How to create a single bitstream with SoftConsole for a SmartFusion2

Created by: David Rubio G.

Blog post: https://soceame.wordpress.com/2025/03/10/how-to-create-a-single-bitstream-with-softconsole-for-a-smartfusion2/

Blog: https://soceame.wordpress.com/

GitHub: https://github.com/DRubioG

Last modification date: 10/03/25

In this post I’m going to explain how to generate a single bitstream with the Libero bitstream and the binary generated by the SoftConsole.
This is very useful when you want to record many SoCs with the same bitstream, such as when you are manufacturing PCBs.

To do this you first need to know the basics of how Libero and SoftConsole work.

https://soceame.wordpress.com/2025/03/09/basic-project-with-libero/
https://soceame.wordpress.com/2025/03/09/how-to-create-a-project-for-a-smartfusion2-board/

And you also need to be clear about how the SoftConsole compilation profiles are configured.

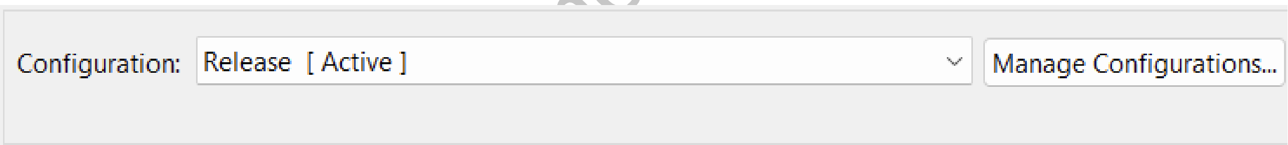
https://soceame.wordpress.com/2025/03/10/how-to-compile-and-debug-in-softconsole-for-smartfusion2/

Part one

First you have to have a project developed in SoftConsole, for this you need to have created a project in Libero where you configure the MSS.

Once you have the project in SoftConsole tested, debugged and you want to unify it with the Libero bitstream we can start generating it.

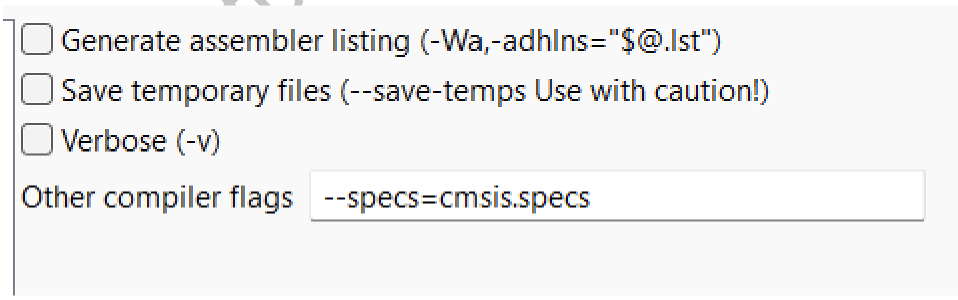
The first thing to do is change the debug compilation profile to ***Release***.



Configuration: Release [Active]

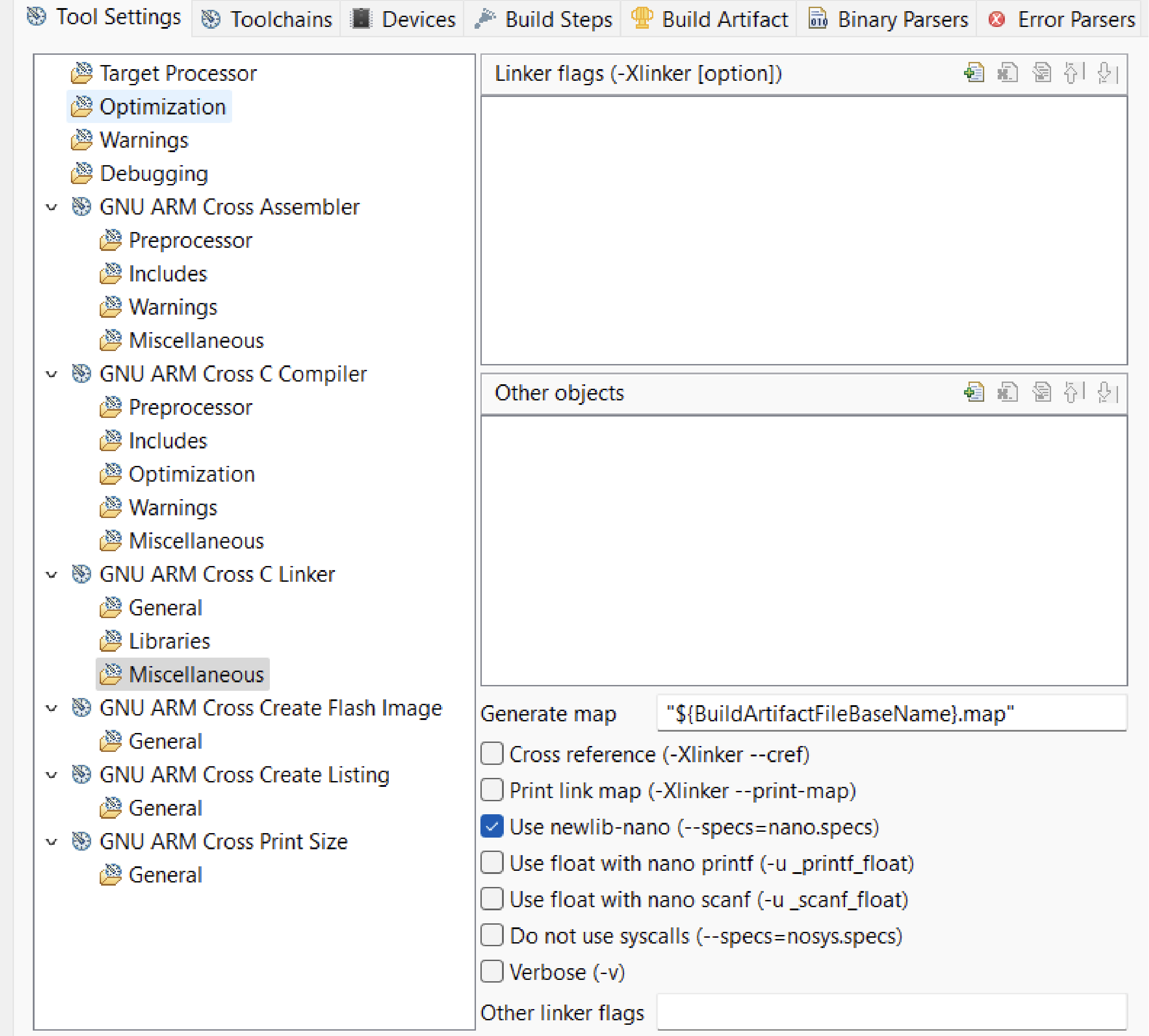
Then you have to configure the same thing that is configured for debugging except the *Script files.* This is done in the following way:

- set the *compiler flags*



☐ Generate assembler listing (-Wa,-adhlns="$@.lst")
☐ Save temporary files (--save-temps Use with caution!)
☐ Verbose (-v)
Other compiler flags --specs=cmsis.specs

- And the *–specs=nano.specs*.



Linker flags (-Xlinker [option])

Other objects

Generate map "${BuildArtifactFileBaseName}.map"

☐ Cross reference (-Xlinker --cref)

☐ Print link map (-Xlinker --print-map)

☑ Use newlib-nano (--specs=nano.specs)

☐ Use float with nano printf (-u _printf_float)

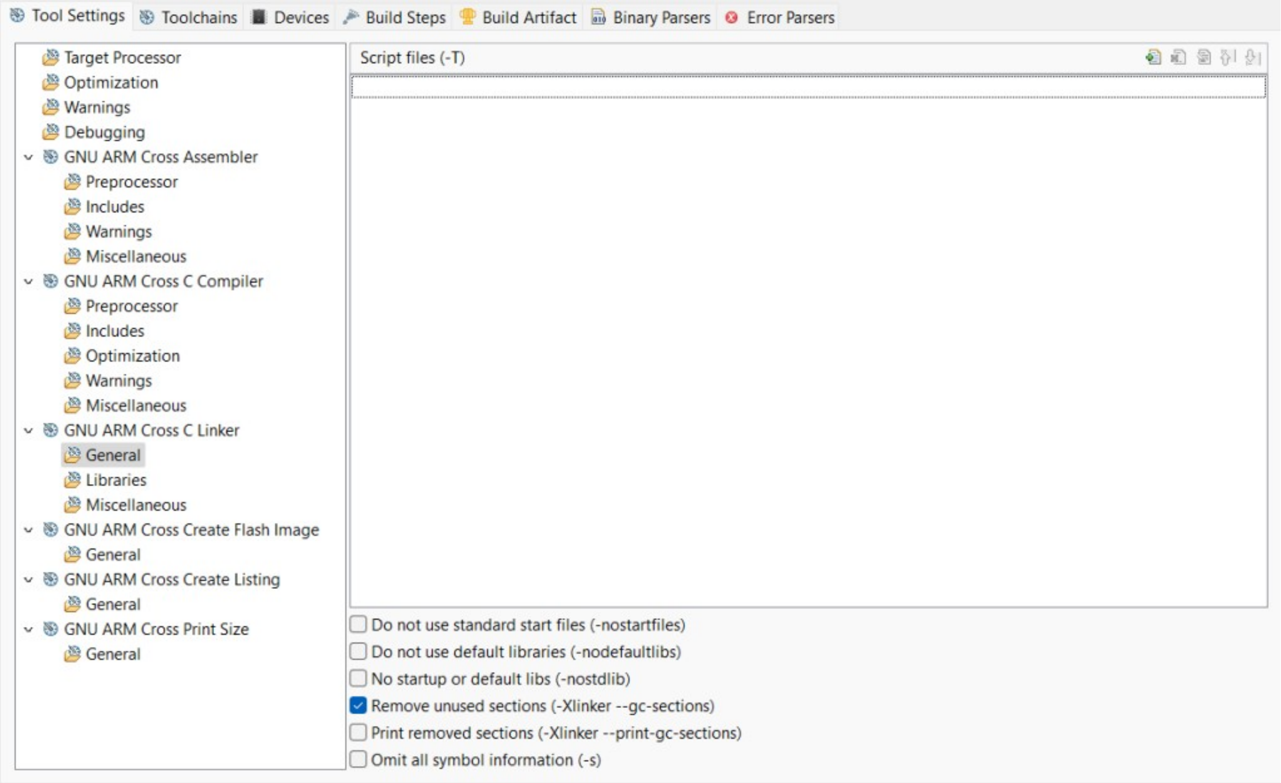
☐ Use float with nano scanf (-u _scanf_float)

☐ Do not use syscalls (--specs=nosys.specs)

☐ Verbose (-v)

Other linker flags

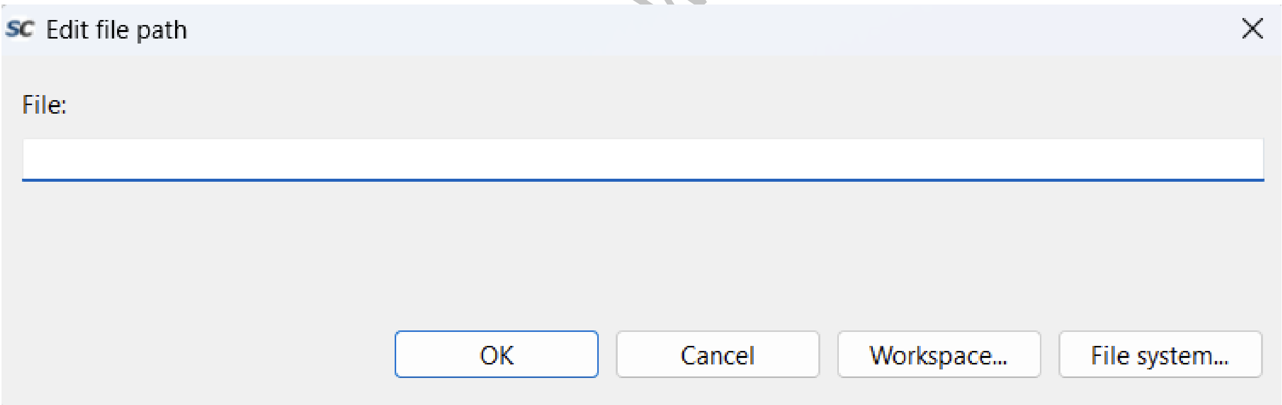
And what we have to change is the *Script File* in *GNU ARM Cross C Linker.*



Script files (-T)

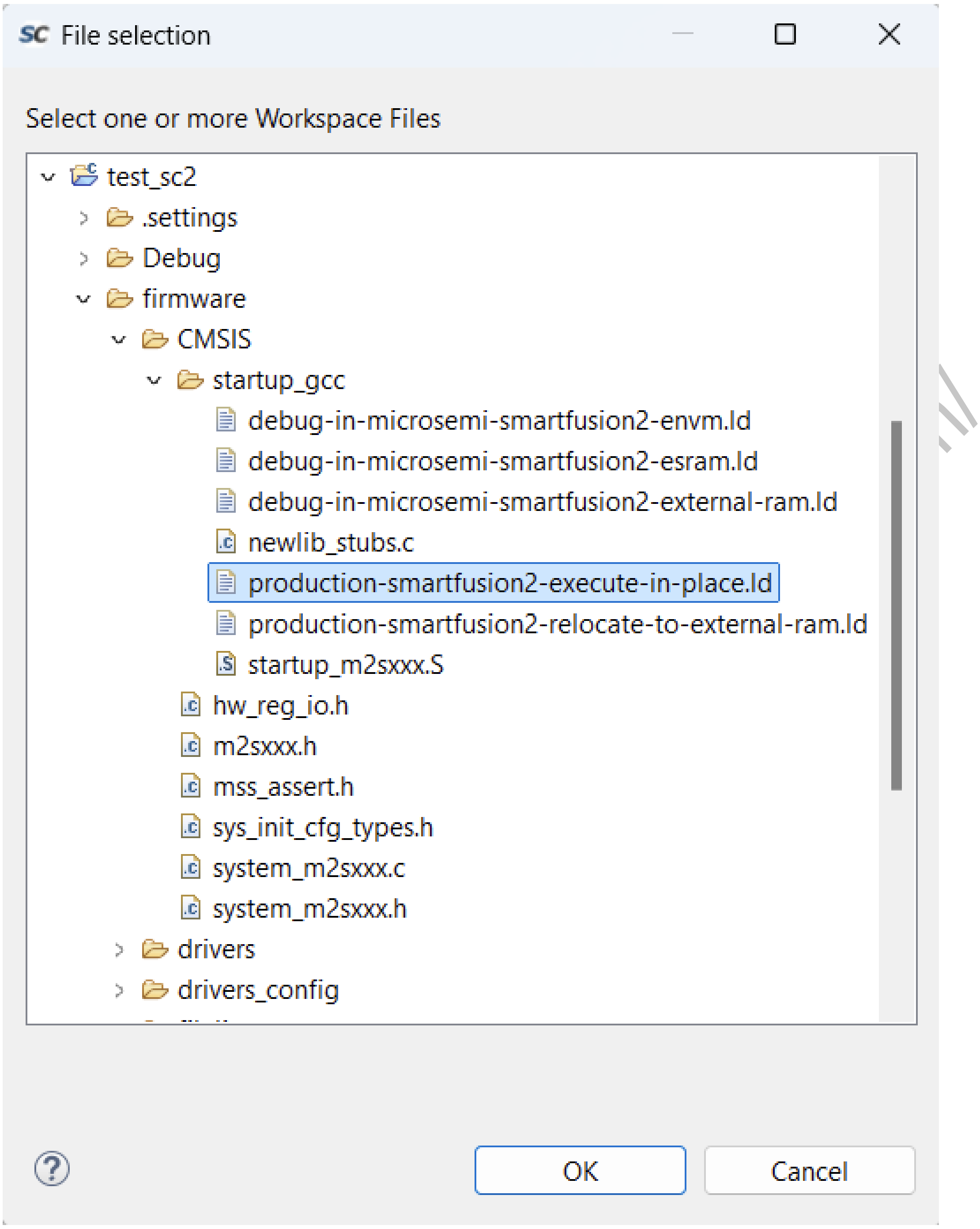
□ Do not use standard start files (-nostartfiles)
□ Do not use default libraries (-nodefaultlibs)
□ No startup or default libs (-nostdlib)
☑ Remove unused sections (-Xlinker --gc-sections)
□ Print removed sections (-Xlinker --print-gc-sections)
□ Omit all symbol information (-s)

We add a new *Script file* from the *Workspace*.



File:

And now we change the file from a *debug* file to a *production* file. Two production options appear, to save it internally or externally. In our case we chose internally (*production-smartfusion2-executable-in-place.ld*).

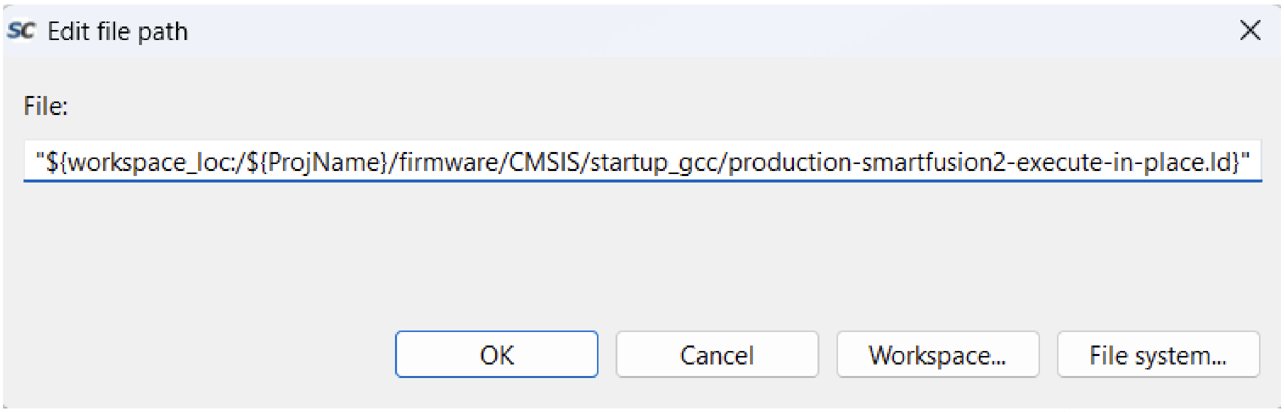


File selection

Select one or more Workspace Files

```
test_sc2
  .settings
  Debug
  firmware
    CMSIS
      startup_gcc
        debug-in-microsemi-smartfusion2-envm.ld
        debug-in-microsemi-smartfusion2-esram.ld
        debug-in-microsemi-smartfusion2-external-ram.ld
        newlib_stubs.c
        production-smartfusion2-execute-in-place.ld
        production-smartfusion2-relocate-to-external-ram.ld
        startup_m2sxxx.S
      hw_reg_io.h
      m2sxxx.h
      mss_assert.h
      sys_init_cfg_types.h
      system_m2sxxx.c
      system_m2sxxx.h
    drivers
    drivers_config
```

Now it appears selected.

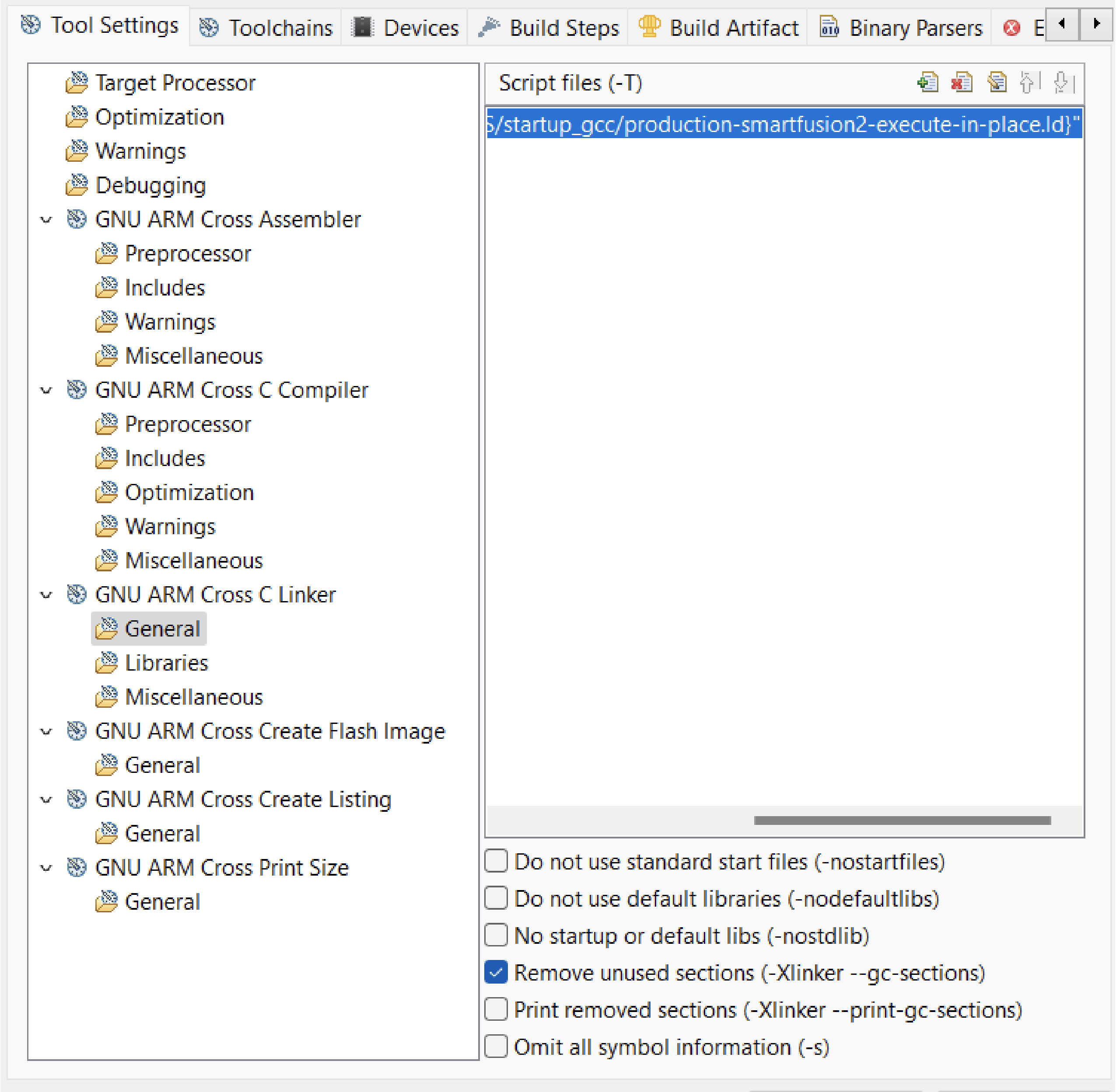


Edit file path

File:

"${workspace_loc:/${ProjName}/firmware/CMSIS/startup_gcc/production-smartfusion2-execute-in-place.ld}"

The file appears in the interface.



Script files (-T)

S/startup_gcc/production-smartfusion2-execute-in-place.ld}"

☐ Do not use standard start files (-nostartfiles)

☐ Do not use default libraries (-nodefaultlibs)

☐ No startup or default libs (-nostdlib)

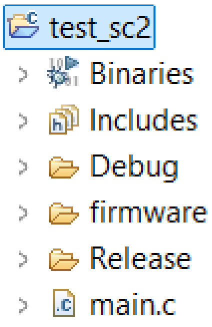
☑ Remove unused sections (-Xlinker --gc-sections)

☐ Print removed sections (-Xlinker --print-gc-sections)

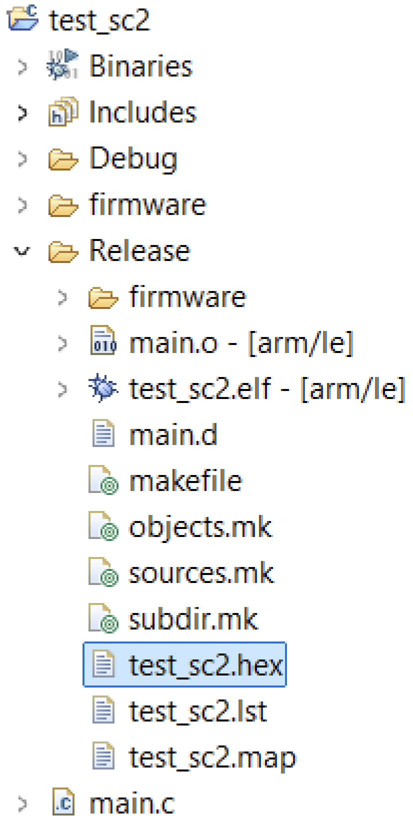
☐ Omit all symbol information (-s)

Well, once we have done all this, we can compile the project again.

When we do this, a new folder called *Release* appears.



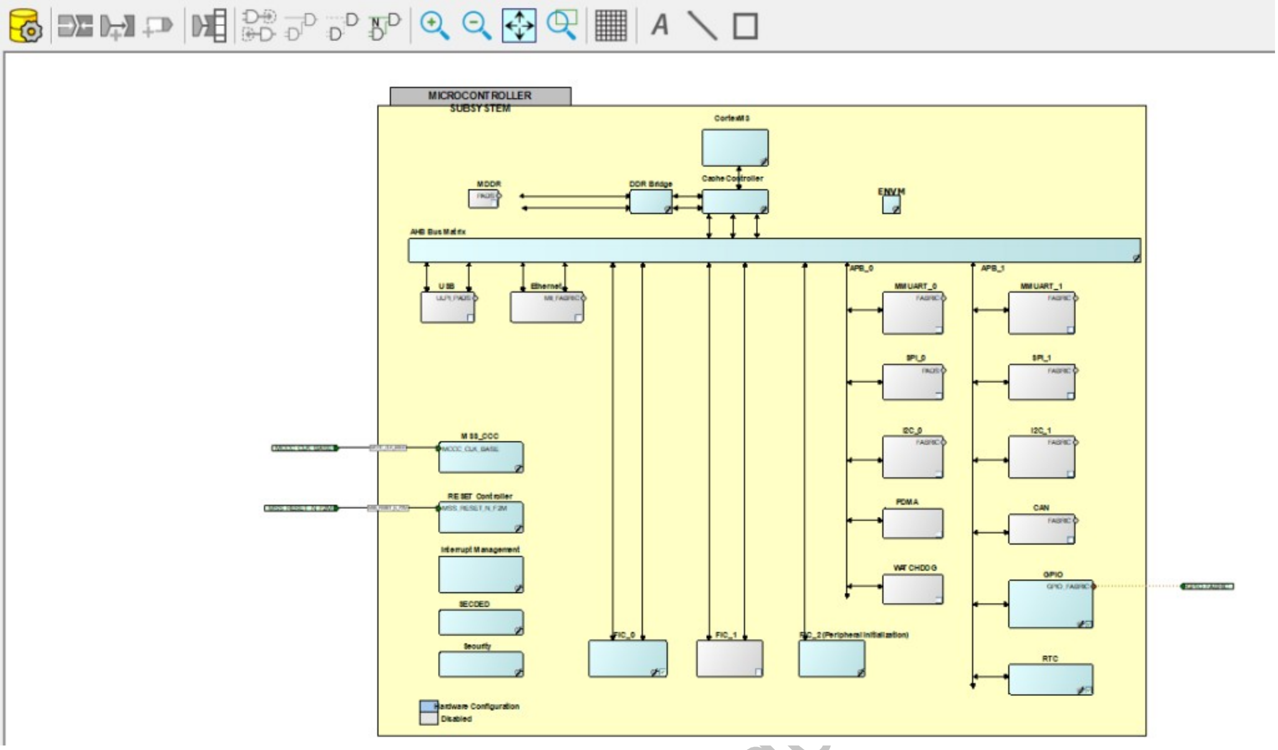
This folder contains the file we need to create the single bitstream, it is the **.hex** file. This file is the one we have to take to Libero.



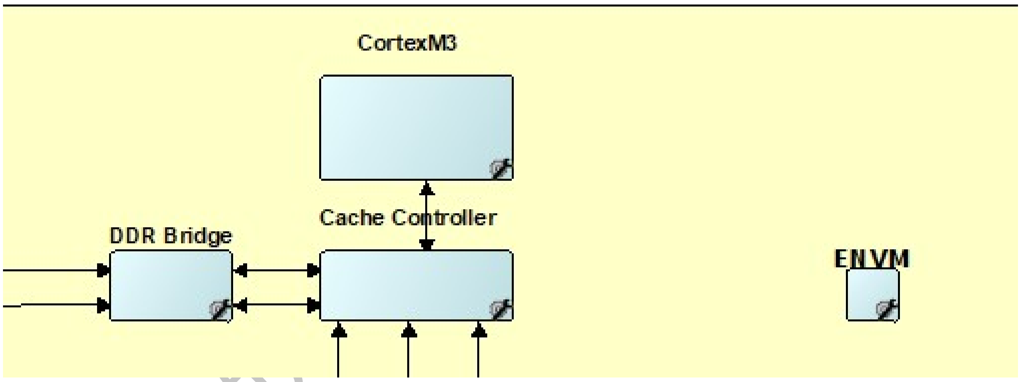
Second part

Once we have generated the .hex in Release, **NOT in Debug**, *because in Debug Libero does not allow the use of that .hex*, we return to the initial Libero project.

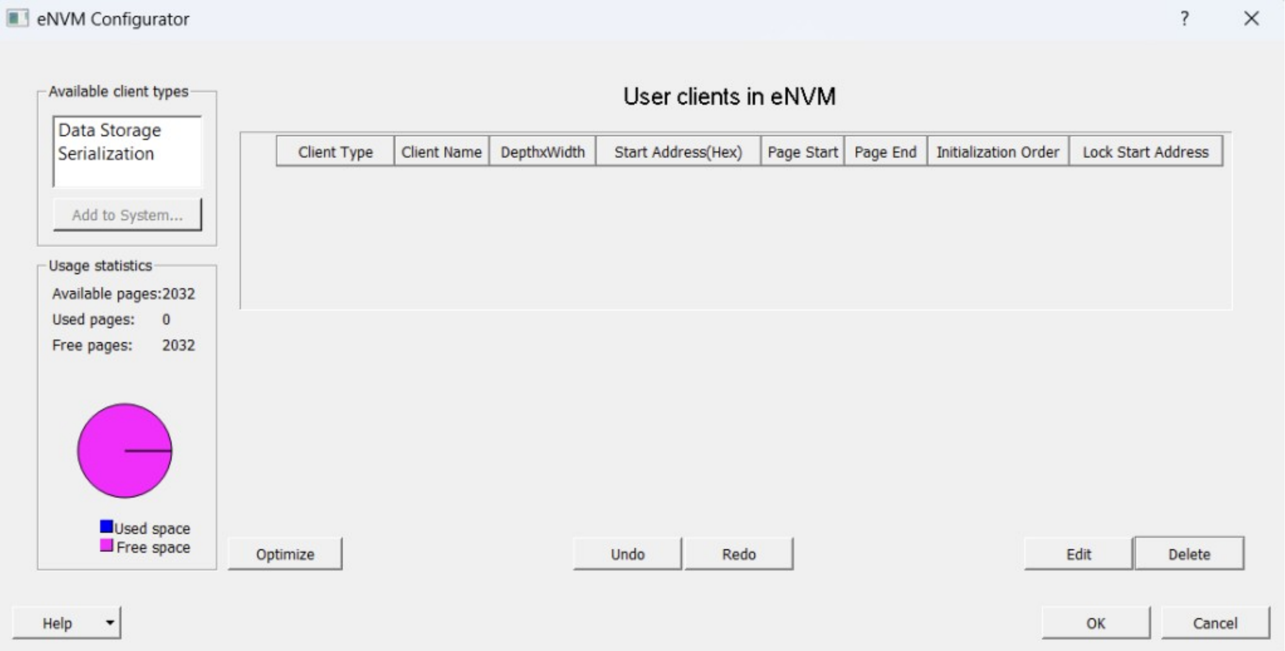
We return to this MSS configuration tab.



And we have to go to the **ENVM** option, which is the one that allows saving certain files in non-volatile memory within the MSS.



When we click on it, a tab like this one opens, empty.



eNVM Configurator

Available client types

Data Storage
Serialization

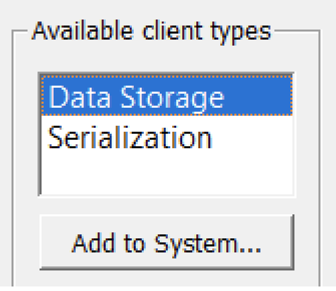
User clients in eNVM

Client Type	Client Name	DepthxWidth	Start Address(Hex)	Page Start	Page End	Initialization Order	Lock Start Address

Usage statistics

Available pages:2032
Used pages: 0
Free pages: 2032

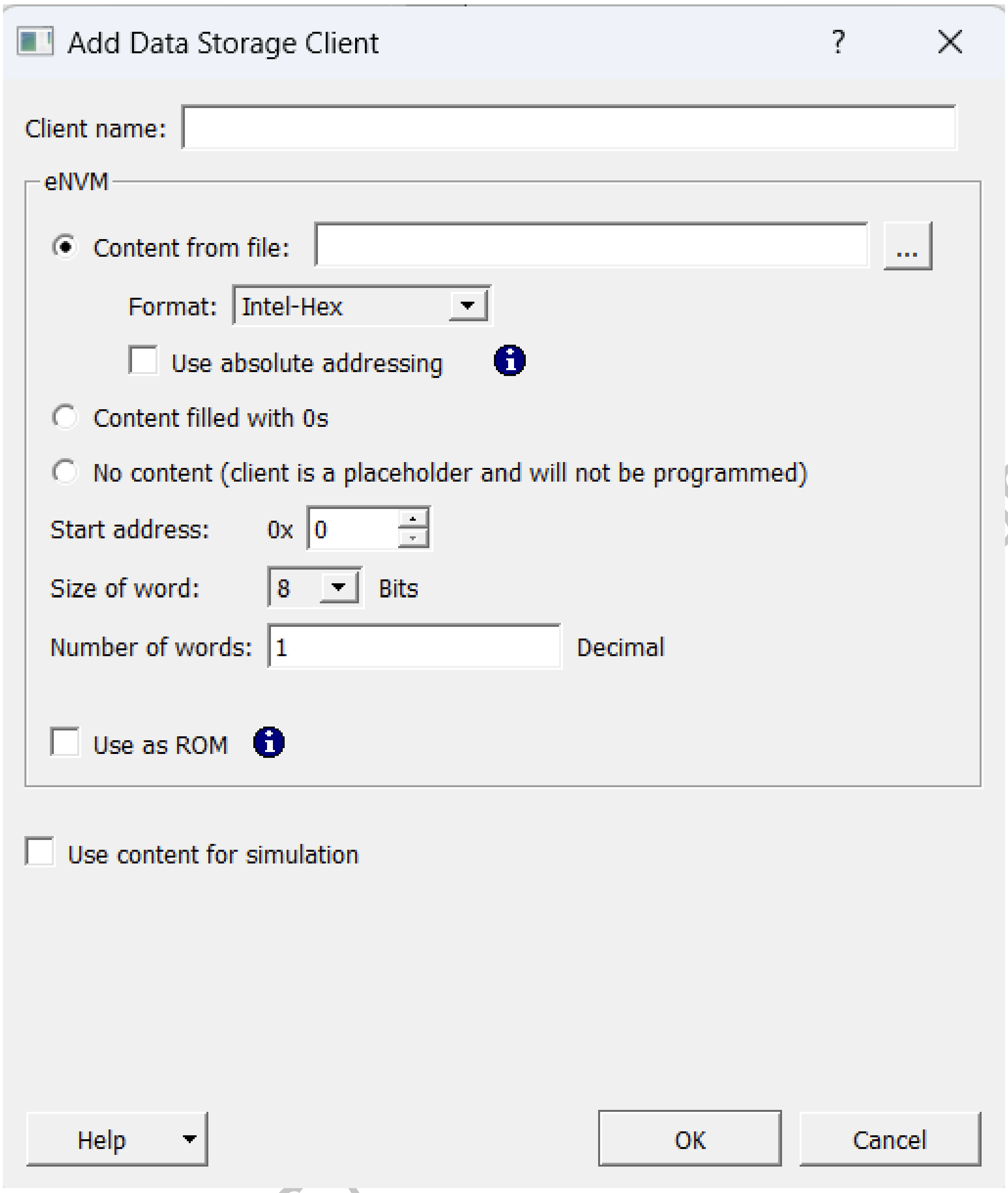
Now we click on *Data Storage* and *Add to System*.



Available client types

Data Storage
Serialization

A tab like the following one opens, which asks us for two things, to give a name to the memory structure and a valid .hex file, which is the one we generated previously.



Client name:

eNVM

● Content from file: ...

Format: Intel-Hex

□ Use absolute addressing

○ Content filled with 0s

○ No content (client is a placeholder and will not be programmed)

Start address: 0x 0

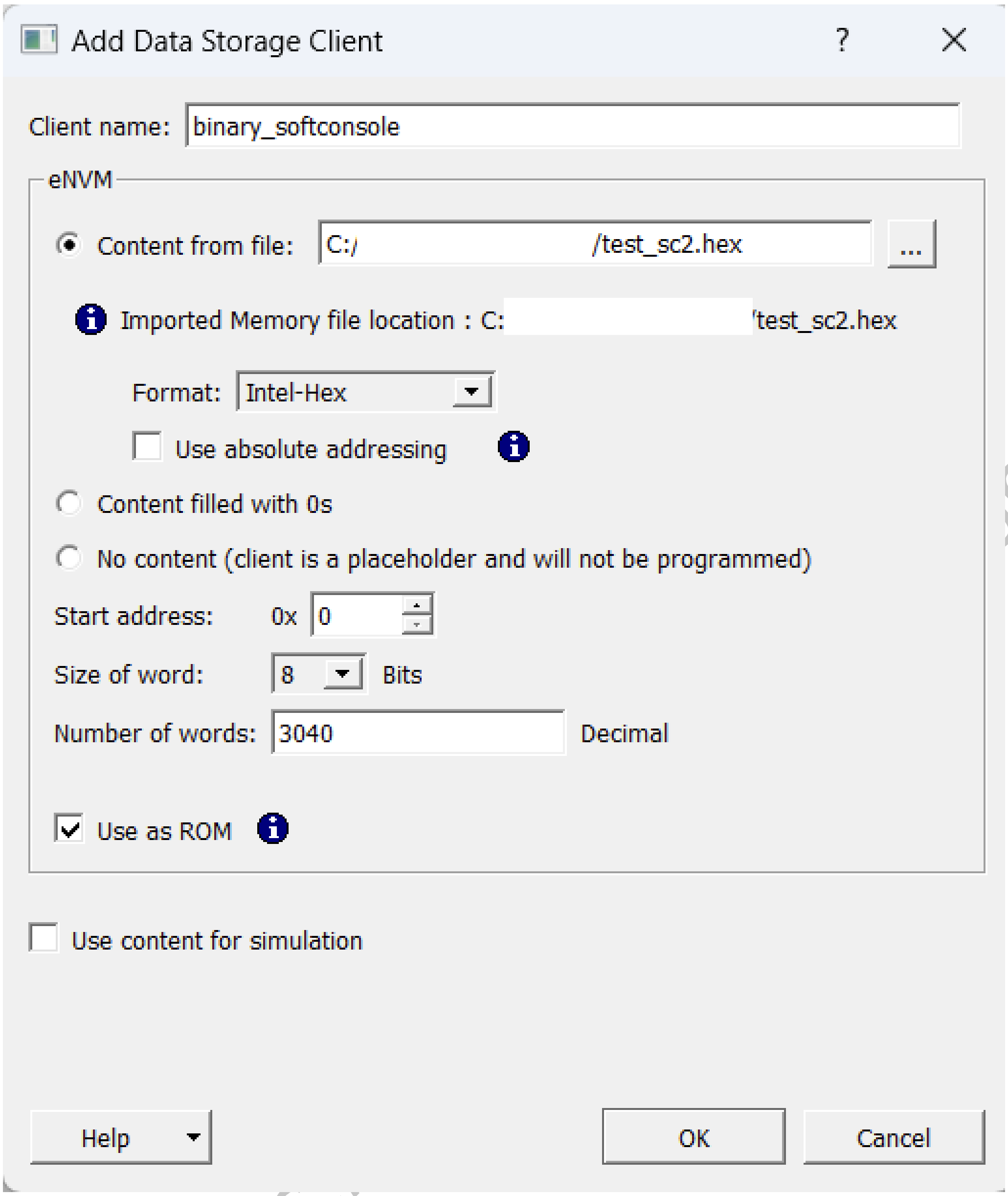
Size of word: 8 Bits

Number of words: 1 Decimal

□ Use as ROM

□ Use content for simulation

We give it a name and a valid production .hex. We also check the *Use as ROM* box.



Add Data Storage Client

Client name: binary_softconsole

eNVVM

● Content from file: C:/ [redacted] /test_sc2.hex

Imported Memory file location : C: [redacted] 'test_sc2.hex

Format: Intel-Hex

□ Use absolute addressing

○ Content filled with 0s

○ No content (client is a placeholder and will not be programmed)

Start address: 0x 0

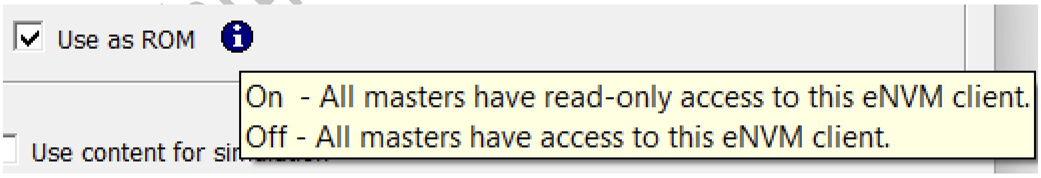
Size of word: 8 Bits

Number of words: 3040 Decimal

☑ Use as ROM

□ Use content for simulation

The *Use as ROM* is so that all the masters can read it, but not modify it.

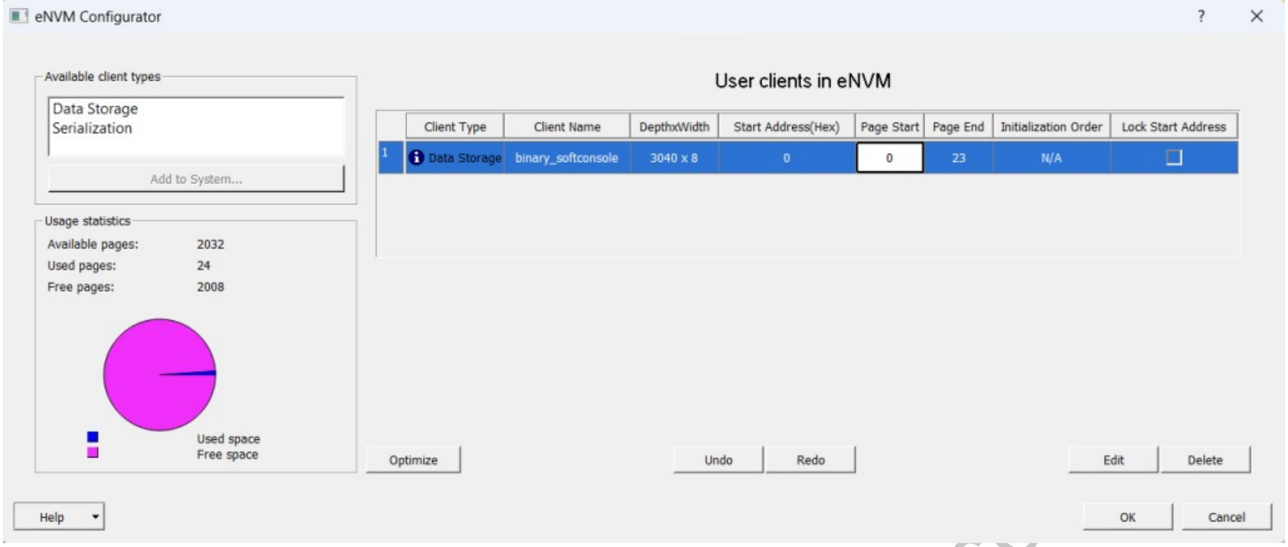


☑ Use as ROM

On - All masters have read-only access to this eNVM client.
Off - All masters have access to this eNVM client.

After adding the new memory layer, it appears in the tab.

https://soceame.wordpress.com/2025/03/10/how-to-create-a-single-bitstream-with-softconsole-for-a-smartfusion2/


eNVM Configurator

Available client types

Data Storage
Serialization

Usage statistics

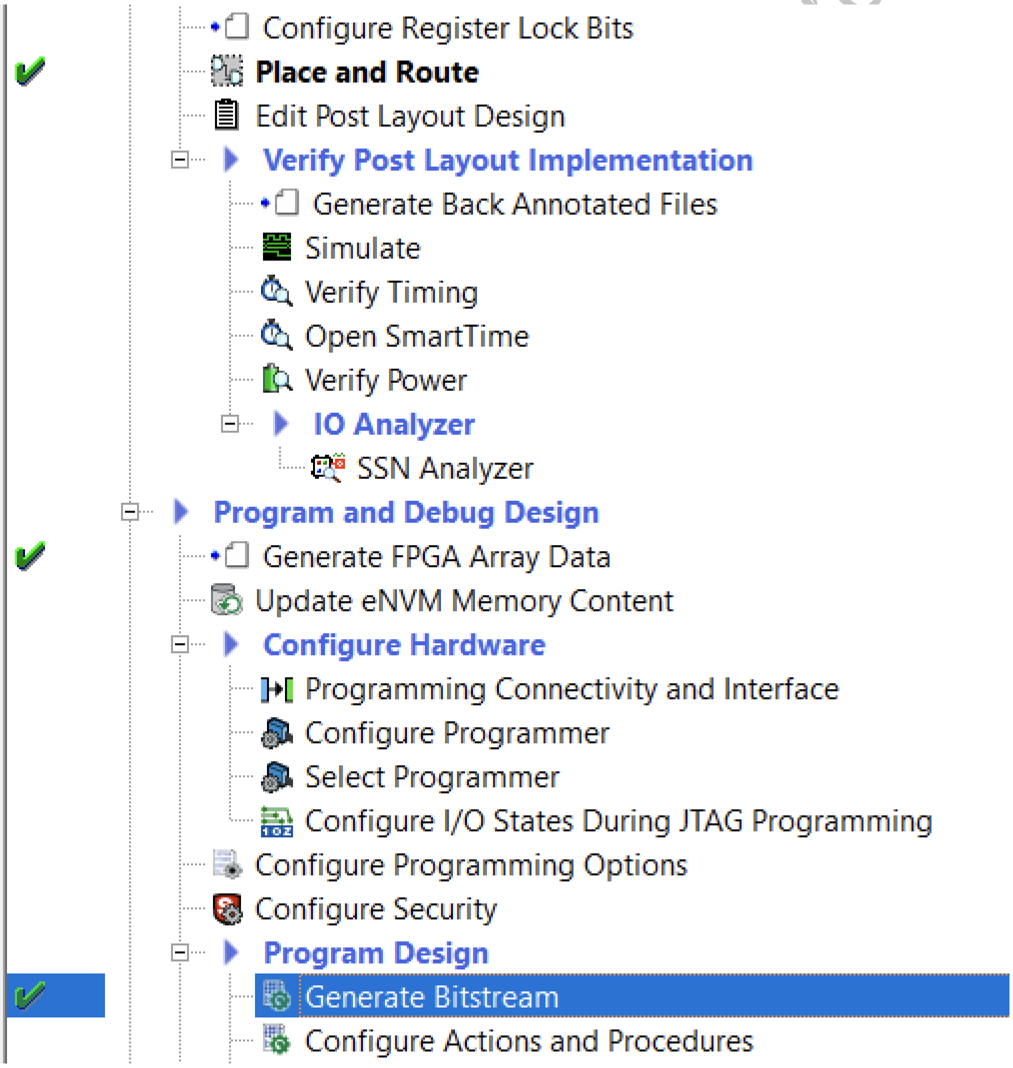
Available pages:	2032
Used pages:	24
Free pages:	2008

User clients in eNVM

	Client Type	Client Name	DepthxWidth	Start Address(Hex)	Page Start	Page End	Initialization Order	Lock Start Address
1	Data Storage	binary_softconsole	3040 x 8	0	0	23	N/A	□

Now all we have to do is click on the yellow gear icons and the *Build Hierarchy* button.

Once we have the project rebuilt and ready, we synthesize it and generate the bitstream.


Configure Register Lock Bits
✓ Place and Route
Edit Post Layout Design
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 Interface
Configure Programmer
Select Programmer
Configure I/O States During JTAG Programming
Configure Programming Options
Configure Security
Program Design
✓ Generate Bitstream
Configure Actions and Procedures

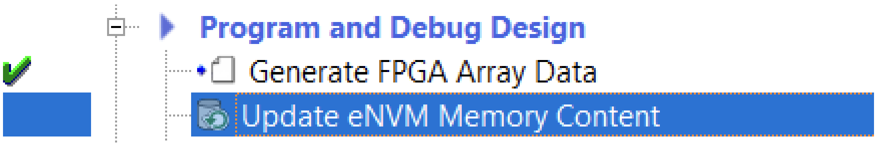
https://soceame.wordpress.com/2025/03/10/how-to-create-a-single-bitstream-with-softconsole-for-a-smartfusion2/

This bitstream that we have generated contains the .hex that we have created. Now we just need to export it to record it with FlashPro Express.

In this post I leave you how to install and use FlashPro Express.

https://soceame.wordpress.com/2025/03/09/flashpro-express-tutorial-installing-and-running/

NOTE: the *Update eNVM Memory Content* option takes us to the tab where we have entered the SoftConsole binary (.hex).


Program and Debug Design
Generate FPGA Array Data
Update eNVM Memory Content