16. For Application, click Search… and choose Echo3.elf under Binaries
   17. Click Apply > Debug
   18. SDK will ask you if you want to switch perspectives, choose yes
   19. Along the toolbar at the top of the screen there will be buttons related to debugging that will illuminate (Resume, Pause, Stop, Step Over/Into, etc)
   20. If you are interfacing with the system using a terminal program, open it now and select the port that the microZed is connected to.
   21. When the system has loaded, click Resume (the play button) to start
   22. If you are using the GUI to interface with the system, open and run it now.
       1. If the GUI was already running when the microZed was started, go to the toolbar and select COM Port -> Restart COM to establish a connection with the board.
5. Set FSBL and Software for Release
   1. Right Click on MZ\_FSBL and click Select Build Configurations > Set Active > Release
   2. Right Click on Echo3 and click Select Build Configurations > Set Active > Release
   3. Project -> Build All
6. Create the BIF that the system will boot from.
   1. Xilinx Tools -> Create Zynq Boot Image
      1. Select Create new BIF file
      2. SDK should populate the box with a path to …\Echo3\bootimage\Echo3.bif
         1. If SDK does not, then click browse and navigate to:
         2. [workspace]\DevKit\Echo3\
         3. Create a new folder called “bootimage” and select this folder. Choose a file name (Echo3.bif) and click save.
      3. Under Boot image partitions, we need a set of three files in our BIF. These files need to be chosen in a specific order for the BIF to be successful. The files are:
         1. (bootloader) MZ\_FSBL.elf
         2. Design\_1\_wrapper.bit
         3. Echo3.elf
      4. To choose them, use the buttons on the right. If there are no files in the list, click add and browse to [workspace]\DevKit\MZ\_FSBL\Release\MZ\_FSBL.elf
         1. Before clicking Ok to select this file, ensure that the Partition Type is “bootloader”.
      5. Next, we must add or edit the second file which is found at the following location: [workspace]\DevKit\originalTrigger\_small\design\_1\_wrapper.bit
      6. Finally, we must add the last file: [workspace]\DevKit\Echo3\Release\Echo3.elf
      7. Once we have all three files in the list, we need to modify the output path to match the bootimage folder in Echo3.
      8. Then change the file being produced from …\BOOT.bin -> …\BOOT.mcs
      9. Click Create Image and SDK will create our BIF
7. Program the Flash Memory
   1. Xilinx Tools -> Program Flash
   2. The Hardware Platform should be originalTrigger\_small
   3. For Image File: browse to the BOOT.mcs that we just create in the previous step.
   4. Flash Type should be qspi single
   5. Check the box for “Blank check after erase”
   6. Check the box for “Verify after erase”
   7. Power on the microZed if it is not already on.
   8. Click program and wait. This step usually takes a few minutes.
      1. SDK will display what it is doing in the “Console” tab at the bottom of the screen. You can follow the text updates it is giving to double check that it is performing the correct processes and when it completes, that it did so successfully.
   9. After successfully programming the flash memory, the red LED labelled “USR” should be illuminated. Power cycle the board.
   10. Now only the blue LED and the green LED should be illuminated. The board has booted successfully and may be interfaced with.