How to compile and debug in SoftConsole for SmartFusion2

Created by: David Rubio G.

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

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

GitHub: https://github.com/DRubioG

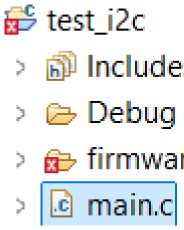
Last modification date: 10/03/25

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

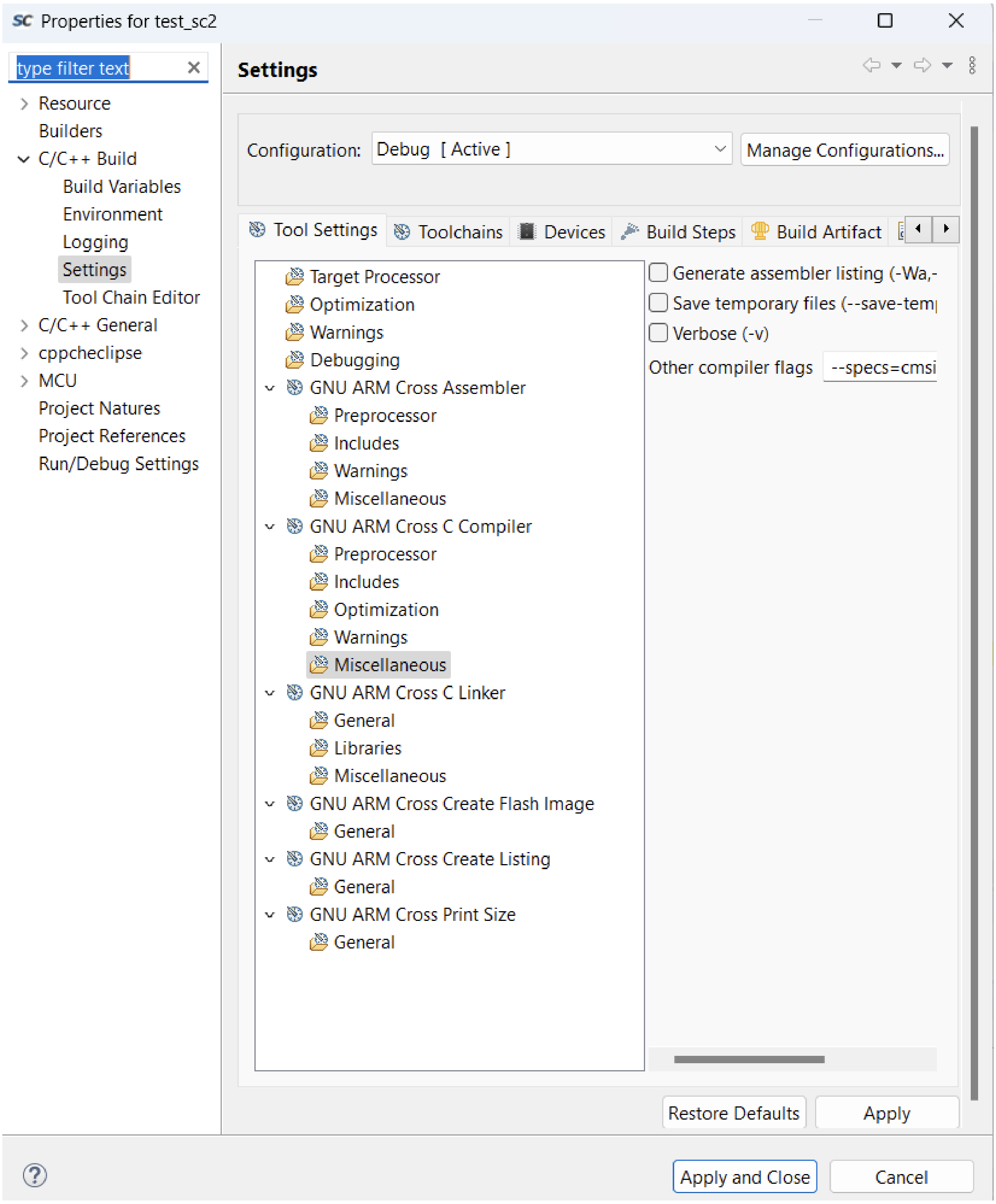
This entry is created because every time you create a new project in SoftConsole for SmartFusion2 you always have to configure the same parameters. So, this entry is a compilation.

Compilation

A typical problem with SoftConsole is creating a project, and when you go to compile it, it generates a compilation error. This is because you have not configured 3 compilation directives that are always the same.



To do this, right-click on the project and click *Properties*. In the tab that opens, go to *C/C++ Build* and *Settings*.



Configuration: Debug [Active]

☐ Generate assembler listing (-Wa,-

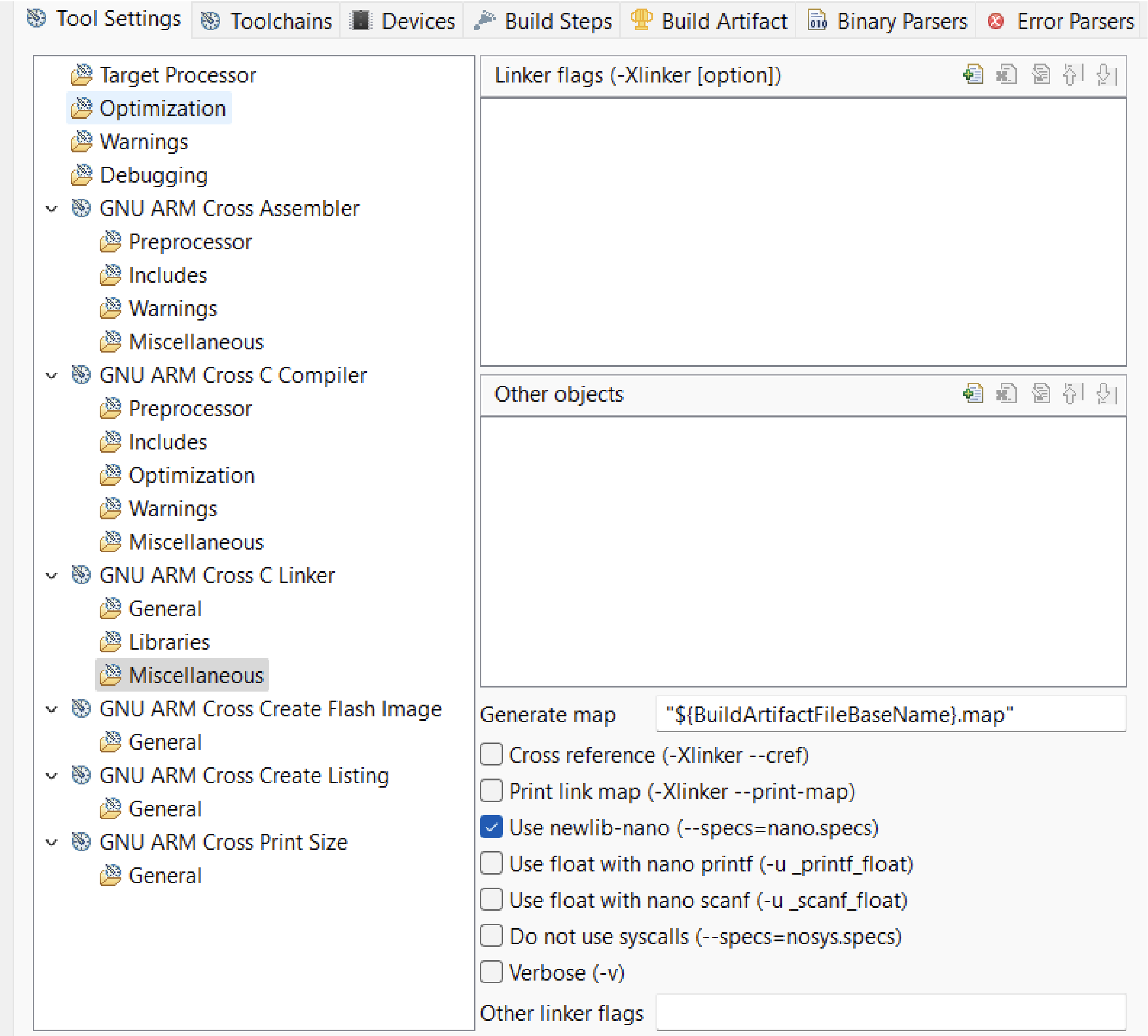
☐ Save temporary files (--save-tem|

☐ Verbose (-v)

Other compiler flags --specs=cmsi

Now we configure the following options.

- In *GNU ARM Cross C Linker*, we check the box «*Use newlib-nano (–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)

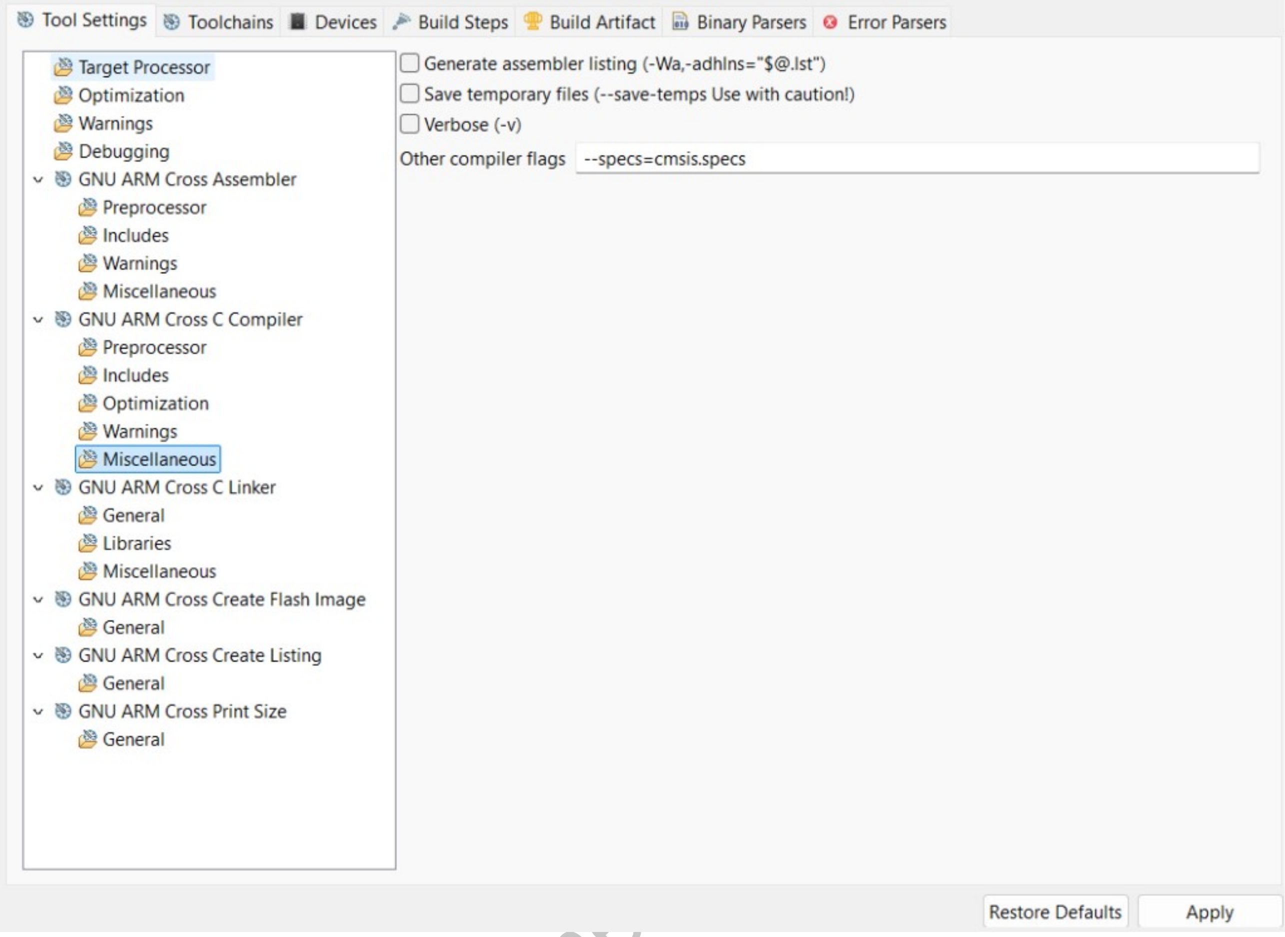
☐ Use float with nano scanf (-u _scanf_float)

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

☐ Verbose (-v)

Other linker flags

- In *GNU ARM Cross C Compiler*, we add the flag «*–specs=cmsis.specs*«.



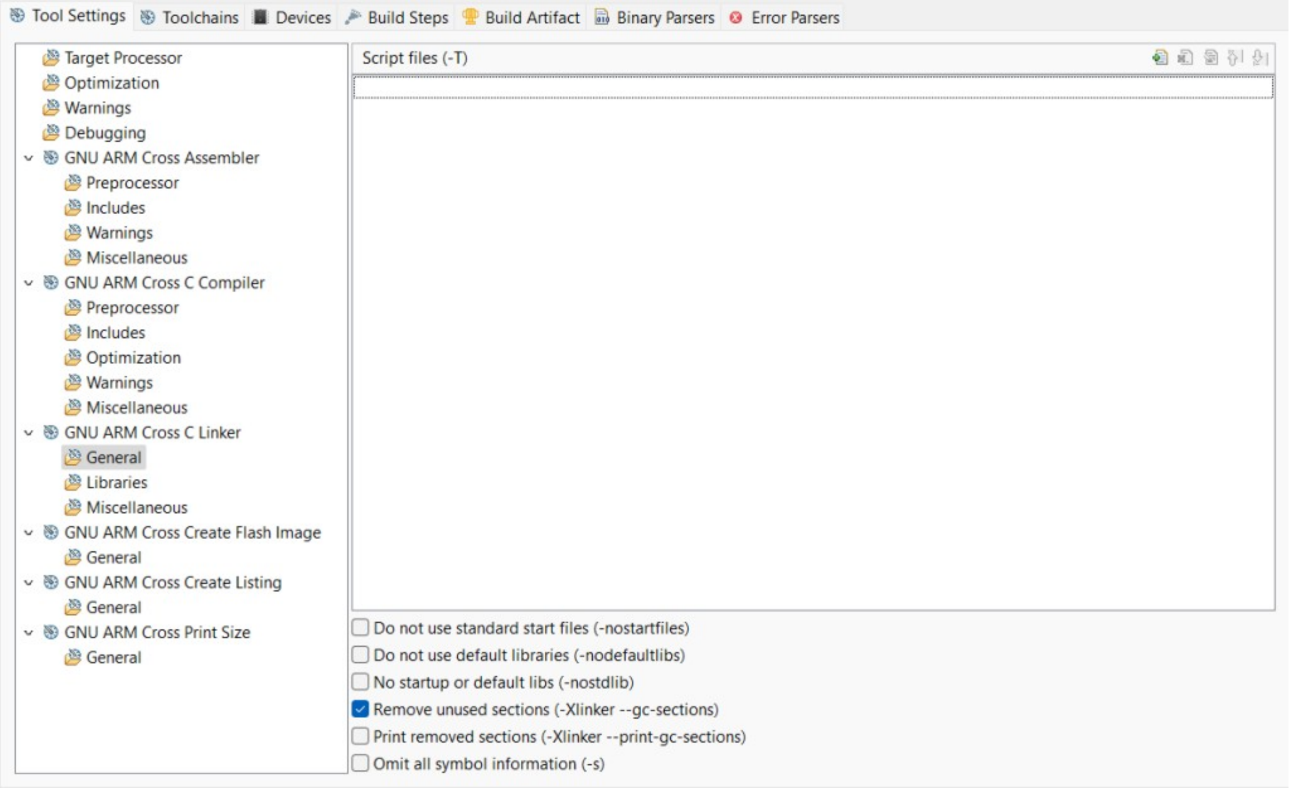
□ Generate assembler listing (-Wa,-adhlns="$@.lst")

□ Save temporary files (--save-temps Use with caution!)

□ Verbose (-v)

Other compiler flags: --specs=cmsis.specs

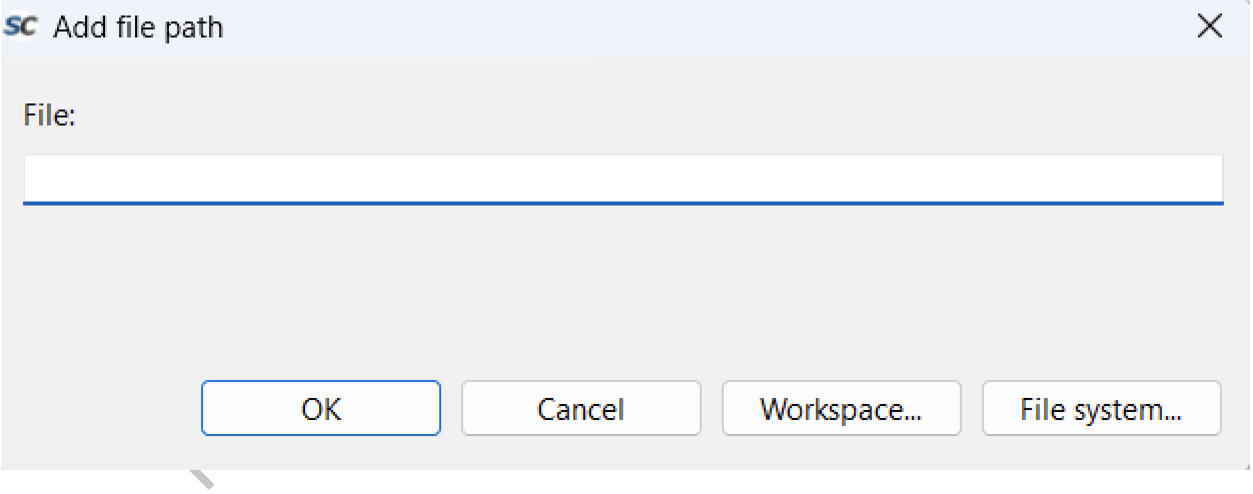
- In *GNU ARM Cross C Linker* we have to add how we are going to record the SoC memory with the SoftConsole executable. This is because the SoC is made with Flash memory, so you have to choose whether you want to save or delete the SoftConsole executable.
  To add the option, click on the ‘+’.



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)

Then, a tab like this opens, then, click on *Workspace…*.

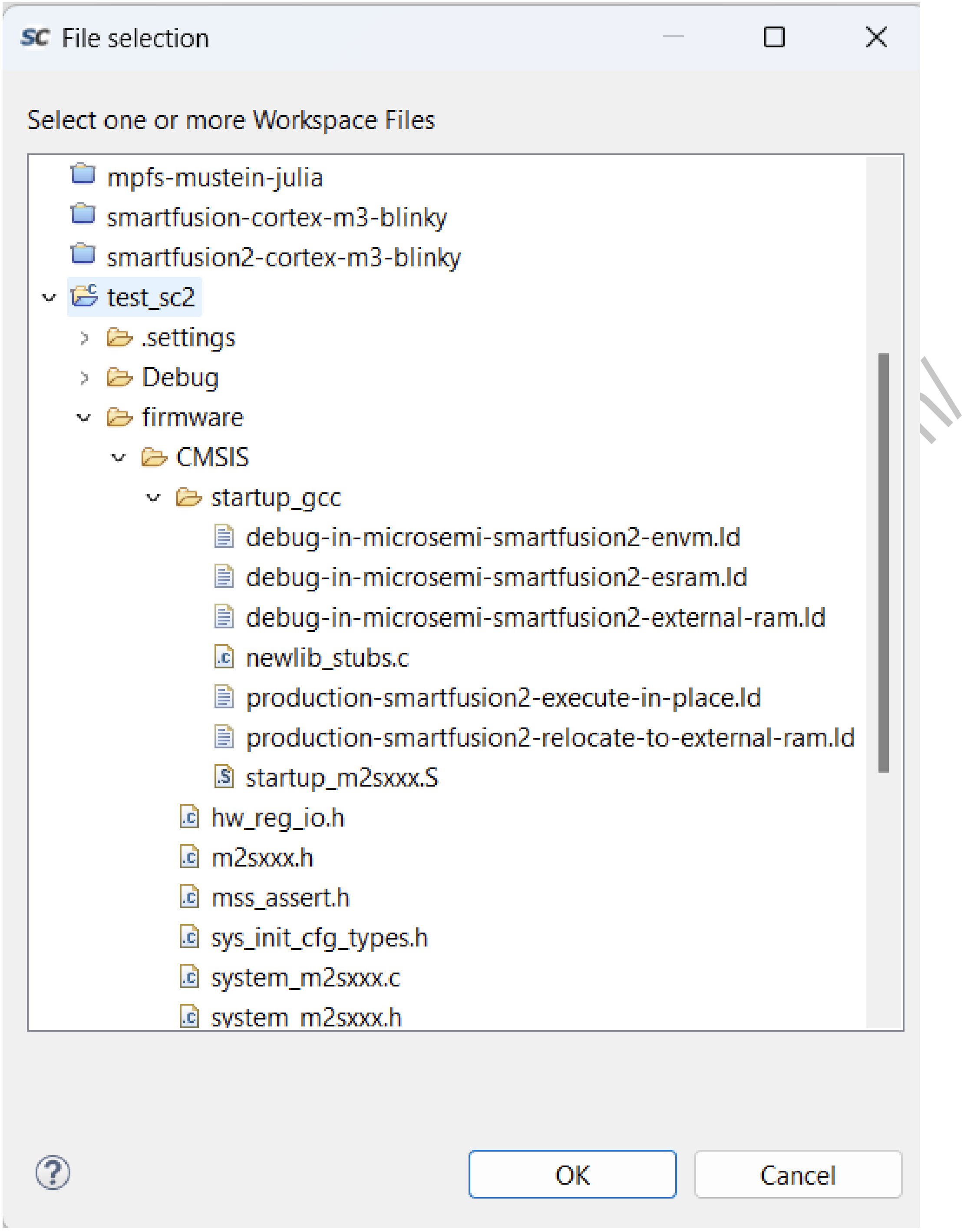

Add file path

File:

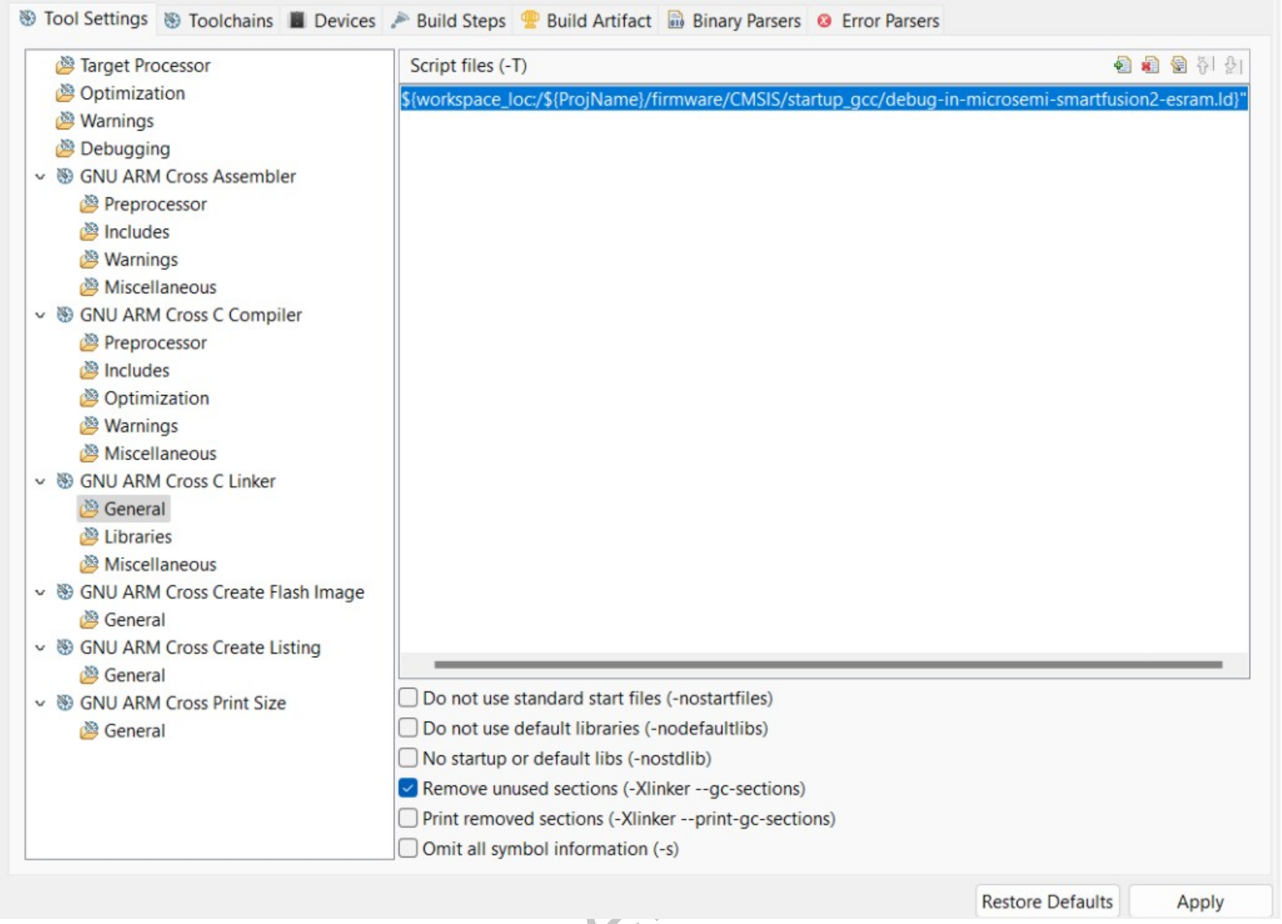
This opens a tab like the following, then, we have to go to the project drivers folder, and look in *CMSIS* > *startup_gcc*. Here you will see some .ld type files, these are the files that choose how the SoftConsole binary is saved.

We mainly focus on two:

- ***debug-in-microsemi-smartfusion2-envm.ld*** : this is the one that saves the SoftConsole binary in the internal non-volatile memory. Because the program is saved.
- ***debug-in-microsemi-smartfusion2-esram.ld*** : this is the one that writes the binary to SRAM memory, so resetting or cutting off the power will make the program disappear.


File selection
Select one or more Workspace Files
mpfs-mustein-julia
smartfusion-cortex-m3-blinky
smartfusion2-cortex-m3-blinky
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
OK
Cancel

Once selected, it appears in the next tab.



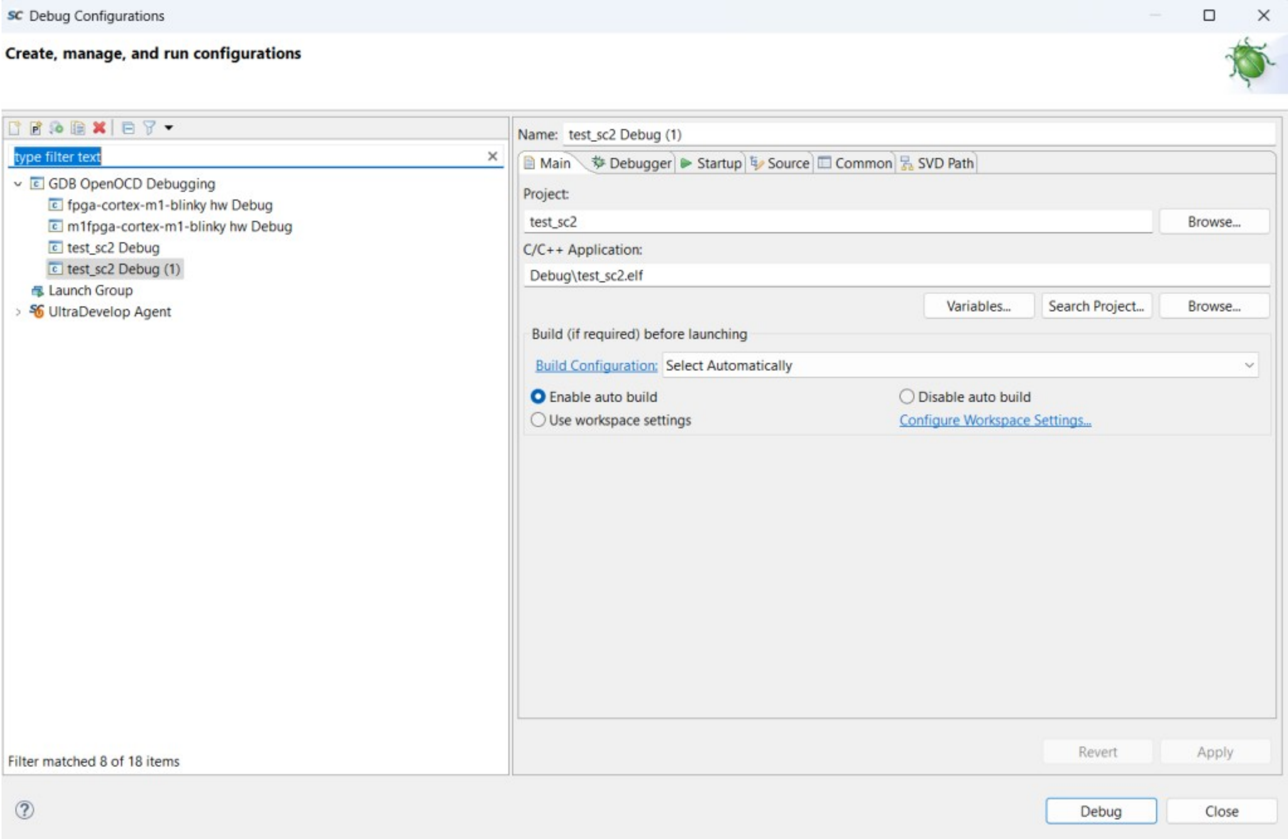
Script files (-T)

"${workspace_loc:/${ProjName}/firmware/CMSIS/startup_gcc/debug-in-microsemi-smartfusion2-esram.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)

Debugging

For debugging, the first thing you have to create is a new debug profile that includes the .elf you want to debug.



Name: test_sc2 Debug (1)

Project: test_sc2

C/C++ Application: Debug\test_sc2.elf

Build (if required) before launching

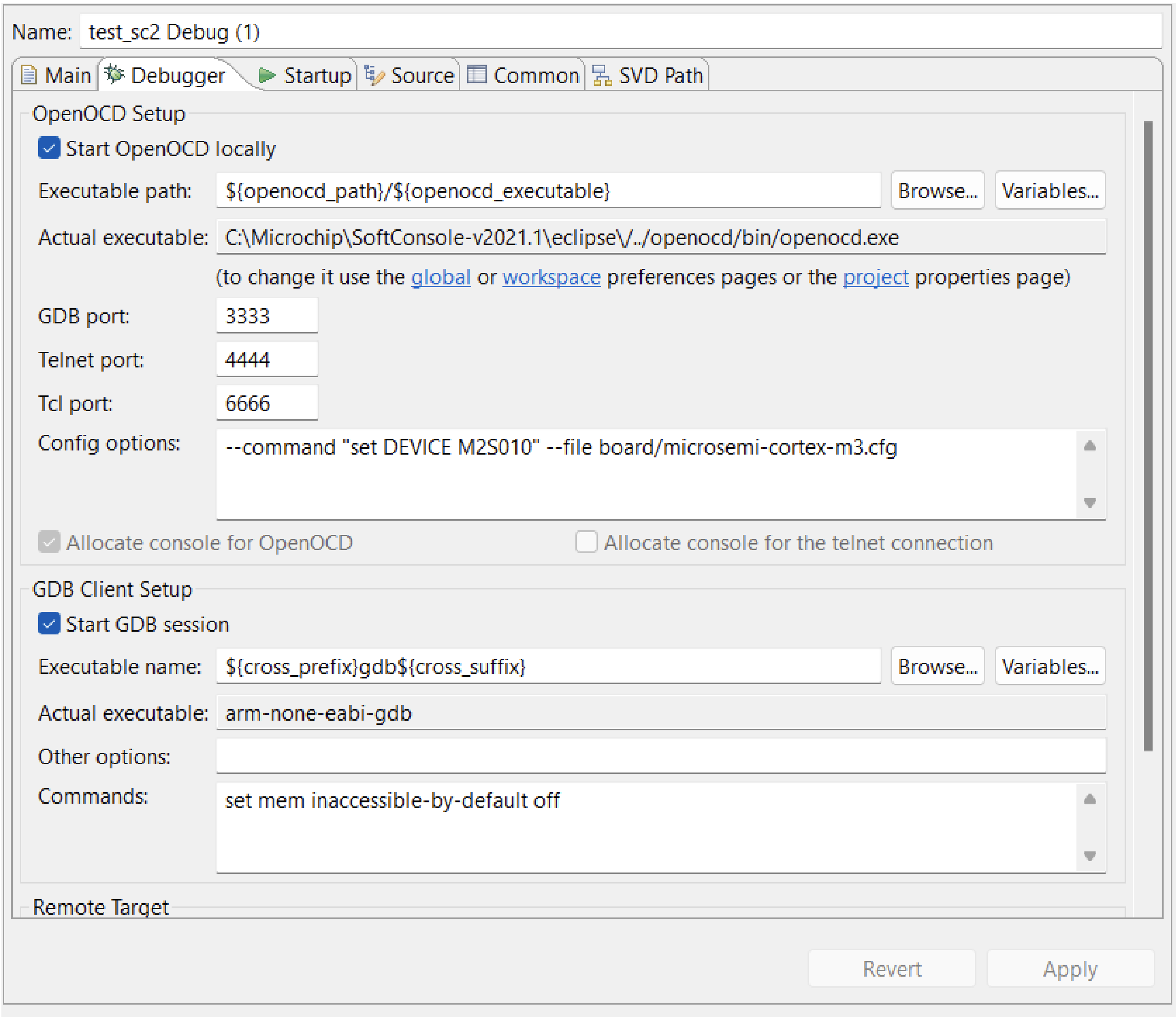
Build Configuration: Select Automatically

● Enable auto build ○ Disable auto build

○ Use workspace settings

In the debugger tab, you have to fill in the *OpenOCD* Setup box with the following settings: ***–command «set DEVICE <Microsemi device>» –file board/microsemi-cortex-m3.cfg***
(in my case, since I'm going to program an M2S010, I put it as a value).

And in the GDB session box, I leave the following command: ***set mem inaccessible-by-default off***



Name: test_sc2 Debug (1)

OpenOCD Setup

☑ Start OpenOCD locally

Executable path: ${openocd_path}/${openocd_executable}

Actual executable: C:\Microchip\SoftConsole-v2021.1\eclipse\/../openocd/bin/openocd.exe

(to change it use the global or workspace preferences pages or the project properties page)

GDB port: 3333

Telnet port: 4444

Tcl port: 6666

Config options: --command "set DEVICE M2S010" --file board/microsemi-cortex-m3.cfg

☑ Allocate console for OpenOCD ☐ Allocate console for the telnet connection

GDB Client Setup

☑ Start GDB session

Executable name: ${cross_prefix}gdb${cross_suffix}

Actual executable: arm-none-eabi-gdb

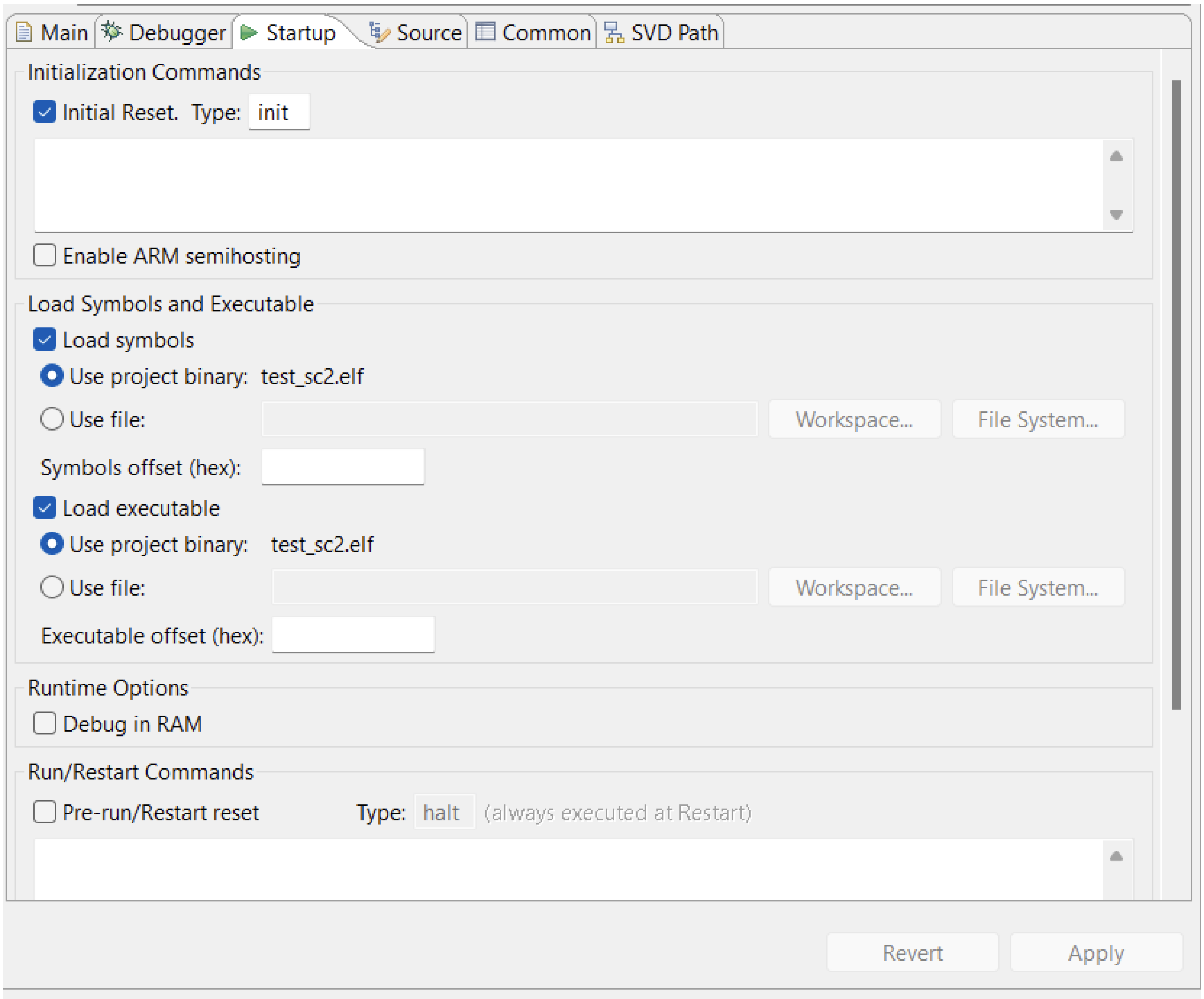
Other options:

Commands: set mem inaccessible-by-default off

Remote Target

Finally, in the Startup tab, you have to have the *Pre-run/Restart reset* box unchecked.

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



Initialization Commands

☑ Initial Reset. Type: init

☐ Enable ARM semihosting

Load Symbols and Executable

☑ Load symbols

● Use project binary: test_sc2.elf

○ Use file:

Symbols offset (hex):

☑ Load executable

● Use project binary: test_sc2.elf

○ Use file:

Executable offset (hex):

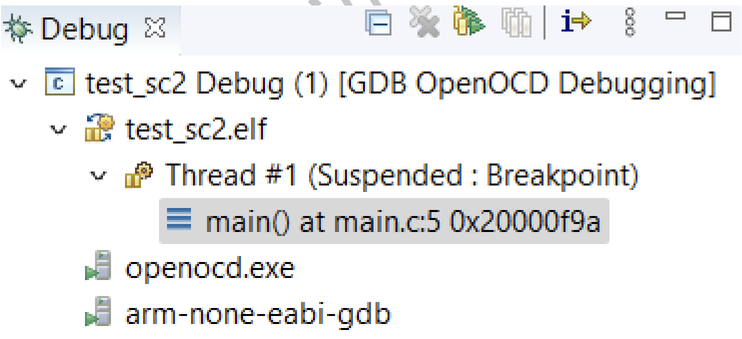
Runtime Options

☐ Debug in RAM

Run/Restart Commands

☐ Pre-run/Restart reset Type: halt (always executed at Restart)

Now you can launch debugging. To know what you are debugging you can see it in the *Debug* tab of the SoftConsole.



test_sc2 Debug (1) [GDB OpenOCD Debugging]

test_sc2.elf

Thread #1 (Suspended : Breakpoint)

main() at main.c:5 0x20000f9a

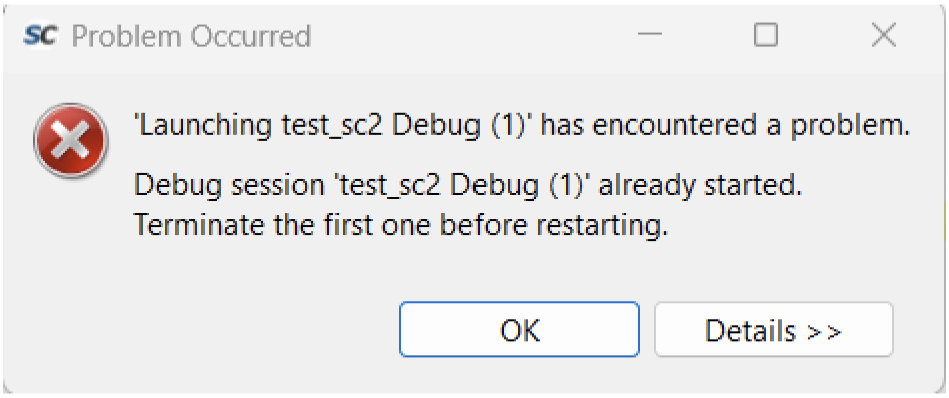
openocd.exe

arm-none-eabi-gdb

Final NOTE

If you try to launch a new debugging while the board is debugging another program, an error will appear and you will be taken out of debugging, having to launch the debugging again.

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



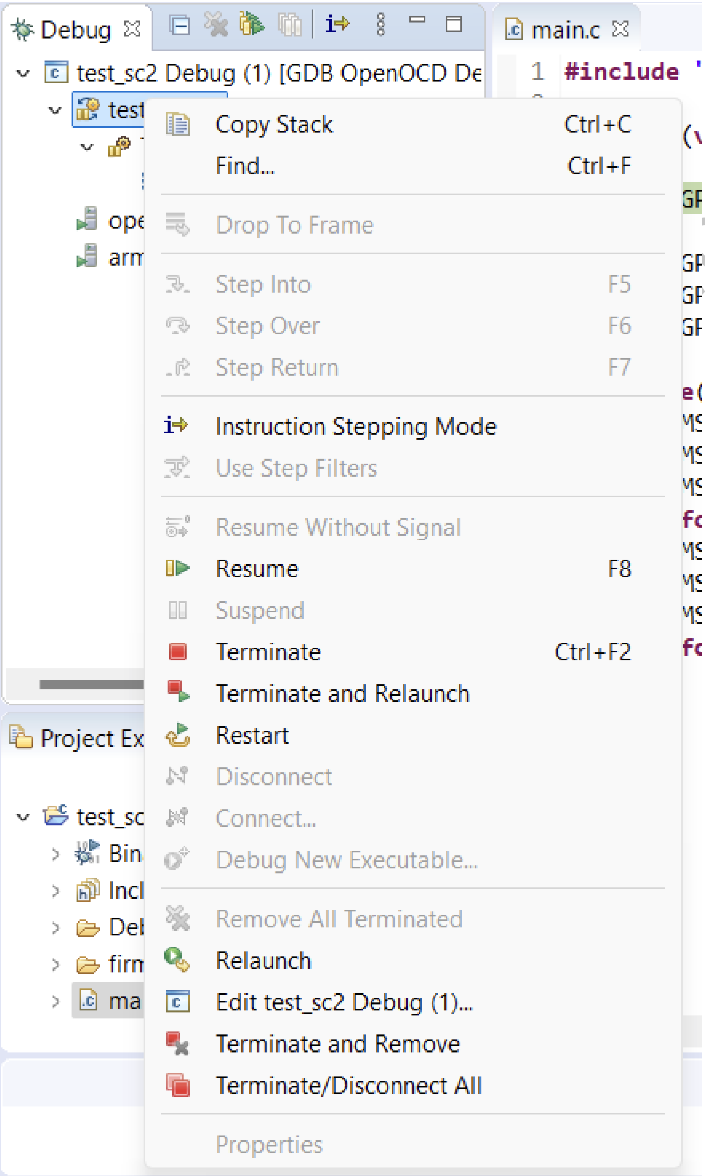
Problem Occurred

'Launching test_sc2 Debug (1)' has encountered a problem.

Debug session 'test_sc2 Debug (1)' already started. Terminate the first one before restarting.

To avoid this error, what you can do is right click on the debugging that is running and click *Terminate and Relaunch*. This will not take you out of debugging.

NOTE: remember to compile the program before launching it again in debugging.



Debug
main.c
test_sc2 Debug (1) [GDB OpenOCD De
1 #include
test
ope
arm
Project Ex
test_sc
Bin
Incl
Del
firm
ma
Copy Stack Ctrl+C
Find... Ctrl+F
Drop To Frame
Step Into F5
Step Over F6
Step Return F7
Instruction Stepping Mode
Use Step Filters
Resume Without Signal
Resume F8
Suspend
Terminate Ctrl+F2
Terminate and Relaunch
Restart
Disconnect
Connect...
Debug New Executable...
Remove All Terminated
Relaunch
Edit test_sc2 Debug (1)...
Terminate and Remove
Terminate/Disconnect All
Properties