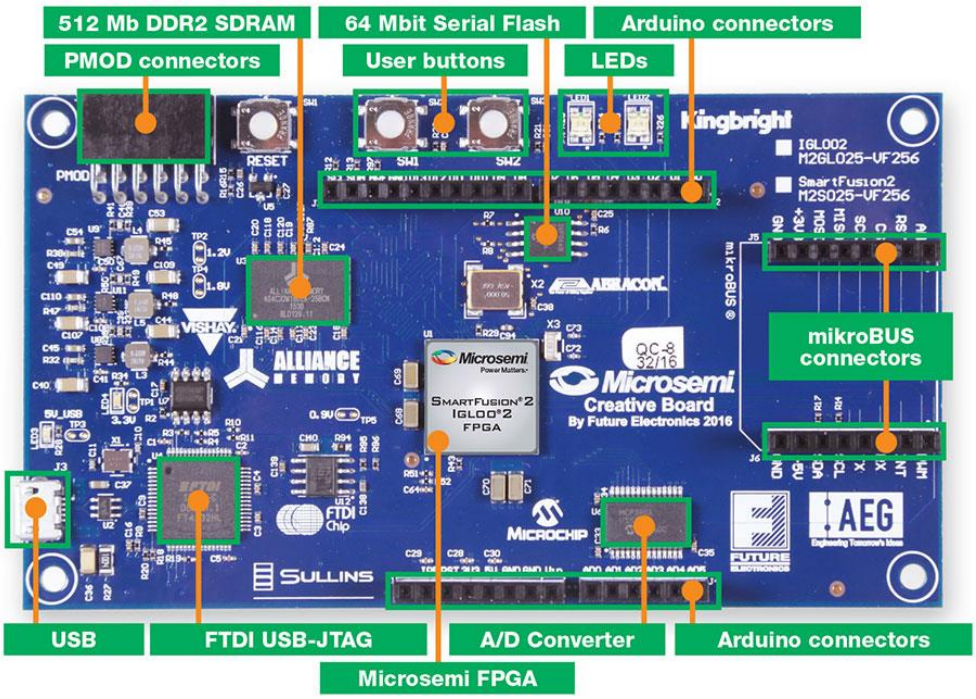
UC Irvine Microsemi Innovation Lab Guide on RISC-V SmartFusion2 Supplementary Implementation Guide

Abstract:
The following document pertains to the new RISC-V implementation that is released by Microsemi. A Tic-Tac-Toe game demo is provided as an example by Microsemi. This guide serves as a visual primer on loading the demo to augment provide Microsemi documentation. This guide is focused toward beginners seeking to get started with Microsemi RISC-V implementations.

Introduction:
This guide discusses how to load the Microsemi Provided Tic- Tac-Toe demo program provided by Microsemi. It discusses the three means for loading the demonstration. The implementation was tested on a Future Electronics Creative board.

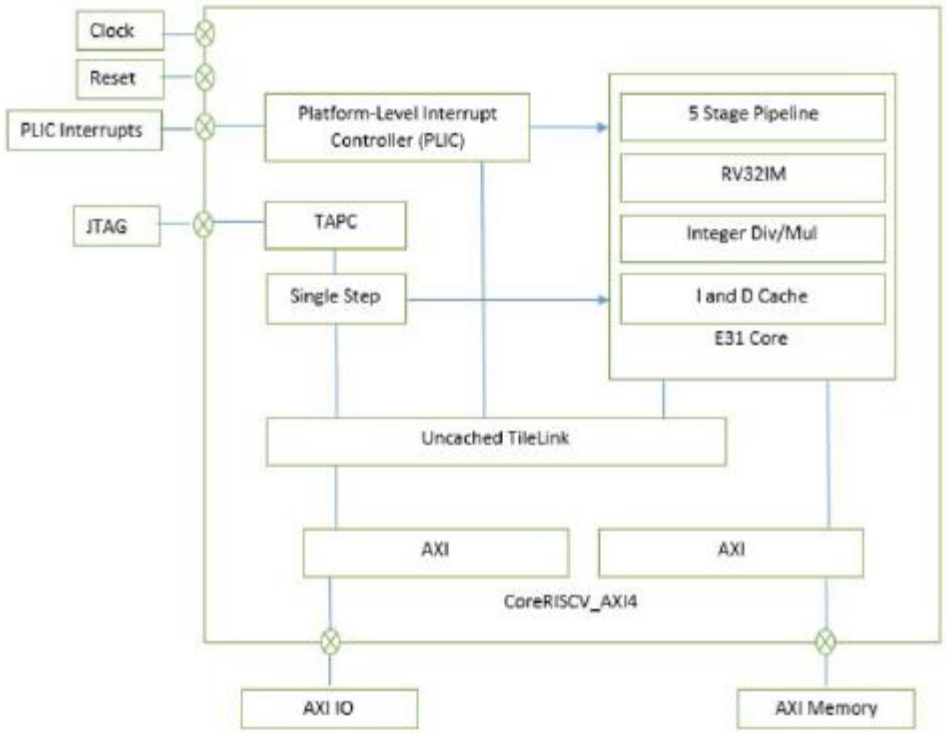
- Design Target Silicon: SmartFusion2 (M2S025-VF256)
- Design Target Board: Future Electronics Creative Board – SmartFusion2 (P/N: FUTUREM2SF-EVB)
- Design Required Software Development Toolchain:
    - Libero SoC 11.8 (minimum "Evaluation" or "Silver" license)
    - SoftConsole 5.1 (required for RISC-V) Remember: Project files are different for SoftConsole 5.2, so for a forward-facing project, consider starting to develop with SoftConsole 5.2.
    - Modelsim ME 10.5 (for waveform simulation)



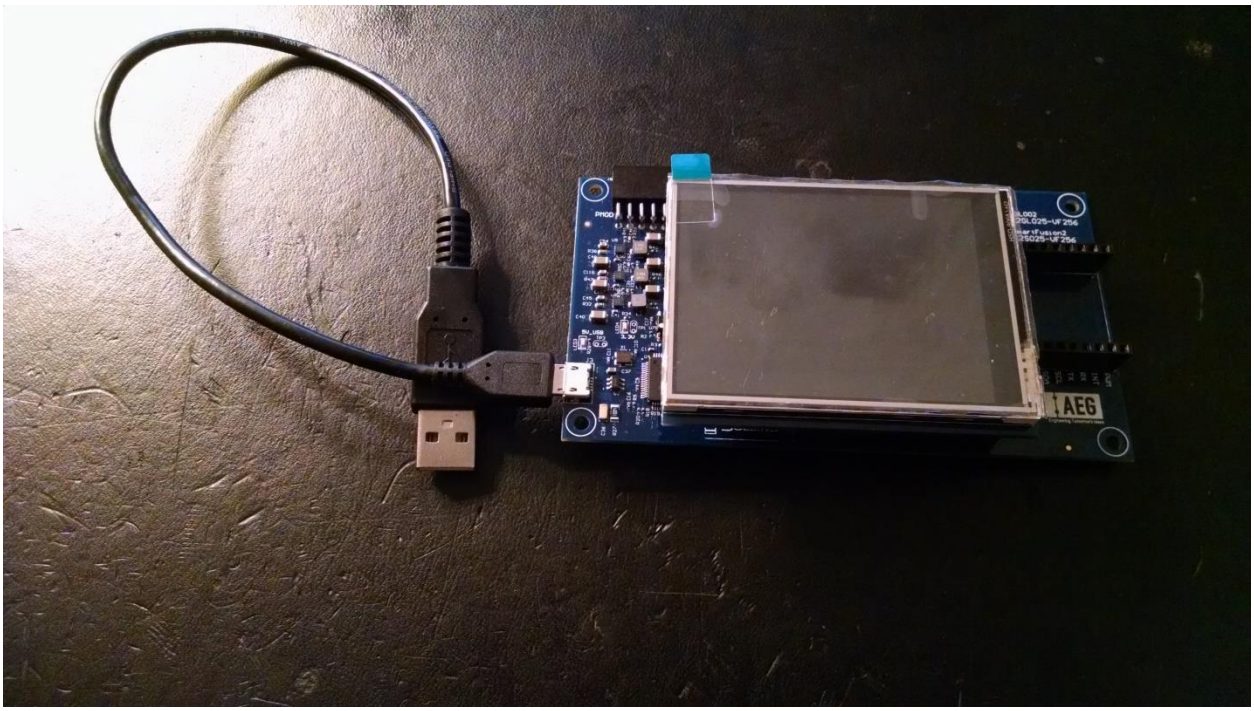
The following Microsemi provided link will provide more in depth information of the RISC-V ISA as well as how to install and use both FlashPro and SoftConsole. After going into the link a list of files will be shown, navigate through each file for better understanding of the process to use RISC-V or

for general knowledge of this architecture. Link: https://github.com/RISCV-on-Microsemi-FPGA/Documentation

Introduction to RISC-V:
In general, RISC-V is an Instruction Set Architecture (ISA) that was developed at UC Berkeley. The goal of this development is to have open source and proprietary implementations of the instruction set. As a result of this ISA, a soft processor was designed called CoreRISC_AX14 which was the core uploaded into the SmartFusion2 FPGA. The block diagram of this core can be seen below.



In the implementation provided by Microsemi for the Future Electronics Creative Board, the RISC-V processor is implemented via a Verilog based design file and loaded via Libero SoC. The program itself that is executed by the RISC-V processor is developed in SoftConsole and either loaded into the RAM to execute by SoftConsole's debugger or a "Release" linker script is used to generate a HEX file that is flashed into the eNVM memory using Libero SoC. This flashed program will execute at bootup. To visualize the Tic-Tac Toe example, an Adafruit SPI 2.8" touch screen is required (https://www.adafruit.com/product/1651). This mounted on the Arduino headers of the Future Creative board is shown below:

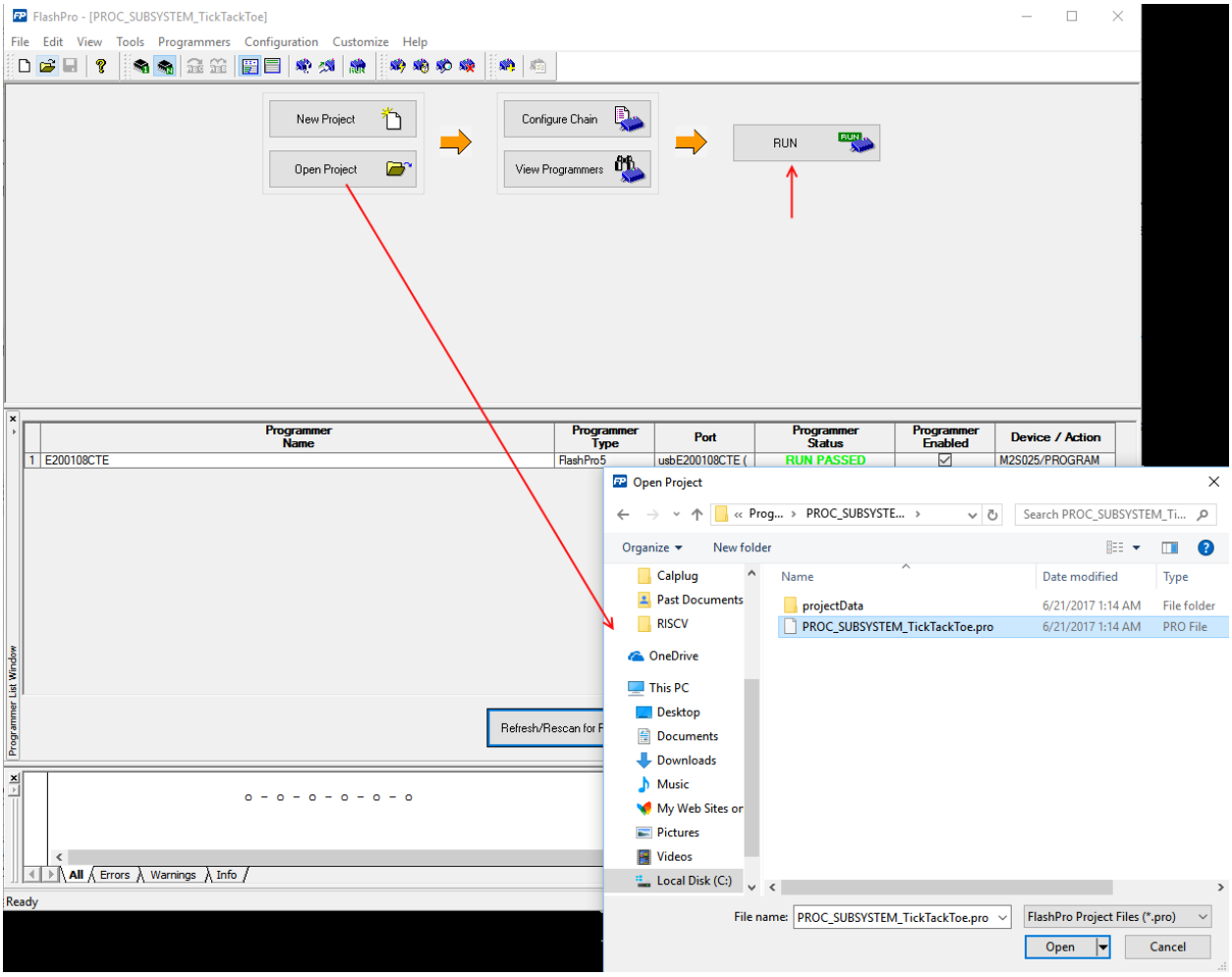


The example project is available here to download for Tic-Tac-Toe and Systic-Blinky demos: https://github.com/RISCV-on-Microsemi-FPGA/M2S025-Creative-Board/

There are three approaches to operate the demonstration. The first method (1) uses Microsemi's FlashPro software to upload the example as a completed project to execute at bootup. The second and third methods methods use Libero SoC and Soft Console to execute the demonstration either from the SoftConsole debugger (2), or to use Soft Console to generate a HEX file that is loaded into the eNVM to execute the RISC-V program at bootup (3).

Using FlashPro to Load the Tic-Tac-Toe Demo:

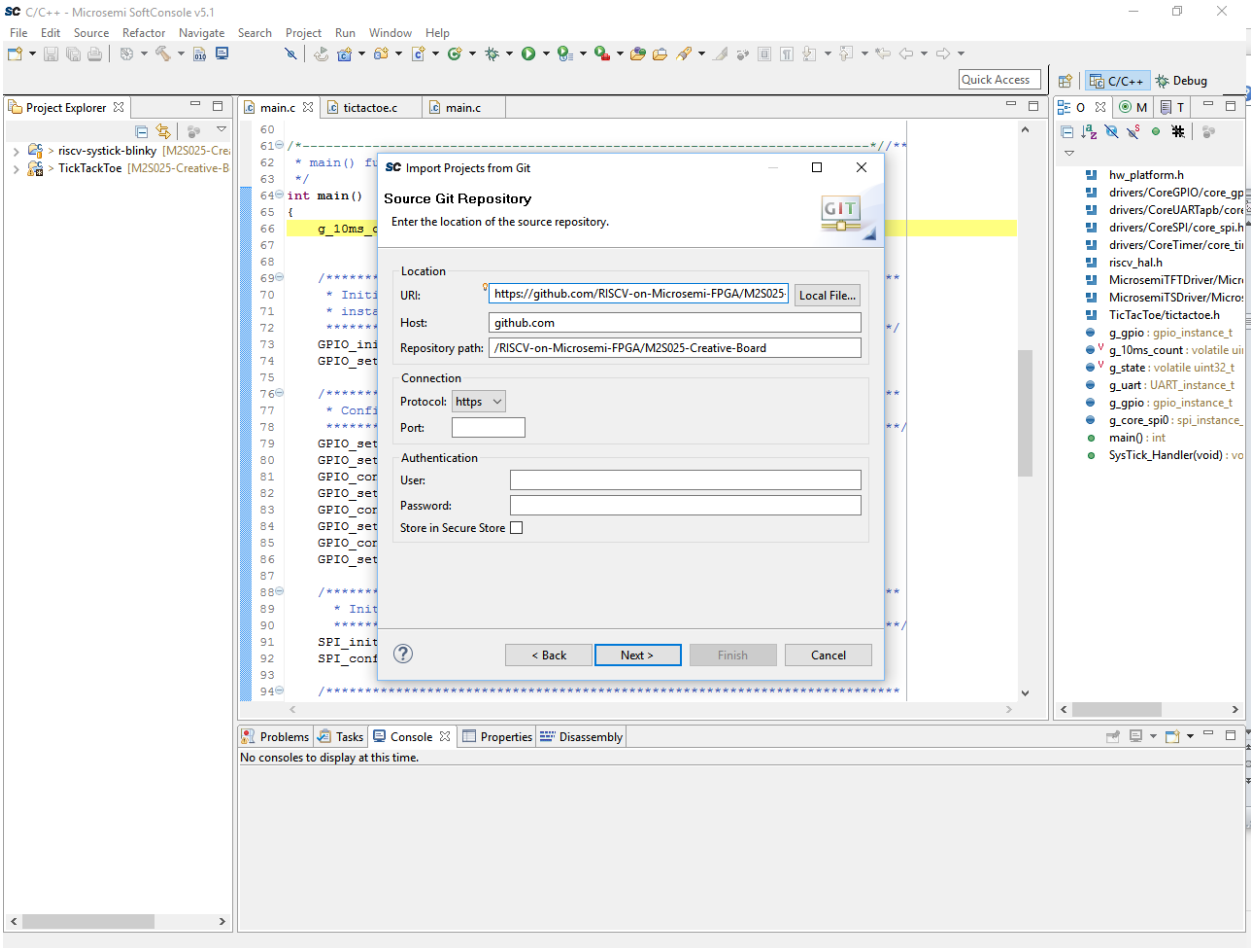
Microsemi FlashPro is downloaded and installed in preparation for use. The program is used to load the Microsemi provided example project. The following is the current location of the project: "M2S025-Creative-Board\Programming_The_Target_Device\PROC_SUBSYSTEM_TickTackToe" Please refer to project README instructions for more information.



Using Libero and Soft Console to Load the Tic-Tac-Toe Demo:

Debugger Program Execution

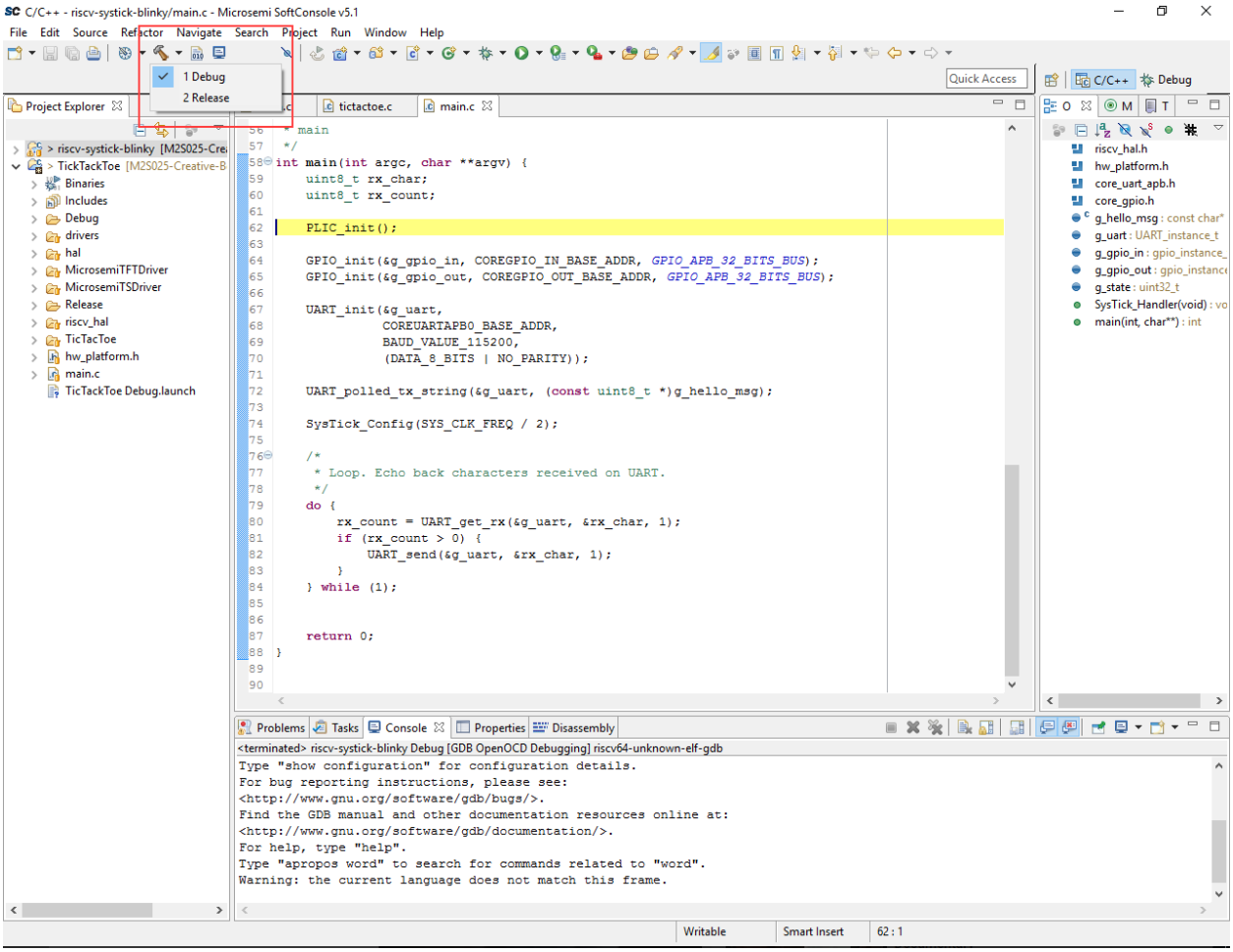
Libero SoC and Soft Console are used to build and load the demo project. Currently, the workspace must be imported into SoftConsole directly from github. This can be done using the Git tool from inside the SoftConsole import utility. Downloading and importing the workspace directly can cause issues that result in the project not loading properly. These steps show the process to load both the RISC-V core and the program. Because of workflow, the program build with SoftConsole is shown first. Be aware that until Libero is used to program the FPGA fabric, the RISC-V programs will not execute.



Be sure the project is saved to a folder location where the path to the file has no spaces, for instance:

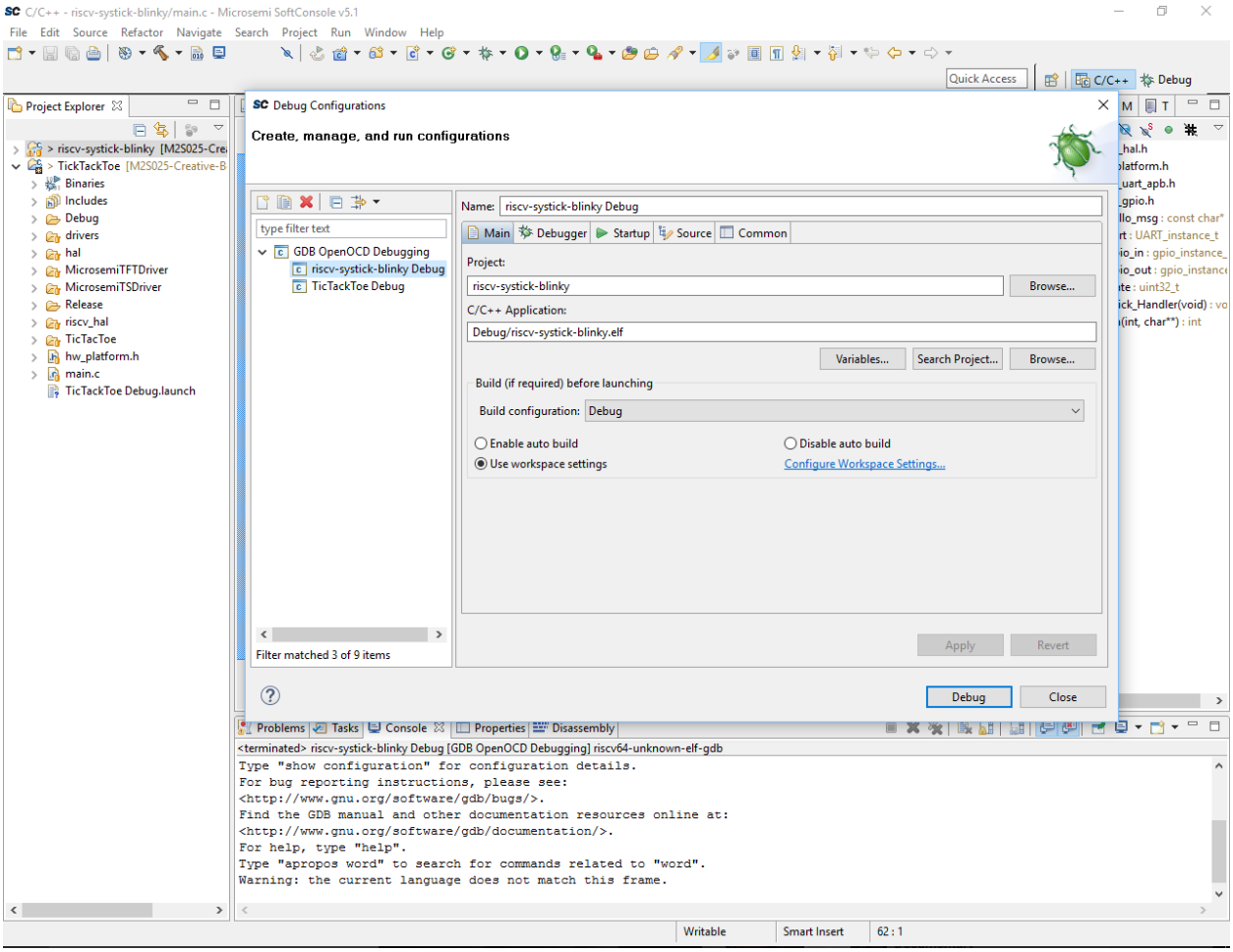
"C:\Microsemi\RISCV\M2S025-Creative-Board\Example_Software_Projects\"

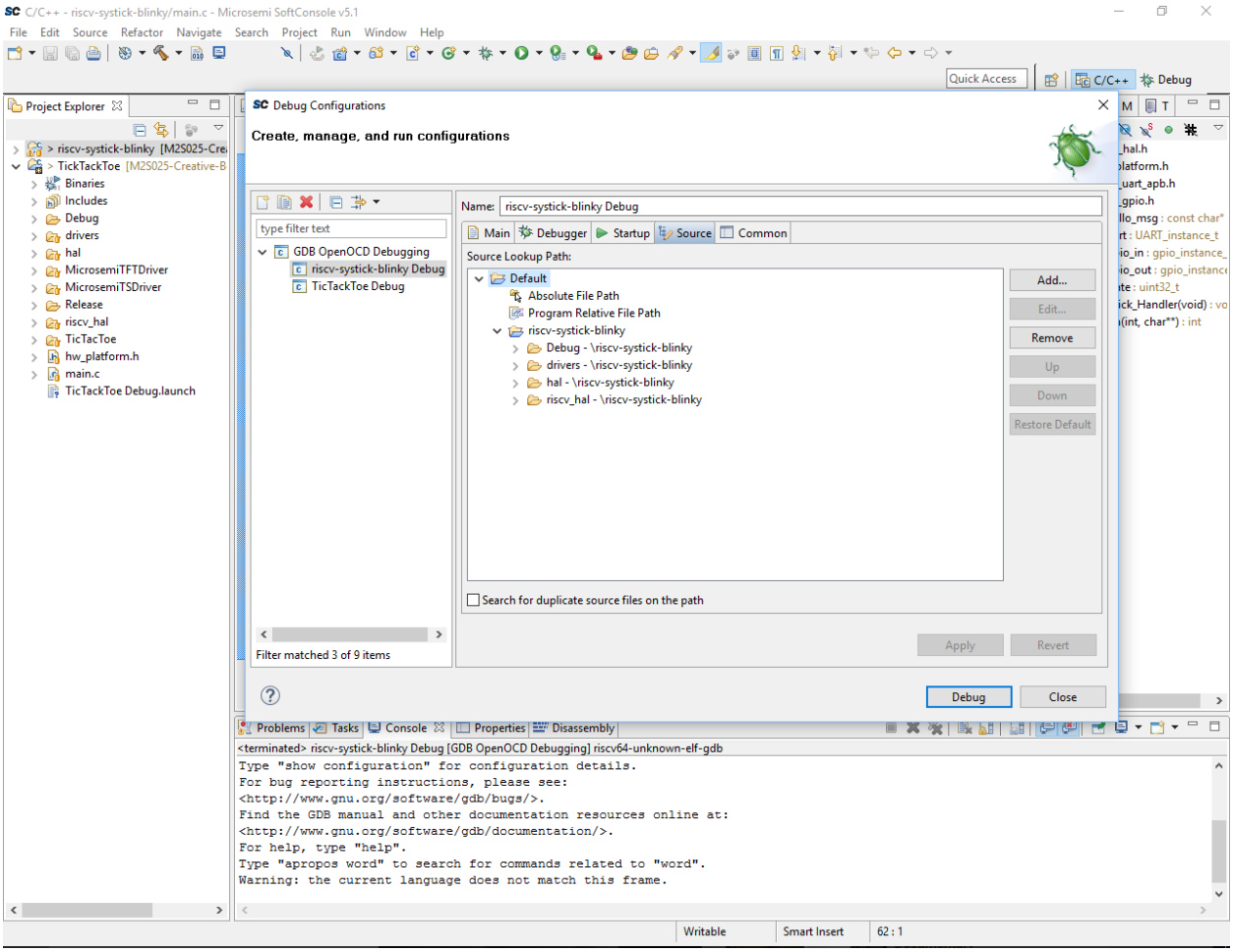
Once the program is loaded, try to build it and verify the project builds without issue. This is a first step in making sure the project is properly loaded and ready to use. Both Debug and Release build modes are shown. Make sure both build options are operational for both projects:

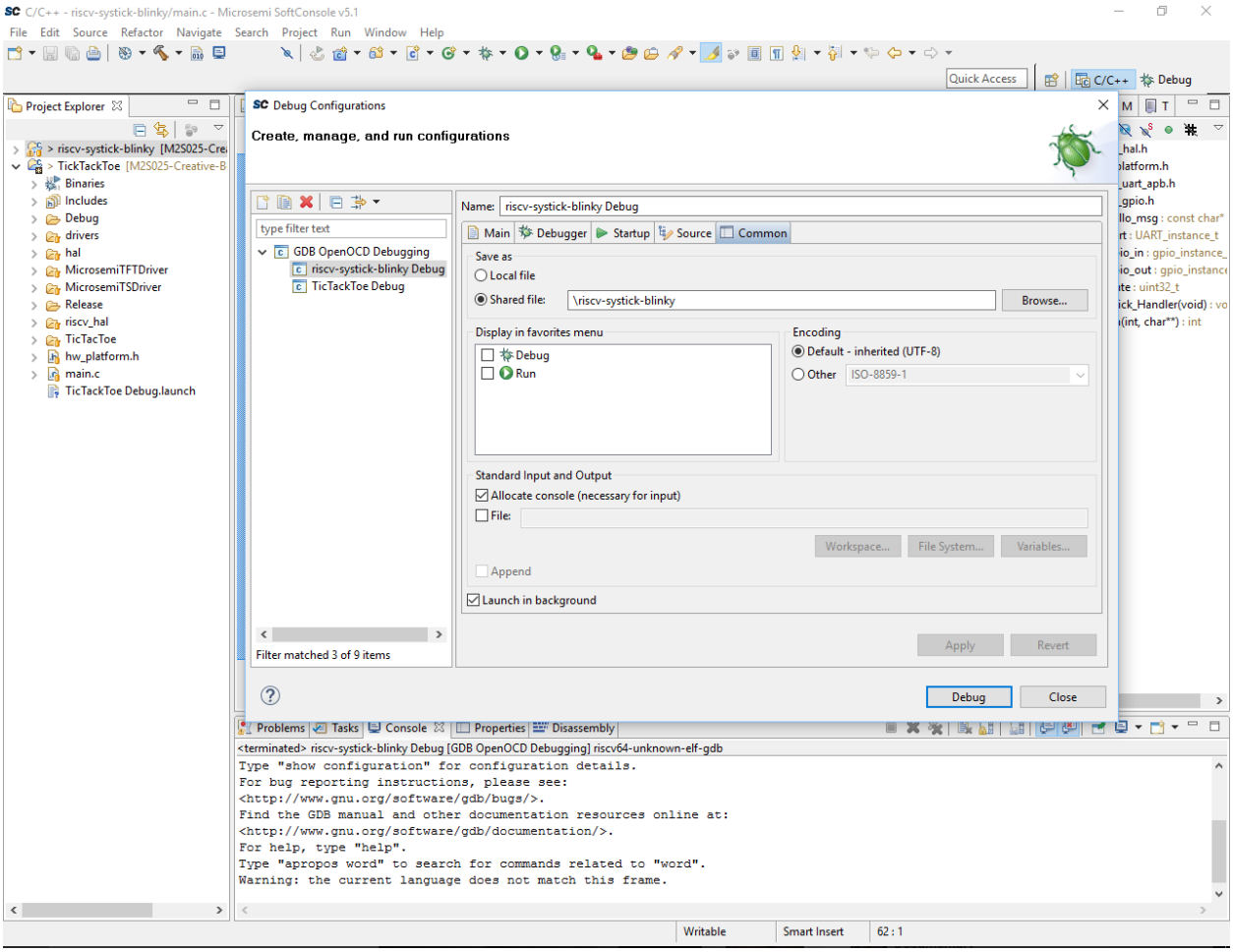


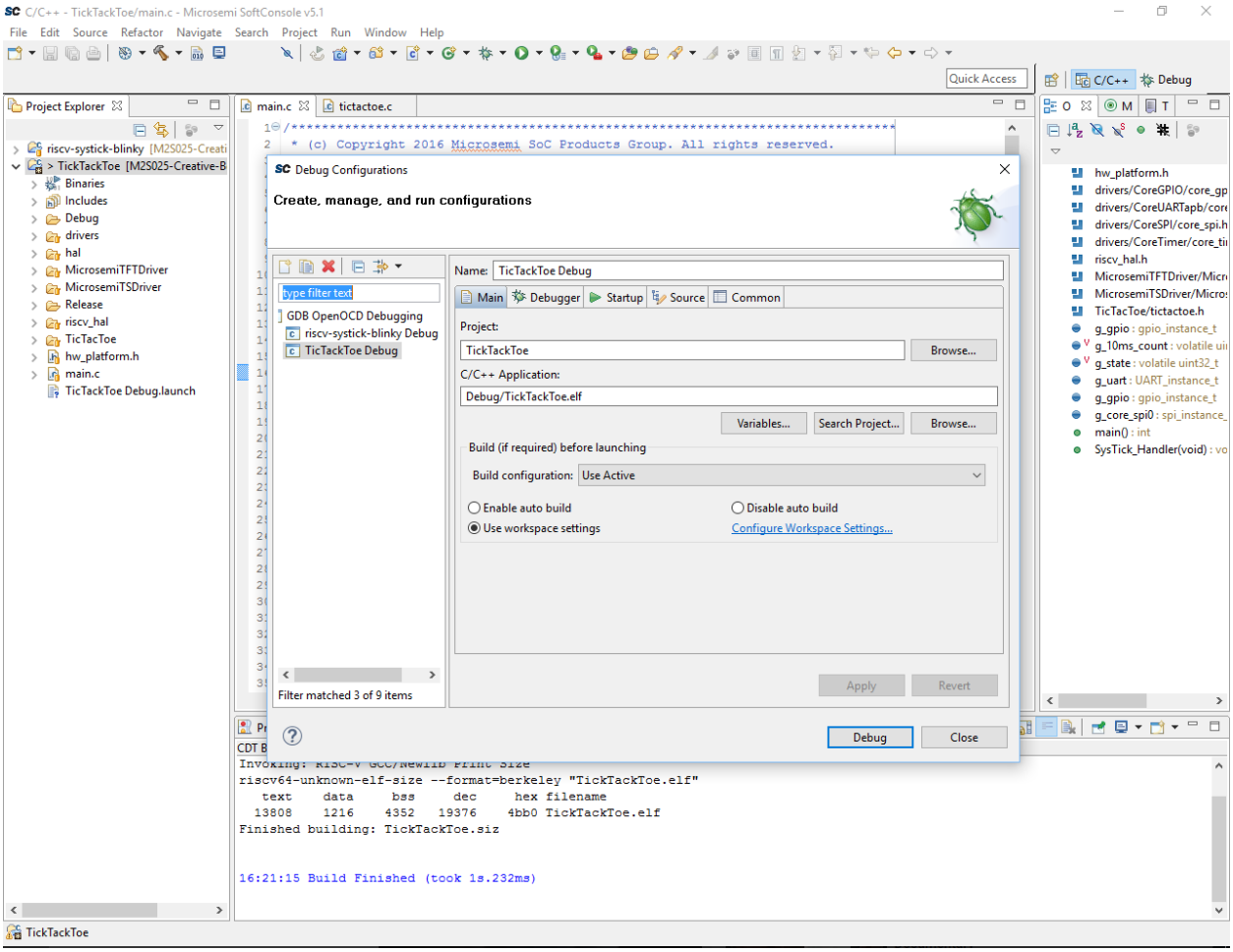
As mentioned in the known issues, the linker settings that are used to generate the project may not be set properly to allow execution in the debugger. When debugging using Soft Console the programed file gets loaded into the DDR at memory address 0x80000000. Additionally, if the production linker script is not in place, the final production executable that would be loaded into eNVM would not be properly generated.

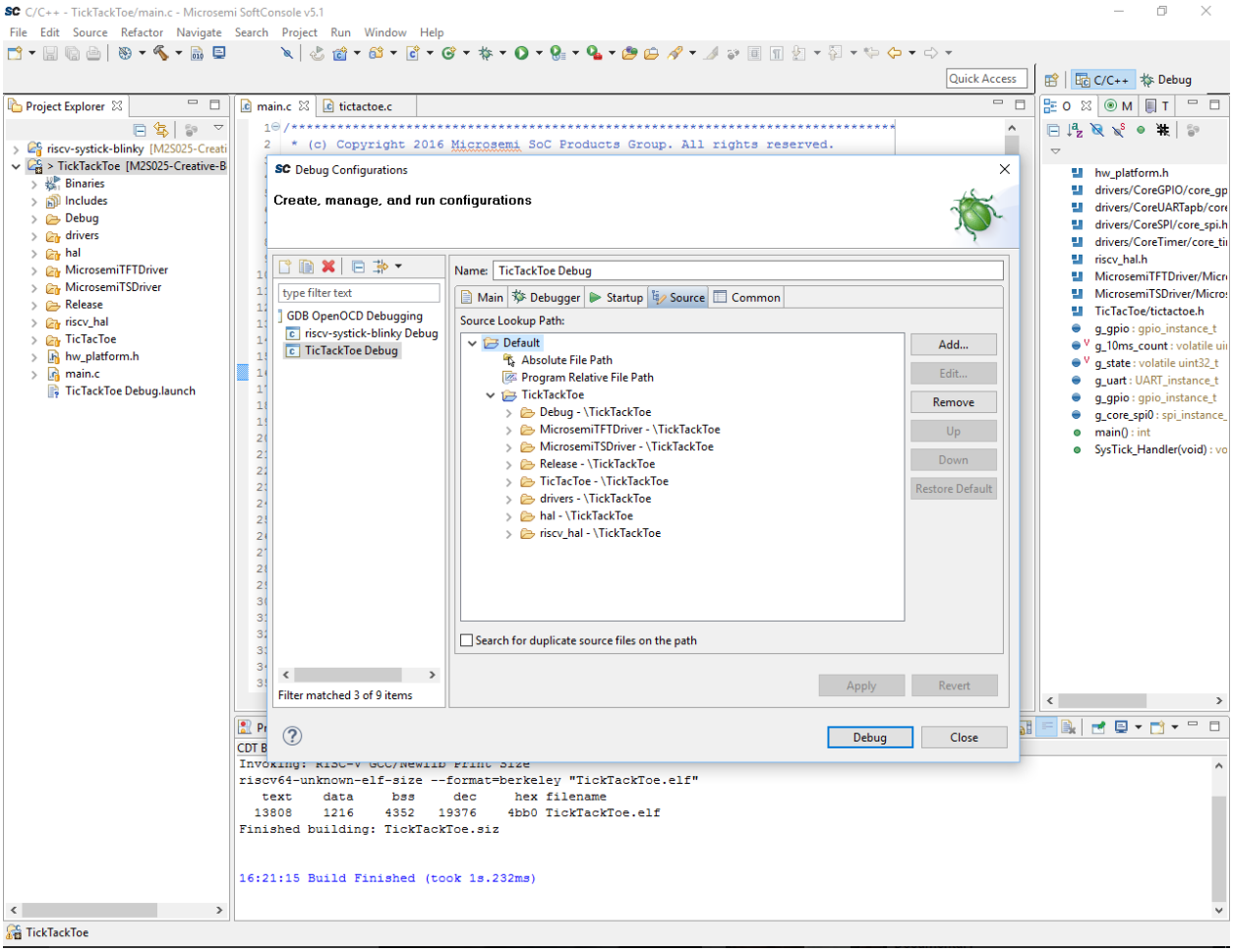
Please use the following images to properly set the location for the linker for the debugger for the Tic Tac Toe and Systic-Blinky projects. Shown are the settings required to execute from RAM via the debugger for both projects.

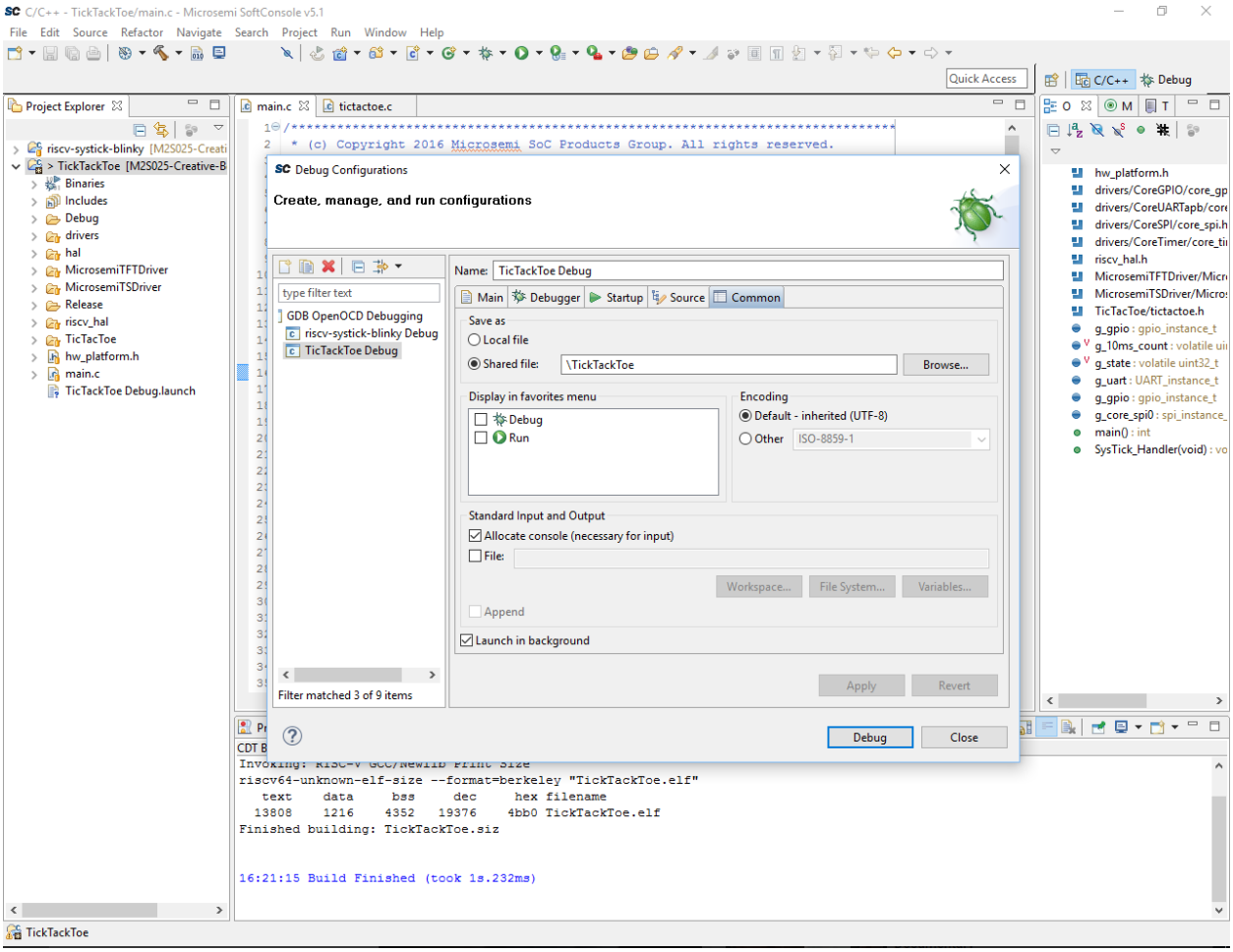

C/C++ - riscv-systick-blinky/main.c - Microsemi SoftConsole v5.1
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
C/C++
Debug
Project Explorer
riscv-systick-blinky [M2S025-Cre
TickTackToe [M2S025-Creative-B
Binaries
Includes
Debug
drivers
hal
MicrosemiTFTDriver
MicrosemiTSDriver
Release
riscv_hal
TicTacToe
hw_platform.h
main.c
TicTackToe Debug.launch
Debug Configurations
Create, manage, and run configurations
type filter text
GDB OpenOCD Debugging
riscv-systick-blinky Debug
TicTackToe Debug
Filter matched 3 of 9 items
Name: riscv-systick-blinky Debug
Main
Debugger
Startup
Source
Common
Project:
riscv-systick-blinky
Browse...
C/C++ Application:
Debug/riscv-systick-blinky.elf
Variables...
Search Project...
Browse...
Build (if required) before launching
Build configuration: Debug
Enable auto build
Disable auto build
Use workspace settings
Configure Workspace Settings...
Apply
Revert
Debug
Close
Problems
Tasks
Console
Properties
Disassembly
<terminated> riscv-systick-blinky Debug [GDB OpenOCD Debugging] riscv64-unknown-elf-gdb
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word".
Warning: the current language does not match this frame.
Writable
Smart Insert
62 : 1


C/C++ - riscv-systick-blinky/main.c - Microsemi SoftConsole v5.1
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
C/C++
Debug
Project Explorer
riscv-systick-blinky [M2S025-Cre
TickTackToe [M2S025-Creative-B
Binaries
Includes
Debug
drivers
hal
MicrosemiTFTDriver
MicrosemiTSDriver
Release
riscv_hal
TicTacToe
hw_platform.h
main.c
TicTackToe Debug.launch
Debug Configurations
Create, manage, and run configurations
type filter text
GDB OpenOCD Debugging
riscv-systick-blinky Debug
TicTackToe Debug
Filter matched 3 of 9 items
Name: riscv-systick-blinky Debug
Main
Debugger
Startup
Source
Common
Source Lookup Path:
Default
Absolute File Path
Program Relative File Path
riscv-systick-blinky
Debug - \riscv-systick-blinky
drivers - \riscv-systick-blinky
hal - \riscv-systick-blinky
riscv_hal - \riscv-systick-blinky
Add...
Edit...
Remove
Up
Down
Restore Default
Search for duplicate source files on the path
Apply
Revert
Debug
Close
Problems
Tasks
Console
Properties
Disassembly
<terminated> riscv-systick-blinky Debug [GDB OpenOCD Debugging] riscv64-unknown-elf-gdb
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word".
Warning: the current language does not match this frame.
Writable
Smart Insert
62 : 1


C/C++ - riscv-systick-blinky/main.c - Microsemi SoftConsole v5.1
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
C/C++
Debug
Project Explorer
riscv-systick-blinky [M2S025-Cre
TickTackToe [M2S025-Creative-B
Binaries
Includes
Debug
drivers
hal
MicrosemiTFTDriver
MicrosemiTSDriver
Release
riscv_hal
TicTacToe
hw_platform.h
main.c
TicTackToe Debug.launch
Debug Configurations
Create, manage, and run configurations
type filter text
GDB OpenOCD Debugging
riscv-systick-blinky Debug
TicTackToe Debug
Filter matched 3 of 9 items
Name: riscv-systick-blinky Debug
Main
Debugger
Startup
Source
Common
Save as
Local file
Shared file:
\riscv-systick-blinky
Browse...
Display in favorites menu
Debug
Run
Encoding
Default - inherited (UTF-8)
Other
ISO-8859-1
Standard Input and Output
Allocate console (necessary for input)
File:
Workspace...
File System...
Variables...
Append
Launch in background
Apply
Revert
Debug
Close
Problems
Tasks
Console
Properties
Disassembly
<terminated> riscv-systick-blinky Debug [GDB OpenOCD Debugging] riscv64-unknown-elf-gdb
Type "show configuration" for configuration details.
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>.
Find the GDB manual and other documentation resources online at:
<http://www.gnu.org/software/gdb/documentation/>.
For help, type "help".
Type "apropos word" to search for commands related to "word".
Warning: the current language does not match this frame.
Writable
Smart Insert
62 : 1


C/C++ - TickTackToe/main.c - Microsemi SoftConsole v5.1
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
C/C++
Debug
Project Explorer
riscv-systick-blinky [M2S025-Creati
TickTackToe [M2S025-Creative-B
Binaries
Includes
Debug
drivers
hal
MicrosemiTFTDriver
MicrosemiTSDriver
Release
riscv_hal
TicTacToe
hw_platform.h
main.c
TicTackToe Debug.launch
main.c
tictactoe.c
* (c) Copyright 2016 Microsemi SoC Products Group. All rights reserved.
Debug Configurations
Create, manage, and run configurations
type filter text
GDB OpenOCD Debugging
riscv-systick-blinky Debug
TicTackToe Debug
Filter matched 3 of 9 items
Name: TicTackToe Debug
Main
Debugger
Startup
Source
Common
Project:
TickTackToe
Browse...
C/C++ Application:
Debug/TickTackToe.elf
Variables...
Search Project...
Browse...
Build (if required) before launching
Build configuration: Use Active
Enable auto build
Disable auto build
Use workspace settings
Configure Workspace Settings...
Apply
Revert
Debug
Close
hw_platform.h
drivers/CoreGPIO/core_gp
drivers/CoreUARTapb/core
drivers/CoreSPI/core_spi.h
drivers/CoreTimer/core_ti
riscv_hal.h
MicrosemiTFTDriver/Micr
MicrosemiTSDriver/Micro
TicTacToe/tictactoe.h
g_gpio : gpio_instance_t
g_10ms_count : volatile ui
g_state : volatile uint32_t
g_uart : UART_instance_t
g_gpio : gpio_instance_t
g_core_spi0 : spi_instance_
main() : int
SysTick_Handler(void) : vo
riscv64-unknown-elf-size --format=berkeley "TickTackToe.elf"
text data bss dec hex filename
13808 1216 4352 19376 4bb0 TickTackToe.elf
Finished building: TickTackToe.siz
16:21:15 Build Finished (took 1s.232ms)
TickTackToe


C/C++ - TickTackToe/main.c - Microsemi SoftConsole v5.1
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
C/C++
Debug
Project Explorer
riscv-systick-blinky [M2S025-Creati
TickTackToe [M2S025-Creative-B
Binaries
Includes
Debug
drivers
hal
MicrosemiTFTDriver
MicrosemiTSDriver
Release
riscv_hal
TicTacToe
hw_platform.h
main.c
TicTackToe Debug.launch
main.c
tictactoe.c
/*****
* (c) Copyright 2016 Microsemi SoC Products Group. All rights reserved.
Debug Configurations
Create, manage, and run configurations
type filter text
GDB OpenOCD Debugging
riscv-systick-blinky Debug
TicTackToe Debug
Filter matched 3 of 9 items
Name: TicTackToe Debug
Main
Debugger
Startup
Source
Common
Source Lookup Path:
Default
Absolute File Path
Program Relative File Path
TickTackToe
Debug - \TickTackToe
MicrosemiTFTDriver - \TickTackToe
MicrosemiTSDriver - \TickTackToe
Release - \TickTackToe
TicTacToe - \TickTackToe
drivers - \TickTackToe
hal - \TickTackToe
riscv_hal - \TickTackToe
Add...
Edit...
Remove
Up
Down
Restore Default
Search for duplicate source files on the path
Apply
Revert
Debug
Close
hw_platform.h
drivers/CoreGPIO/core_gp
drivers/CoreUARTapb/core
drivers/CoreSPI/core_spi.h
drivers/CoreTimer/core_ti
riscv_hal.h
MicrosemiTFTDriver/Micr
MicrosemiTSDriver/Micro
TicTacToe/tictactoe.h
g_gpio : gpio_instance_t
g_10ms_count : volatile ui
g_state : volatile uint32_t
g_uart : UART_instance_t
g_gpio : gpio_instance_t
g_core_spi0 : spi_instance_
main() : int
SysTick_Handler(void) : vo
riscv64-unknown-elf-size --format=berkeley "TickTackToe.elf"
text data bss dec hex filename
13808 1216 4352 19376 4bb0 TickTackToe.elf
Finished building: TickTackToe.siz
16:21:15 Build Finished (took 1s.232ms)
TickTackToe



The Soft Console debugger tool is used to load the built project into the board for execution. Remember, this will only work once the RISC-V FPGA design is loaded into the FPGA. Please continue reading for details on this critical step.

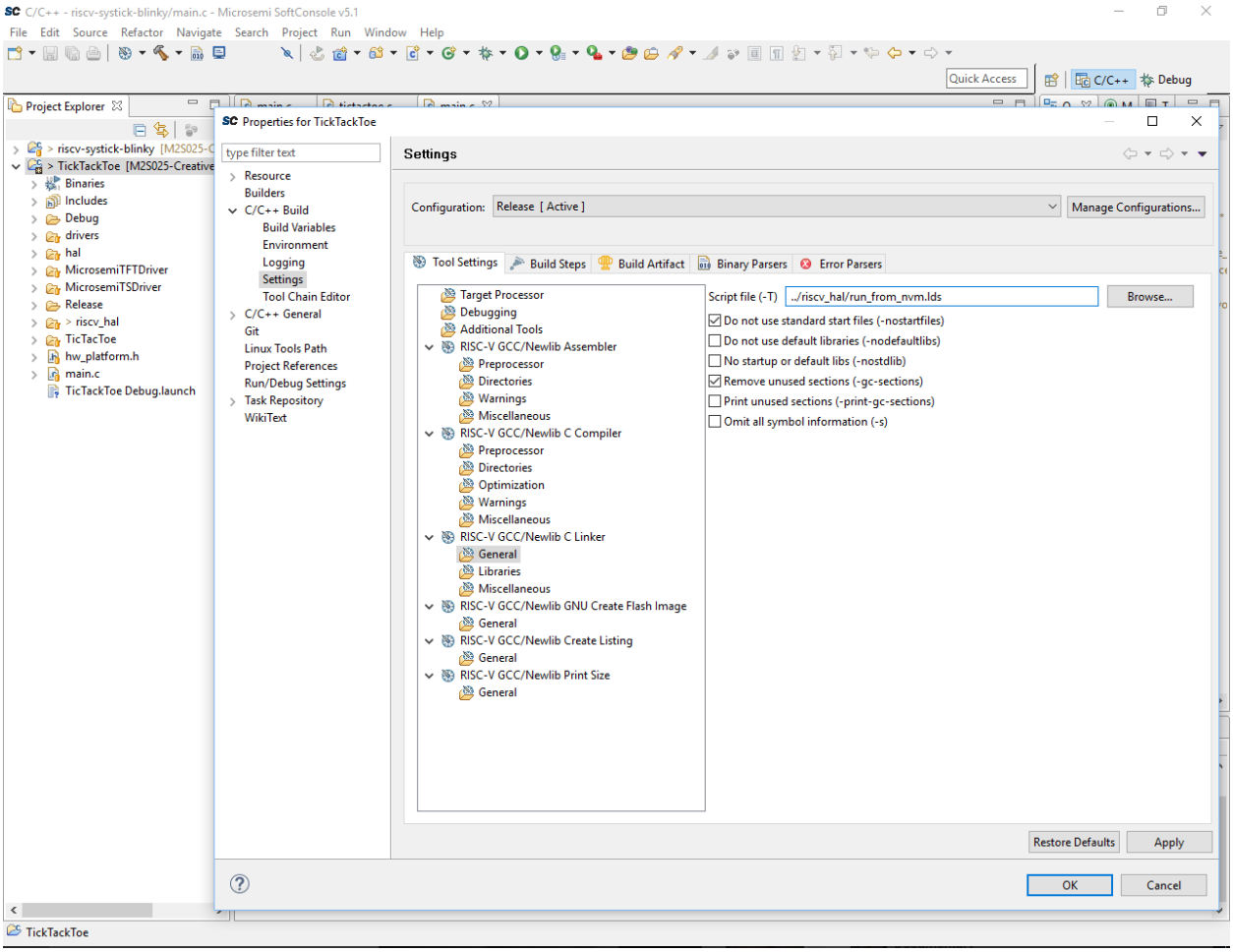
Program Execution from eNVM

A built project can also use the production linker scripts to produce an executable that can be loaded into the eNVM memory to execute at bootup. This is loaded into memory location (at 0x60000000) and should provide execute on bootup if the RISCVs reset vector should be set to 0x60000000. The location of both the RAM (debugger) and the eNVM linker scripts are shown below:

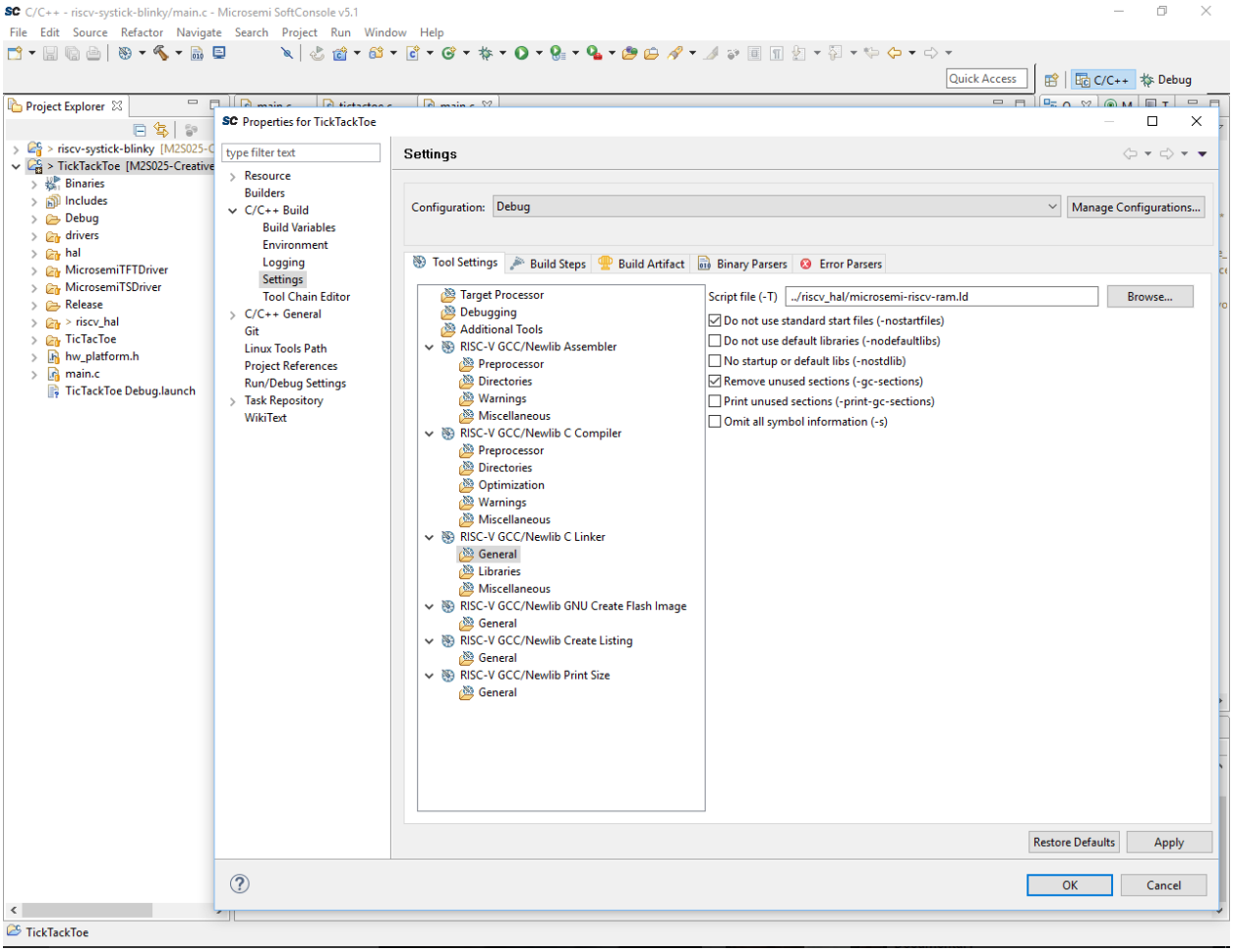
Local Disk (C:) > Microsemi > RISCV > M2S025-Creative-Board > Example_Software_Projects > TickTackToe > riscv_hal

Name	Date modified	Type	Size
encoding.h	6/21/2017 1:14 AM	C Source File	7 KB
entry.S	6/21/2017 1:14 AM	S File	4 KB
init.c	6/21/2017 1:14 AM	C Source File	2 KB
microsemi-riscv-ram.ld	6/21/2017 1:14 AM	LD File	4 KB
riscv_CoreplexE31.h	6/21/2017 1:14 AM	C Source File	8 KB
riscv_hal.c	6/21/2017 1:14 AM	C Source File	7 KB
riscv_hal.h	6/21/2017 1:14 AM	C Source File	2 KB
riscv_hal_stubs.c	6/21/2017 1:14 AM	C Source File	4 KB
run_from_nvm.lds	6/29/2017 4:33 PM	LDS File	2 KB
sample_hw_platform.h	6/21/2017 1:14 AM	C Source File	5 KB
syscall.c	6/21/2017 1:14 AM	C Source File	6 KB

Under the project properties, the Production build settings must refer to the “run_from_nvm.lds” linker script. The settings are shown below:

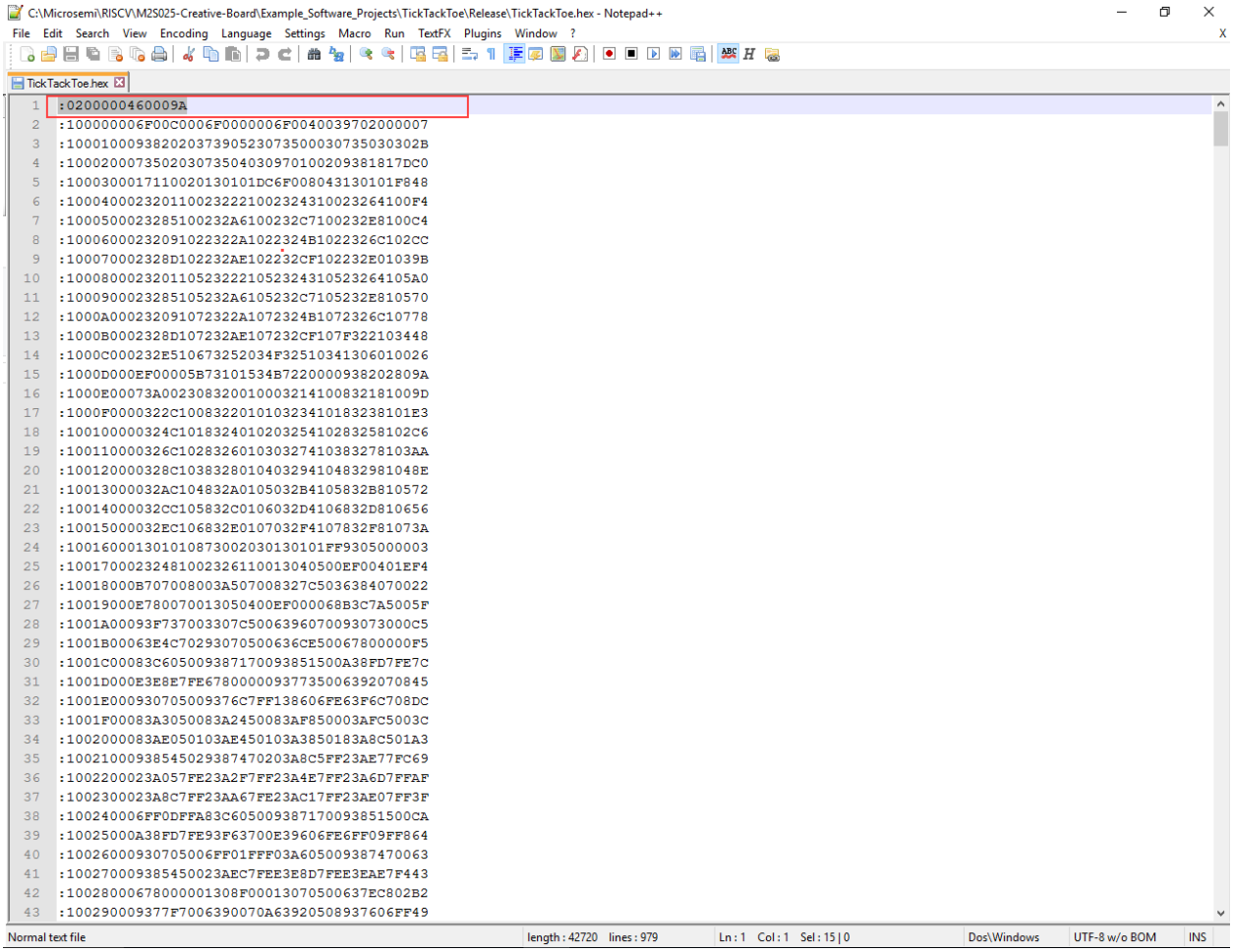


In contrast to the Release linker, the Debug profile must continue to refer to the “microsemi-riscv-ram.ld” debug linker file. This is shown below:



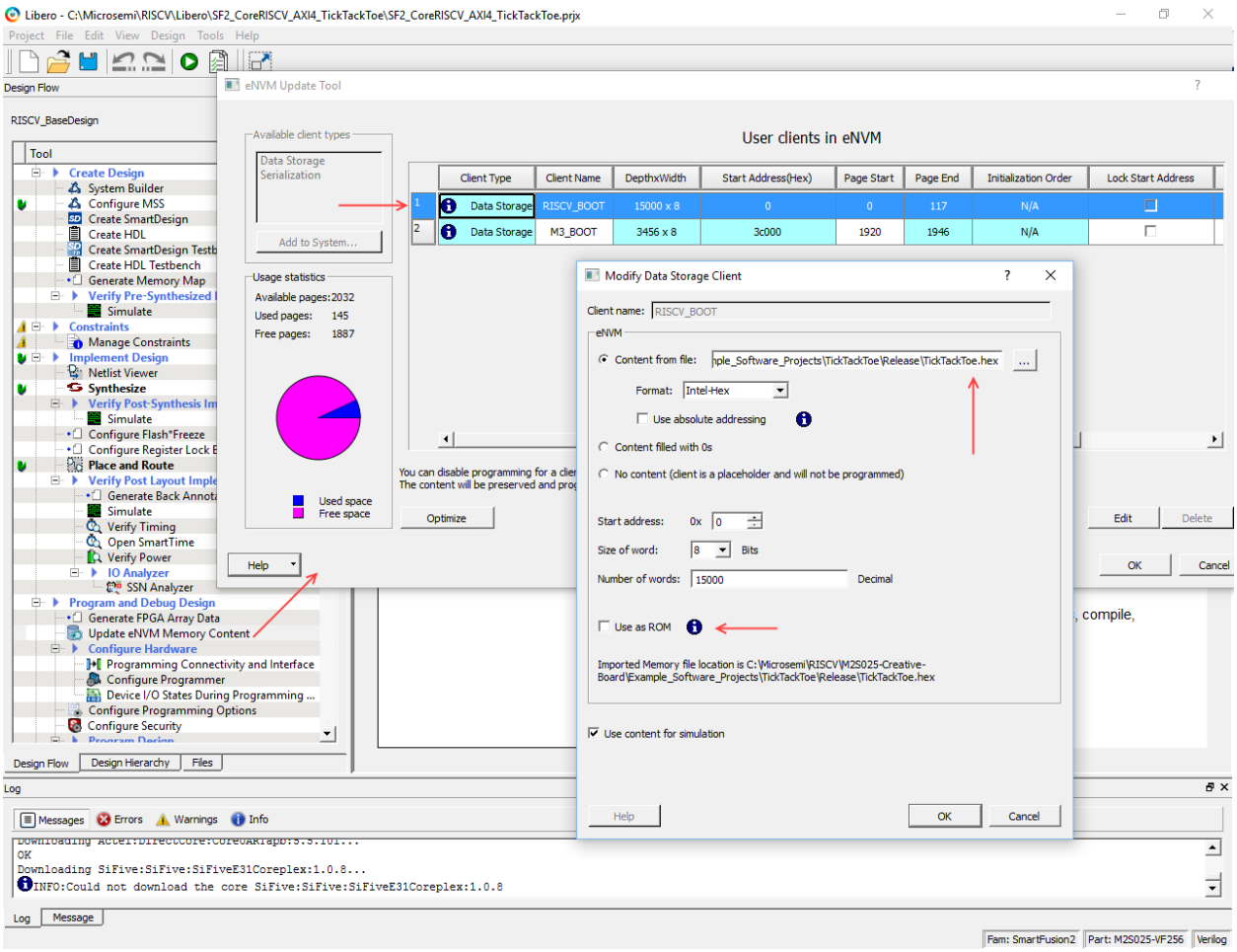
After these settings are changed, the project should build for both Debug and Release without issue.

A modification to the generated HEX file must be made to allow execution. The generated HEX file (which should be produced in the “Release” folder of the project) must be opened in a text editor and the first line deleted then the file is resaved. This is shown below:

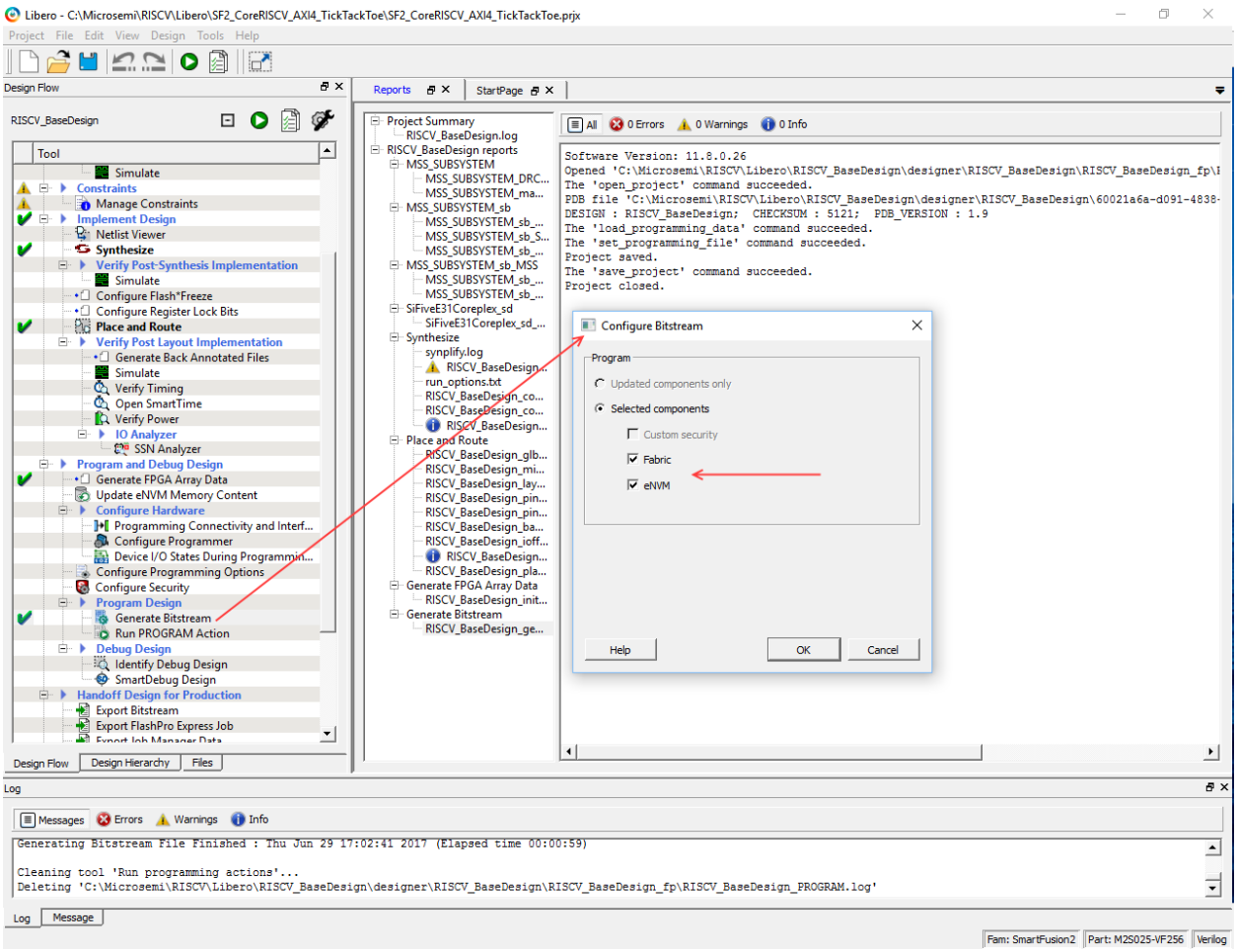


At this point both the Debug and Release (Production) executables have been generated. In order to execute either, the RISC-V core must be programmed. This is done with Libero SoC 11.8. Libero must be properly licenced to execute. Either an evaluation or silver license will work for the SmartFusion 2 device present on the Future Electronics Creative board. The Libero SoC project is available in the project folder from Github in the following location: "\M2S025-Creative-Board\Modify_The_FPGA_Design\SF2_CoreRISCV_AXI4_TickTackToe". This project is loaded and built in Libero SoC. The synthesis step can take a while to finish.

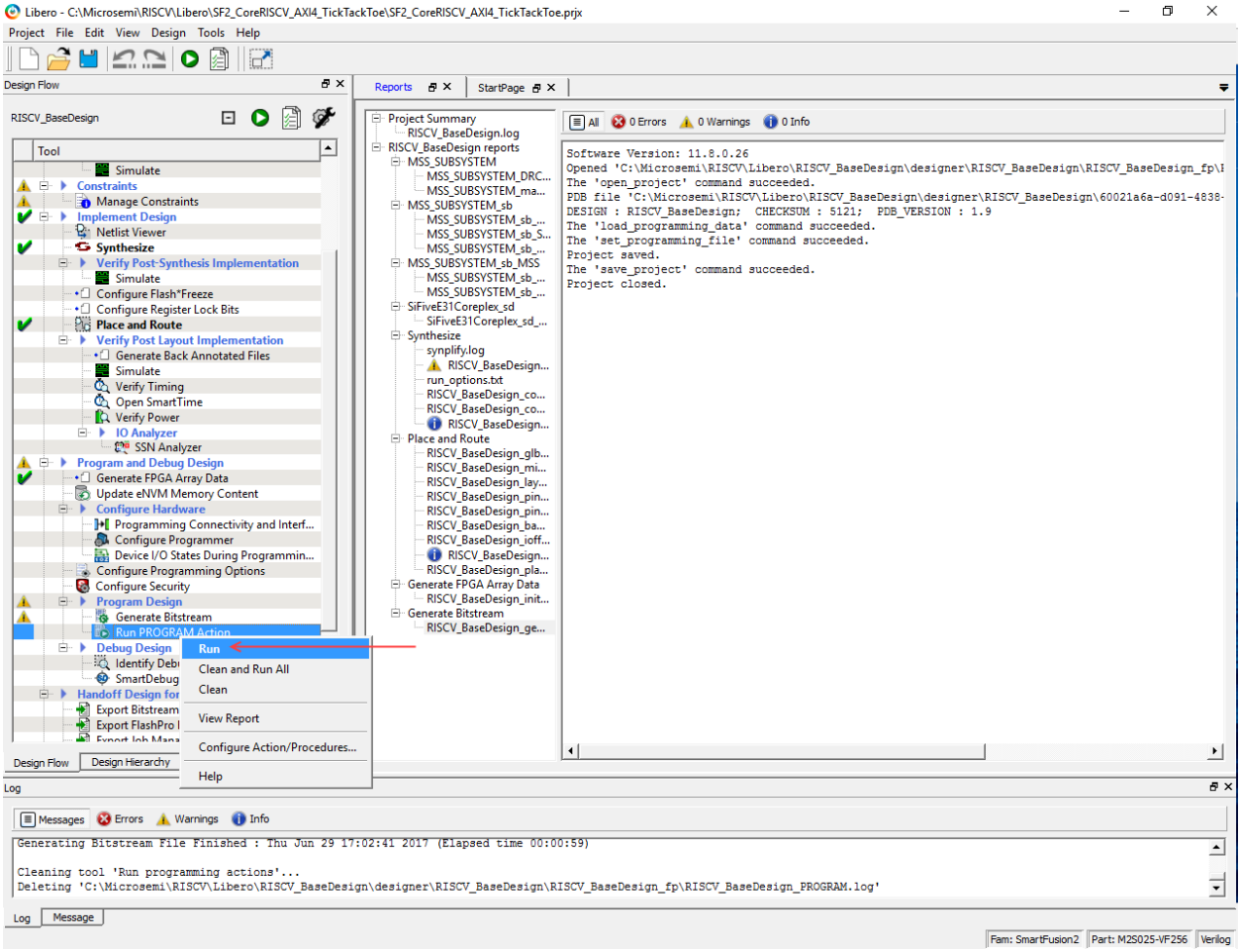
Once the project is built, the generated Release executable can be loaded into the board at the same time as the RISC-V core into the FPGA. The "Update eNVM Memory Content" tool can be used to upload the modified .HEX executable file for the release build. The settings are shown below.

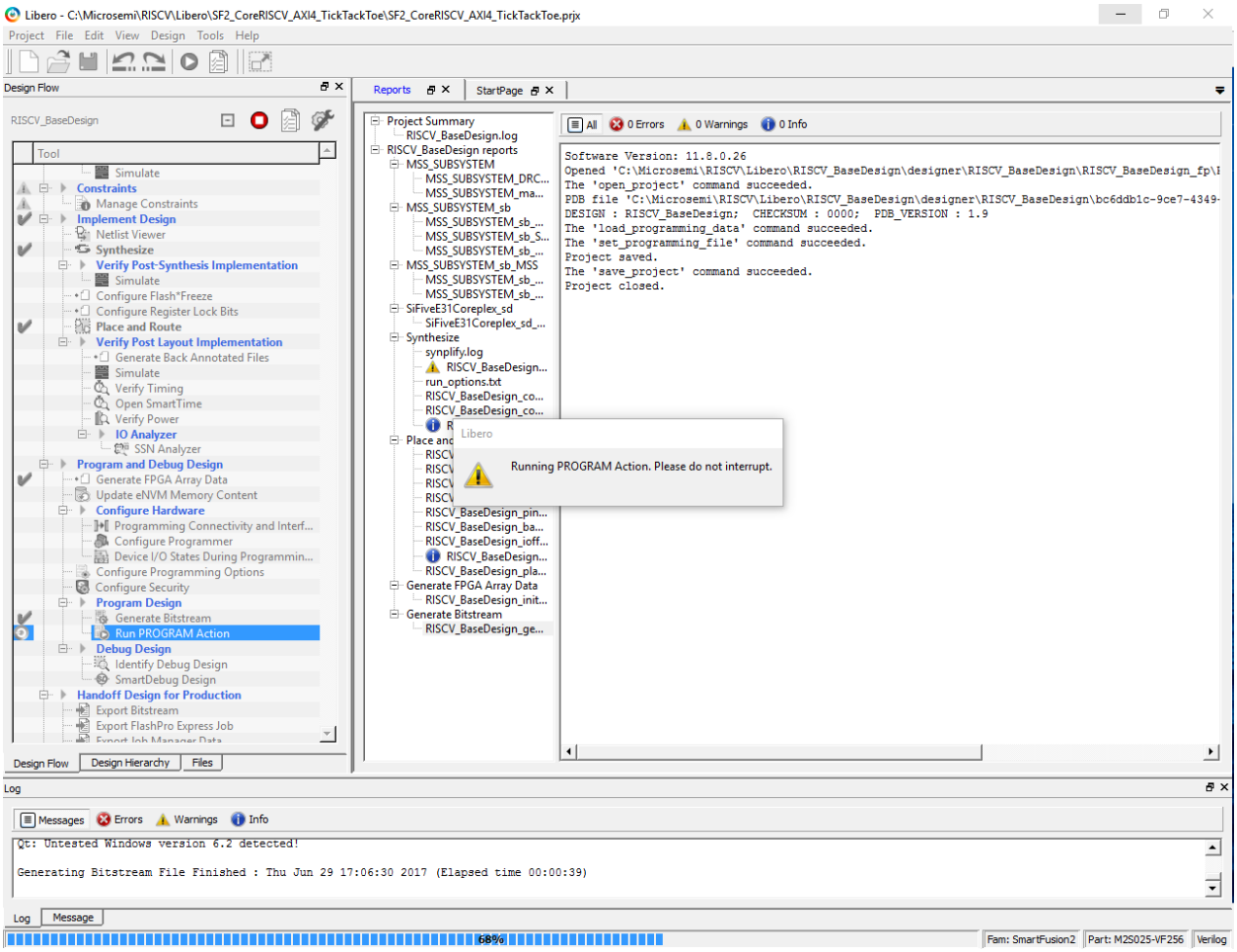


Before programming the SmartFusion2, the decision to program the contents of the eNVM and/or the FPGA fabric can be made. If the fabric has already been uploaded, it is advised to uncheck the "Fabric" option to save time and avoid excessive writes to the logic elements of the FPGA fabric. See image below:



Finally, to upload the Fabric and/or the eNVM contents, execute the “Run PROGRAM Action” function in the design workflow in Libro SoC (see below):


Libero - C:\Microsemi\RISCV\Libero\SF2_CoreRISCV_AXI4_TickTackToe\SF2_CoreRISCV_AXI4_TickTackToe.prjx
Project File Edit View Design Tools Help
Design Flow
RISCV_BaseDesign
Tool
Simulate
Constraints
Manage Constraints
Implement Design
Netlist Viewer
Synthesize
Verify Post-Synthesis Implementation
Simulate
Configure Flash*Freeze
Configure Register Lock Bits
Place and Route
Verify Post Layout Implementation
Generate Back Annotated Files
Simulate
Verify Timing
Open SmartTime
Verify Power
IO Analyzer
SSN Analyzer
Program and Debug Design
Generate FPGA Array Data
Update eNVM Memory Content
Configure Hardware
Programming Connectivity and Interf...
Configure Programmer
Device I/O States During Programmin...
Configure Programming Options
Configure Security
Program Design
Generate Bitstream
Run PROGRAM Action
Debug Design
Identify Debu
SmartDebug
Handoff Design for
Export Bitstream
Export FlashPro
Run
Clean and Run All
Clean
View Report
Configure Action/Procedures...
Help
Design Flow
Design Hierarchy
Reports
StartPage
Project Summary
RISCV_BaseDesign.log
RISCV_BaseDesign reports
MSS_SUBSYSTEM
MSS_SUBSYSTEM_DRC...
MSS_SUBSYSTEM_ma...
MSS_SUBSYSTEM_sb
MSS_SUBSYSTEM_sb_...
MSS_SUBSYSTEM_sb_S...
MSS_SUBSYSTEM_sb_...
MSS_SUBSYSTEM_sb_MSS
MSS_SUBSYSTEM_sb_...
MSS_SUBSYSTEM_sb_...
SiFiveE31Coreplex_sd
SiFiveE31Coreplex_sd_...
Synthesize
synplify.log
RISCV_BaseDesign...
run_options.txt
RISCV_BaseDesign_co...
RISCV_BaseDesign_co...
RISCV_BaseDesign...
Place and Route
RISCV_BaseDesign_glb...
RISCV_BaseDesign_mi...
RISCV_BaseDesign_lay...
RISCV_BaseDesign_pin...
RISCV_BaseDesign_pin...
RISCV_BaseDesign_ba...
RISCV_BaseDesign_ioff...
RISCV_BaseDesign...
RISCV_BaseDesign_pla...
Generate FPGA Array Data
RISCV_BaseDesign_init...
Generate Bitstream
RISCV_BaseDesign_ge...
All 0 Errors 0 Warnings 0 Info
Software Version: 11.8.0.26
Opened 'C:\Microsemi\RISCV\Libero\RISCV_BaseDesign\designer\RISCV_BaseDesign\RISCV_BaseDesign_fp\
The 'open_project' command succeeded.
PDB file 'C:\Microsemi\RISCV\Libero\RISCV_BaseDesign\designer\RISCV_BaseDesign\60021a6a-d091-4838-
DESIGN : RISCV_BaseDesign; CHECKSUM : 5121; PDB_VERSION : 1.9
The 'load_programming_data' command succeeded.
The 'set_programming_file' command succeeded.
Project saved.
The 'save_project' command succeeded.
Project closed.
Log
Messages Errors Warnings Info
Generating Bitstream File Finished : Thu Jun 29 17:02:41 2017 (Elapsed time 00:00:59)
Cleaning tool 'Run programming actions'...
Deleting 'C:\Microsemi\RISCV\Libero\RISCV_BaseDesign\designer\RISCV_BaseDesign\RISCV_BaseDesign_fp\RISCV_BaseDesign_PROGRAM.log'
Log Message
Fam: SmartFusion2 Part: M2S025-VF256 Verilog



Note: There are other RISC-V implementations available in the project folder including the one located at "\M2S025-Creative-Board\Modify_The_FPGA_Design\CoreRISCV_AXI4_BaseDesign" This design without modification will not permit the execution of the Tic-Tac-Toe example from eNVM or Debug execution.

Known Issues:

Expanding elements of a template RISC-V design can cause one to run into the following issue RISC-V design

(Number of RAM64x16 modules (64) exceeds the limit (34) of the selected device).

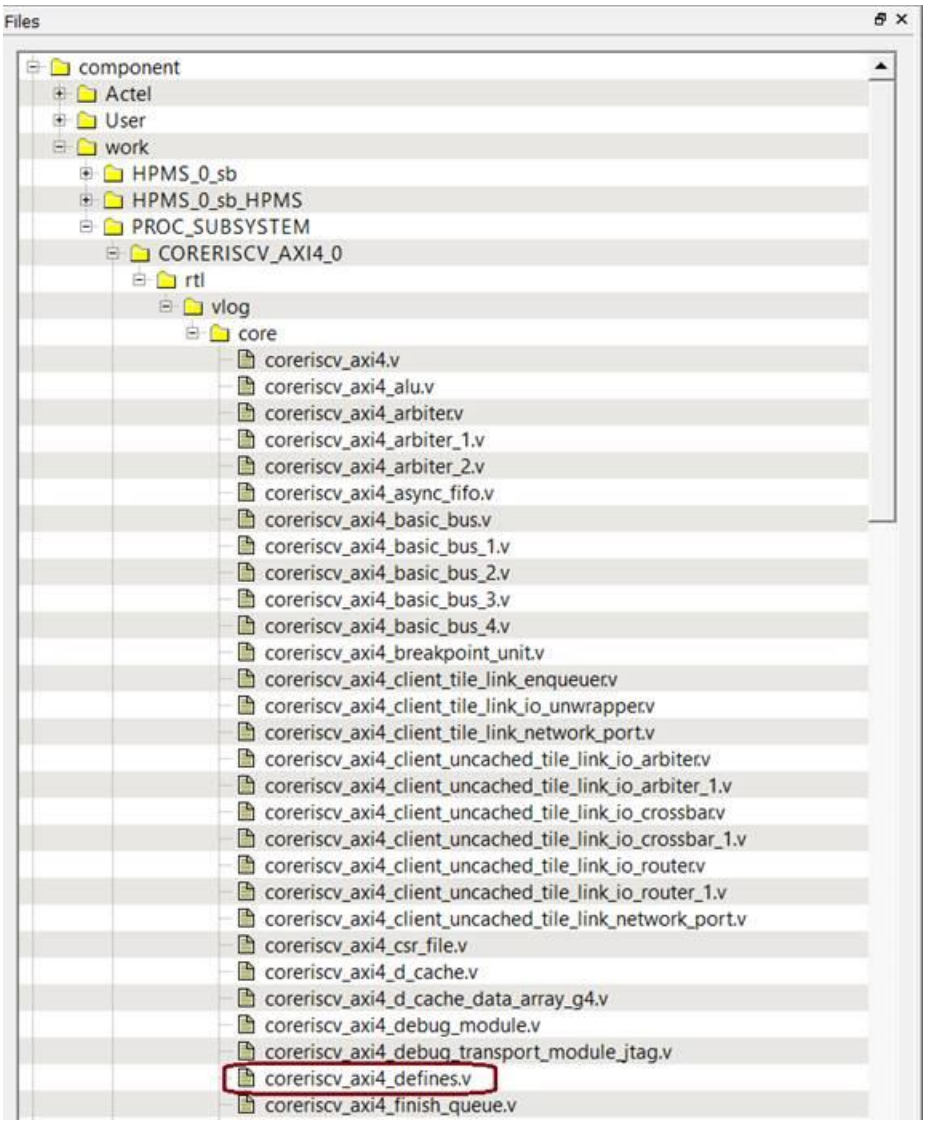
This issue is explained in Section 5.1.1 of the RISC-V handbook (attached):

"In order to facilitate instantiating this core on smaller Microsemi parts with limited RAM resources, all RAM inferences other than the Instruction and Data caches within CoreRISCV_AXI4 can be optionally implemented with fabric registers rather than allowing the synthesis tool to infer RAM. This requires modification of the coreriscv_axi4_defines.v file in the work/SmartDesign_name/CoreRISCV_AXI4_Instance_name/rtl/vlog/core folder of the Libero project as follows:

```
To use registers for all RAM blocks other than the internal instruction and data
caches, uncomment the USE_REGISTERS define.
Note: The contents of the coreriscv_axi4_defines.v file will need to be replaced
every time that the SmartDesign sheet containing the CorRISCV_AXI4 instance is
generated."
```

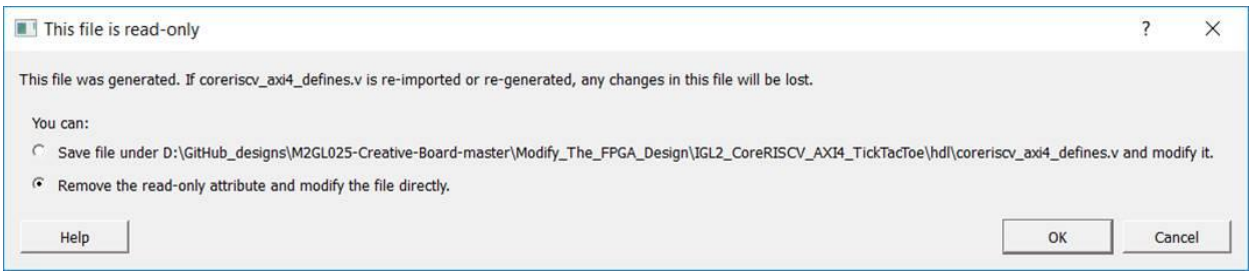
To address this issue, one can modify the coreriscv_axi_defines.v file within the Libero project as follows:

1. Select the Libero SoC Files tab and expand the component folder at the top of the list. Navigate to component > work > <SmartDesign_name> > <CoreRISCV_AXI4_Instance_name> > rtl > vlog > core

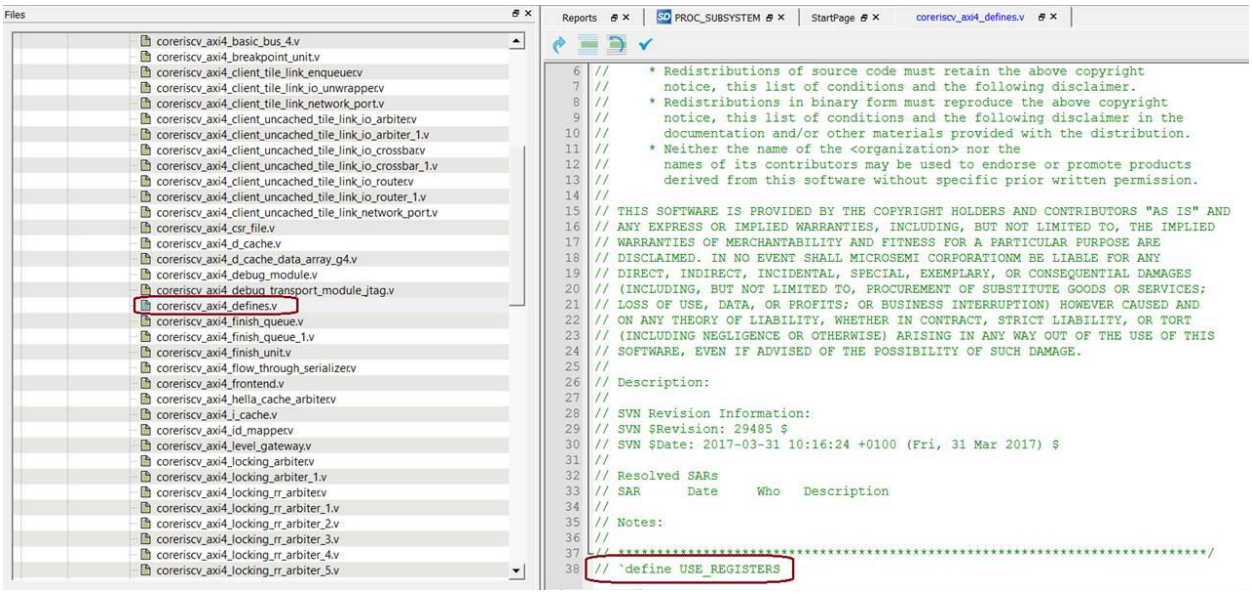


In the GitHub designs the SmartDesign_name is PROC_SUBSYSTEM and the CoreRISCV_AXI4_instance_name is CORERISCV_AXI_0

2. Double-click the file name to open it in the Libero SoC editor.
3. Remove the comments on line 38. Note that when you try to edit the file a dialog box will open indicating the file is read-only. You can save the file to a different location or remove the read only attribute. I chose the remove read-only attribute option.

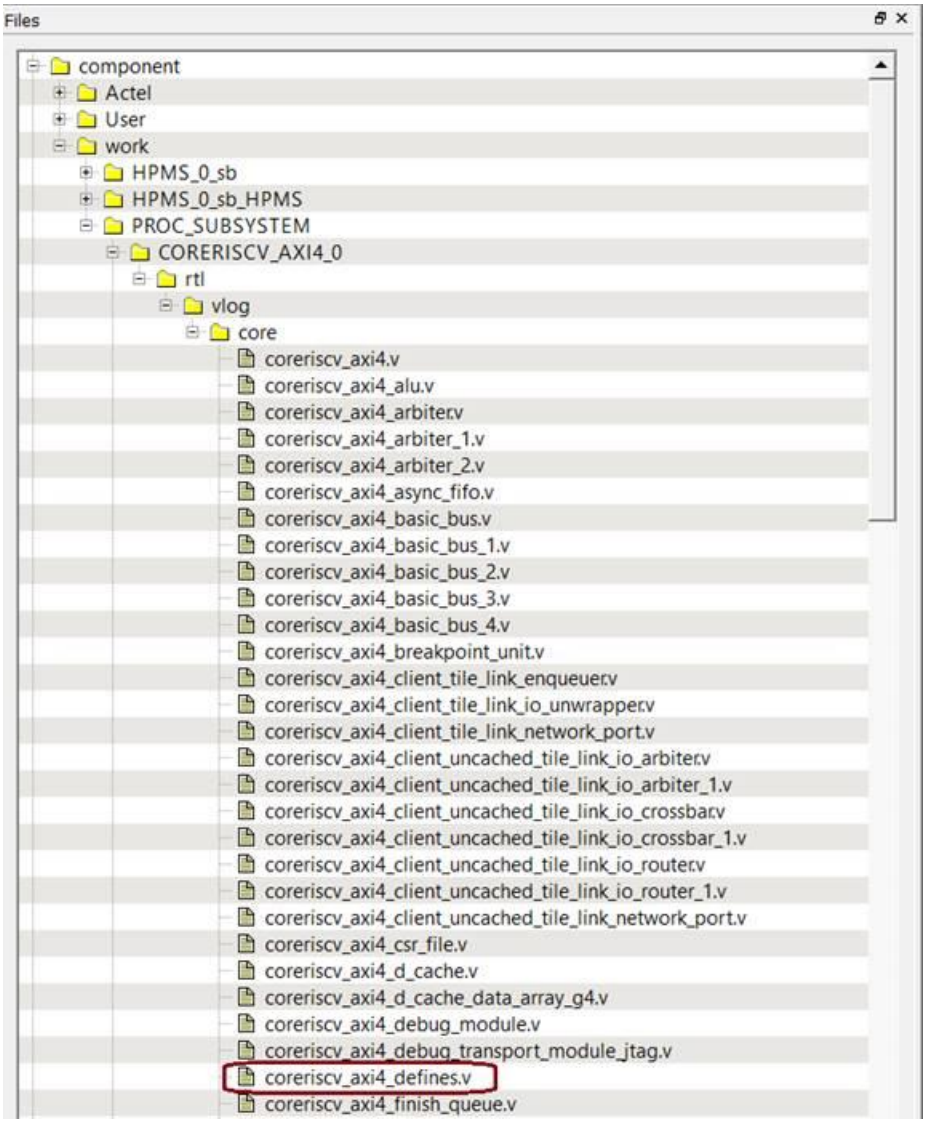


4. Click OK then un-comment line 38 and save the file.



5. After saving the coreriscv_axi4_defines.v file, continue through the normal design flow.

It is important to remember that the coreriscv_axi4_defines.v file is re-generated any time the design is generated in SmartDesign. So any time you change the design you will have to repeat the steps above to avoid the error about exceeding the number of RAM blocks.
Let me know if you have any questions.



Appendix:

In addition to the project README files and links contained within these. Here are some helpful links Microsemi's Implementation of RISC-V:

RISC-V info:
https://github.com/RISCV-on-Microsemi-FPGA/Documentation/blob/master/RISC-V_Soft_Processor_on_PolarFire.pdf

CoreRISC-V_AX14 info:
https://github.com/RISCV-on-Microsemi-FPGA/Documentation/blob/master/CoreRISC-V_AXI4_HB.pdf

Use of SoftConsole:
RISC-V on Microsemi FPGA products requires SoftConsole 5.0 under Linux and SoftConsole 5.1 under Windows. Libero SoC 11.8 was used.

SoftConsole Guide for RISC-V:
https://github.com/RISCV-on-Microsemi-FPGA/Documentation/blob/master/SoftConsoleV5%200_RISCV_Guide_v1%200.pdf