S32K344 MCAL Integrate with S32DS

Stephen Du

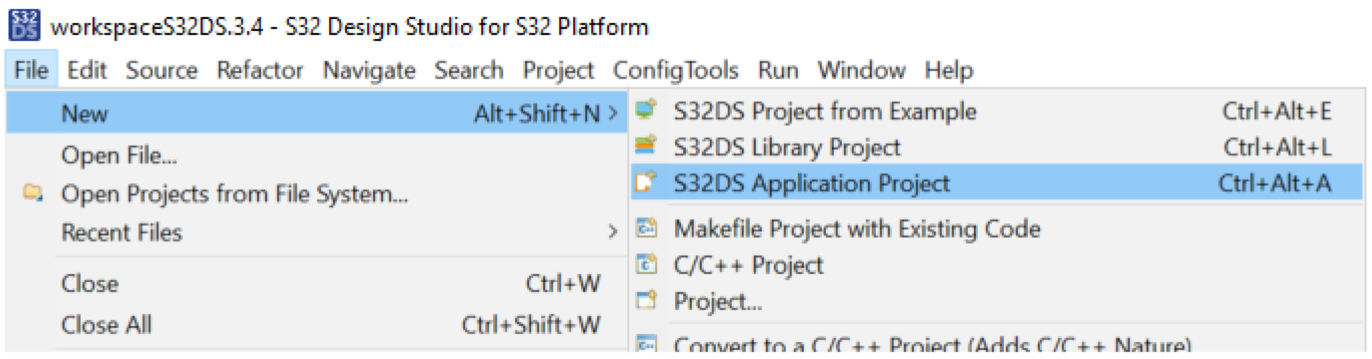
Foreword

This document will not guide you how to configure a MCAL project with EB Tresos. The pre-condition is that you already complete the configuration of your MCAL project and there is no error with code generation. (If you want to know how to work with EB Tresos to create a project, please refer other material)

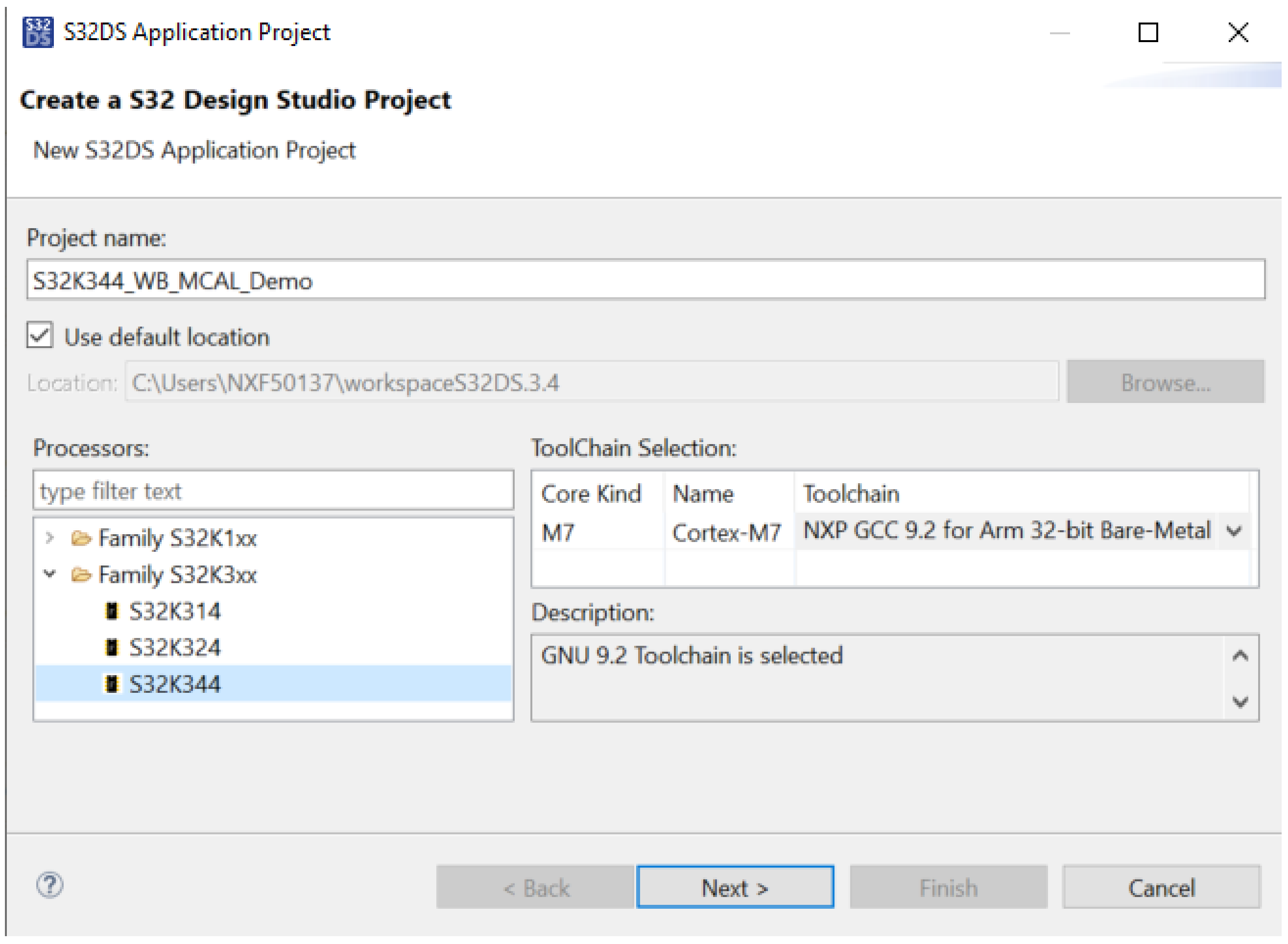
And if your project is using makefile or something similar with that to construct your target files, you can ignore this document. This document is for who wants to use S32DS to compiling and debug.

1. New Project

1. Open S32DS V3.4 or later version. Then `File` -> `New` -> `S32DS Application Project`.


workspaceS32DS.3.4 - S32 Design Studio for S32 Platform
File Edit Source Refactor Navigate Search Project ConfigTools Run Window Help
New Alt+Shift+N
Open File...
Open Projects from File System...
Recent Files
Close Ctrl+W
Close All Ctrl+Shift+W
S32DS Project from Example Ctrl+Alt+E
S32DS Library Project Ctrl+Alt+L
S32DS Application Project Ctrl+Alt+A
Makefile Project with Existing Code
C/C++ Project
Project...
Convert to a C/C++ Project (Adds C/C++ Nature)

2. Input you project name and select the chip type.


S32DS Application Project

Create a S32 Design Studio Project

New S32DS Application Project

Project name: S32K344_WB_MCAL_Demo

[✓] Use default location

Location: C:\Users\NXF50137\workspaceS32DS.3.4

Processors:

ToolChain Selection:

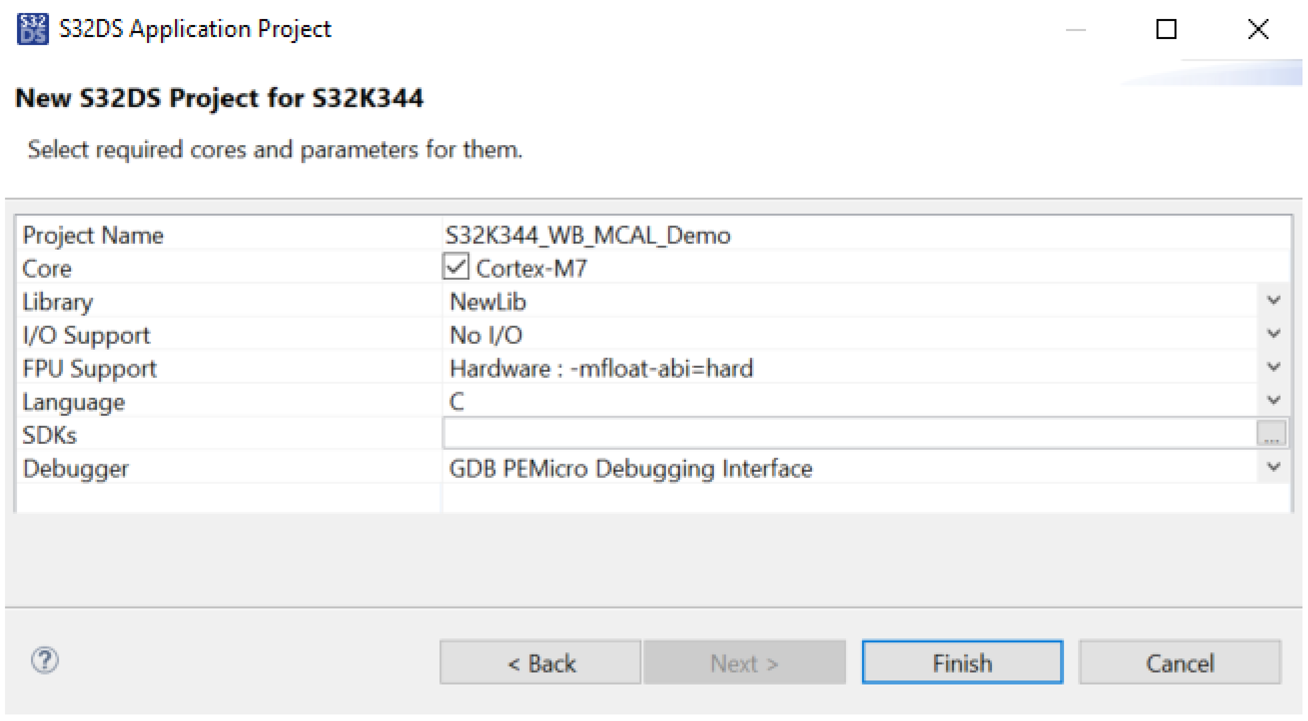
Core Kind	Name	Toolchain
M7	Cortex-M7	NXP GCC 9.2 for Arm 32-bit Bare-Metal

Description: GNU 9.2 Toolchain is selected

3. Select library and set FPU and so on.

   If you want to print something in console, you can select `Debugger Consoler` at `I/O Support` item, Otherwise select `No I/O`. For "FPU Support" item, suggest you select: `Hardware: -mfloat-abi=hard`.

   SDKs item keep None, because we will not use any LLD but MCAL. Debugger item depends on your device.

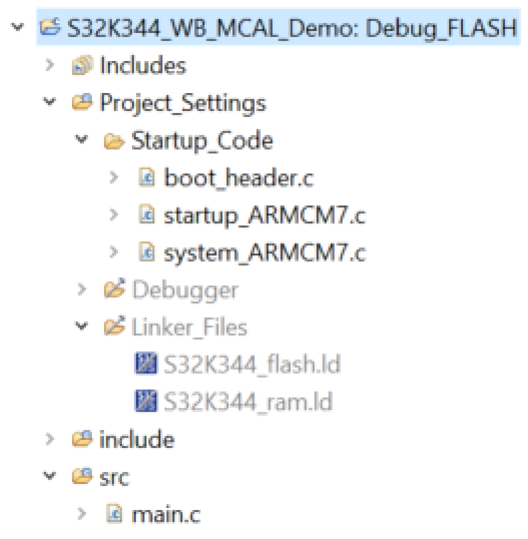

S32DS Application Project

New S32DS Project for S32K344

Select required cores and parameters for them.

Project Name	S32K344_WB_MCAL_Demo
Core	☑ Cortex-M7
Library	NewLib
I/O Support	No I/O
FPU Support	Hardware : -mfloat-abi=hard
Language	C
SDKs	
Debugger	GDB PEMicro Debugging Interface

4. Click `Finish` to create project. In default, it will help create the startup code, vector table file, link file, some header files and the main.c file and so on.


```
S32K344_WB_MCAL_Demo: Debug_FLASH
  Includes
  Project_Settings
    Startup_Code
      boot_header.c
      startup_ARMCM7.c
      system_ARMCM7.c
    Debugger
    Linker_Files
      S32K344_flash.ld
      S32K344_ram.ld
  include
  src
    main.c
```

2. Remove Files

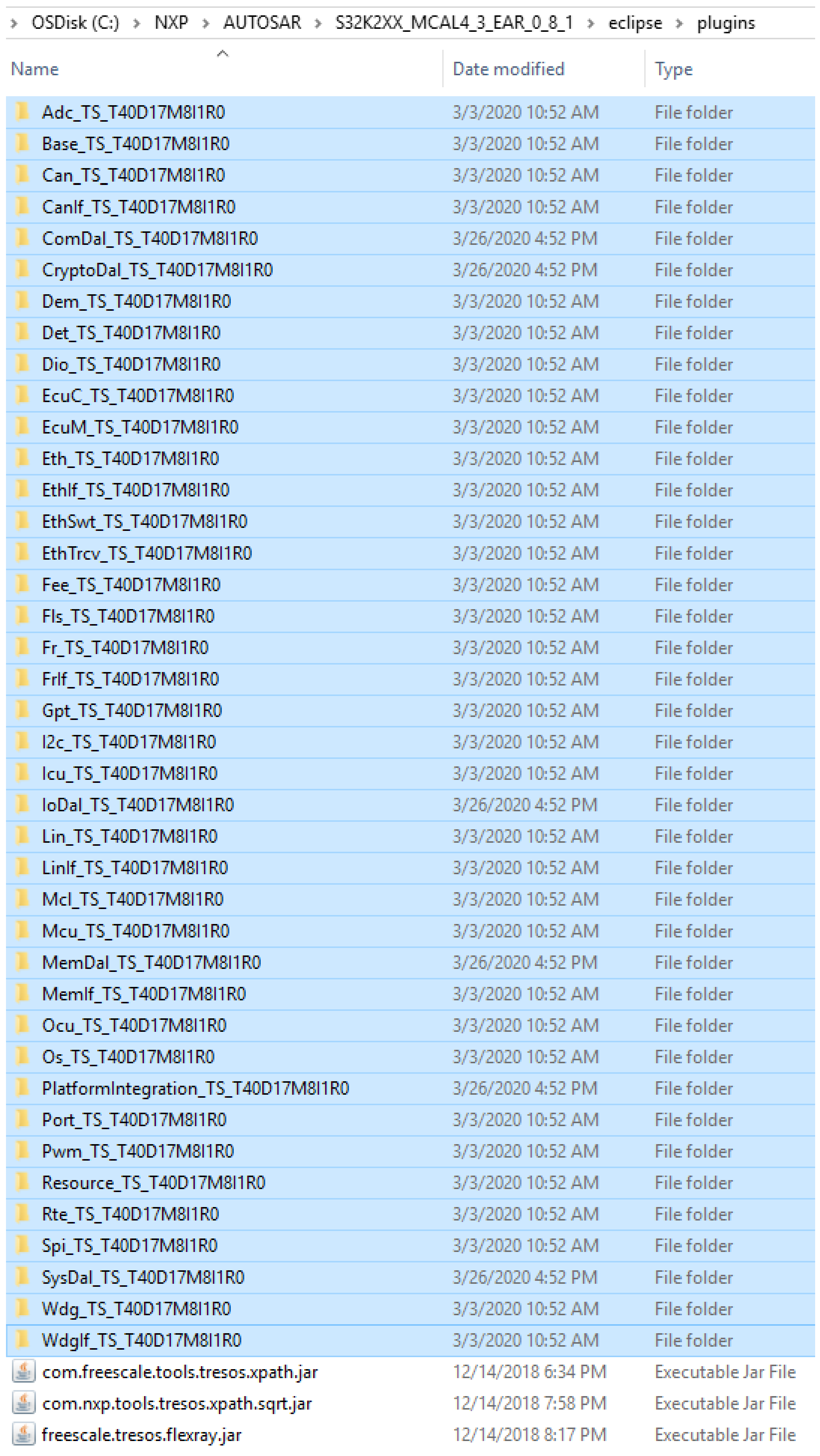
1. Delete unnecessary files. Here we need delete the `"include"` folder which created by default. These header files in "include" folder are for LLD code using and MCAL has its own header files(even through most files are the same, but still few files are different) which are located in `"MCAL\Base_Ts_T4xxx\header"`.
2. Delete the `"Startup_Code"` folder and `"Linker_Files"` folder which located in "Project_Settings". MCAL will use its own link file and startup file, which are located in the following path: `"MCAL\Platform_TS_T4xxx\build_files\gcc"` and `"MCAL\Platform_TS_T4xxx\src\m7"`.

> In fact, we can also use files created by S32DS, but some content may need to be modified, so in order to reduce our workload, we use MCAL files, which can save a lot of time.

3. Add MCAL

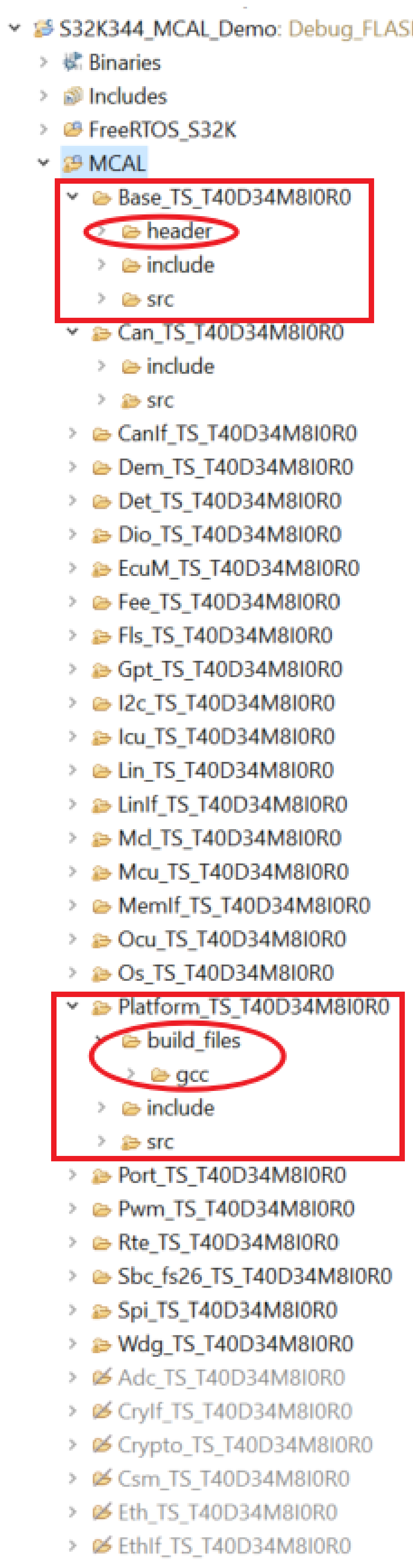
1. Copy the `"include"` and `"src"` folder of the modules which you used. These folders located in the plugin folder. The default install path is: `"C:\NXP\AUTOSAR\"`. If you installed multiple product versions, there will be many different product version folders, go into the right folder and find `"eclipse\plugins"`. You also can copy all modules' folder except the *.jar file. Please refer the following picture, just copy the selected folder. We put these folder into one folder named `"MCAL"` and located in the project root path.

> If you only copy the modules which you used, you can ignore step 3.3. Attention, their may have some problem if you are fresher on MCAL, the problem is how to know which module you used. for the these module which related with hardware, its easy to identify. For example, the "Adc" module or "Spi" module and so on. If we want to use it, we need configure it. but the problem is that some modules are the common module, like the "Base" and "Rte" module, even we didn't configure it, but some other module dependent on them. so we also need copy these common modules.



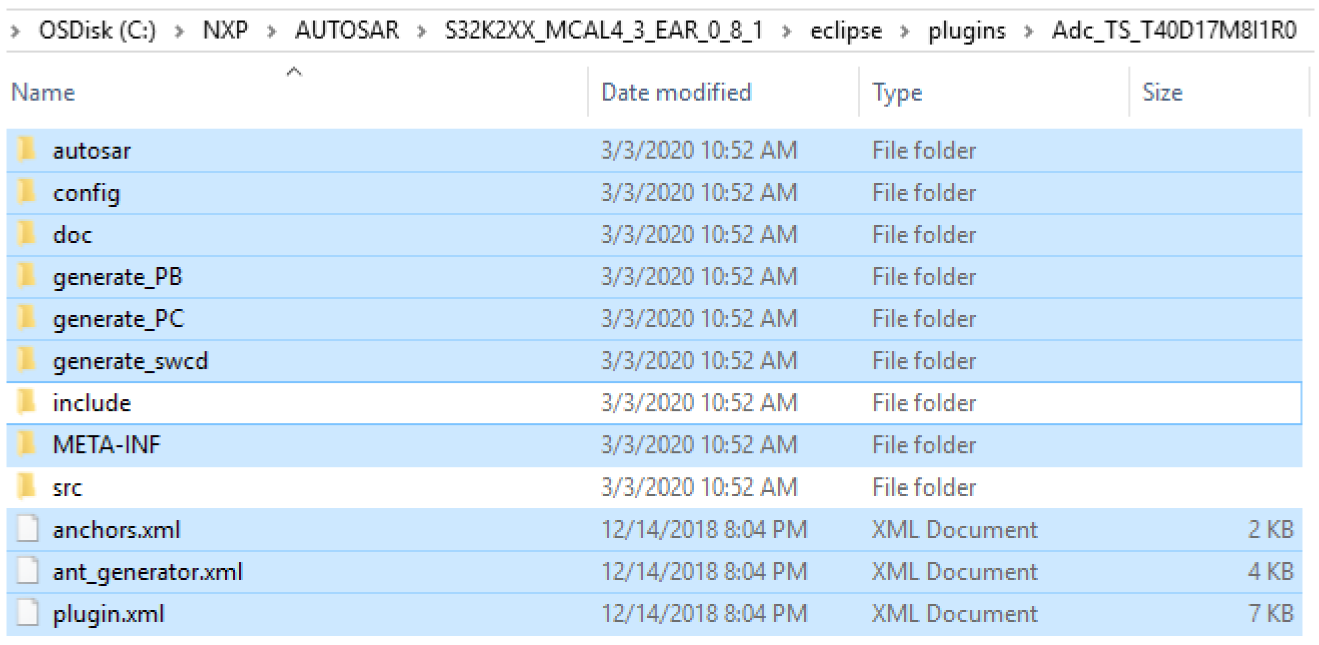
> OSDisk (C:) > NXP > AUTOSAR > S32K2XX_MCAL4_3_EAR_0_8_1 > eclipse > plugins

Name	Date modified	Type
Adc_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
Base_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
Can_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
CanIf_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
ComDal_TS_T40D17M8I1R0	3/26/2020 4:52 PM	File folder
CryptoDal_TS_T40D17M8I1R0	3/26/2020 4:52 PM	File folder
Dem_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
Det_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
Dio_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
EcuC_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
EcuM_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
Eth_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
EthIf_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
EthSwt_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
EthTrcv_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
Fee_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
Fls_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
Fr_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
FrIf_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
Gpt_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
I2c_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
Icu_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
IoDal_TS_T40D17M8I1R0	3/26/2020 4:52 PM	File folder
Lin_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
LinIf_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
Mcl_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
Mcu_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
MemDal_TS_T40D17M8I1R0	3/26/2020 4:52 PM	File folder
MemIf_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
Ocu_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
Os_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
PlatformIntegration_TS_T40D17M8I1R0	3/26/2020 4:52 PM	File folder
Port_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
Pwm_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
Resource_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
Rte_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
Spi_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
SysDal_TS_T40D17M8I1R0	3/26/2020 4:52 PM	File folder
Wdg_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
WdgIf_TS_T40D17M8I1R0	3/3/2020 10:52 AM	File folder
com.freescale.tools.tresos.xpath.jar	12/14/2018 6:34 PM	Executable Jar File
com.nxp.tools.tresos.xpath.sqrt.jar	12/14/2018 7:58 PM	Executable Jar File
freescale.tresos.flexray.jar	12/14/2018 8:17 PM	Executable Jar File



```
S32K344_MCAL_Demo: Debug_FLASH
  Binaries
  Includes
  FreeRTOS_S32K
  MCAL
    Base_TS_T40D34M8I0R0
      header
      include
      src
    Can_TS_T40D34M8I0R0
      include
      src
    CanIf_TS_T40D34M8I0R0
    Dem_TS_T40D34M8I0R0
    Det_TS_T40D34M8I0R0
    Dio_TS_T40D34M8I0R0
    EcuM_TS_T40D34M8I0R0
    Fee_TS_T40D34M8I0R0
    Fls_TS_T40D34M8I0R0
    Gpt_TS_T40D34M8I0R0
    I2c_TS_T40D34M8I0R0
    Icu_TS_T40D34M8I0R0
    Lin_TS_T40D34M8I0R0
    LinIf_TS_T40D34M8I0R0
    Mcl_TS_T40D34M8I0R0
    Mcu_TS_T40D34M8I0R0
    MemIf_TS_T40D34M8I0R0
    Ocu_TS_T40D34M8I0R0
    Os_TS_T40D34M8I0R0
    Platform_TS_T40D34M8I0R0
      build_files
        gcc
      include
      src
    Port_TS_T40D34M8I0R0
    Pwm_TS_T40D34M8I0R0
    Rte_TS_T40D34M8I0R0
    Sbc_fs26_TS_T40D34M8I0R0
    Spi_TS_T40D34M8I0R0
    Wdg_TS_T40D34M8I0R0
    Adc_TS_T40D34M8I0R0
    CryIf_TS_T40D34M8I0R0
    Crypto_TS_T40D34M8I0R0
    Csm_TS_T40D34M8I0R0
    Eth_TS_T40D34M8I0R0
    EthIf_TS_T40D34M8I0R0
```

2. Attention again, except **Base** and **Platform** module, all other modules only copy `"include"` and `"src"` folders. You need delete all unnecessary files/folders under each module's folder. Only keep `"include"` and `"src"` folders. The following picture take "Adc" module as an example, delete the selected files/folders, the other modules are same. For `"Base"` and `"Platform"` modules, we can see from above picture, we need copy `"header"` and `"build_files"` folders besides `"include"` and `"src"`.

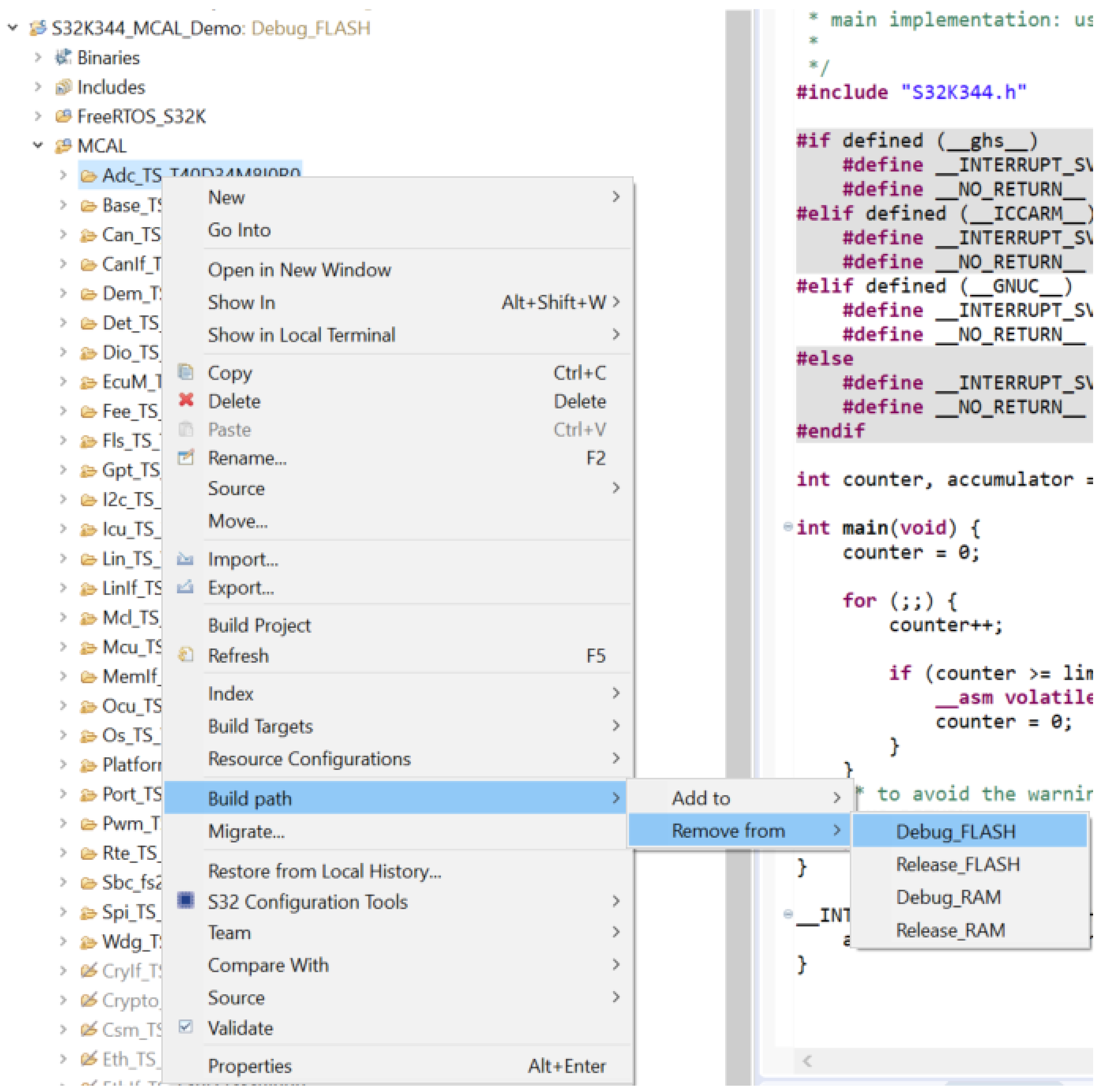


OSDisk (C:) > NXP > AUTOSAR > S32K2XX_MCAL4_3_EAR_0_8_1 > eclipse > plugins > Adc_TS_T40D17M8I1R0

Name	Date modified	Type	Size
autosar	3/3/2020 10:52 AM	File folder	
config	3/3/2020 10:52 AM	File folder	
doc	3/3/2020 10:52 AM	File folder	
generate_PB	3/3/2020 10:52 AM	File folder	
generate_PC	3/3/2020 10:52 AM	File folder	
generate_swcd	3/3/2020 10:52 AM	File folder	
include	3/3/2020 10:52 AM	File folder	
META-INF	3/3/2020 10:52 AM	File folder	
src	3/3/2020 10:52 AM	File folder	
anchors.xml	12/14/2018 8:04 PM	XML Document	2 KB
ant_generator.xml	12/14/2018 8:04 PM	XML Document	4 KB
plugin.xml	12/14/2018 8:04 PM	XML Document	7 KB

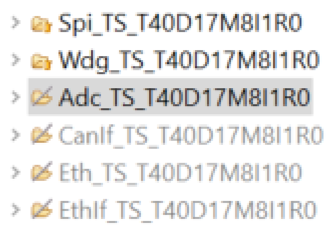
3. Before you do the following process, I strongly suggest you "Refresh" the project folder tree first(You can press "F5"). In the previous step 3.1, If you only copied the module you used, you can ignore this step.

Disable the modules which you did not used, otherwise it will cause many compiling error. Select the module folder in the project tree and right click. Then `Build path` -> `Remove from` -> `Debug_FLASH`.

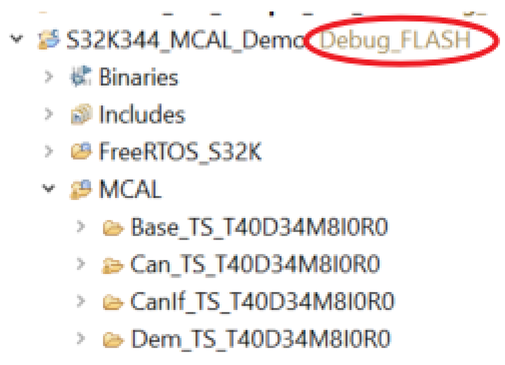


```
 * main implementation: us
 *
 */
#include "S32K344.h"

#if defined (__ghs__)
    #define __INTERRUPT_SV
    #define __NO_RETURN__
#elif defined (__ICCARM__)
    #define __INTERRUPT_SV
    #define __NO_RETURN__
#elif defined (__GNUC__)
    #define __INTERRUPT_SV
    #define __NO_RETURN__
#else
    #define __INTERRUPT_SV
    #define __NO_RETURN__
#endif

int counter, accumulator =

int main(void) {
    counter = 0;

    for (;;) {
        counter++;

        if (counter >= lim
            __asm volatile
            counter = 0;
        }
    }
    * to avoid the warnir
}

__INT
    a
}
```

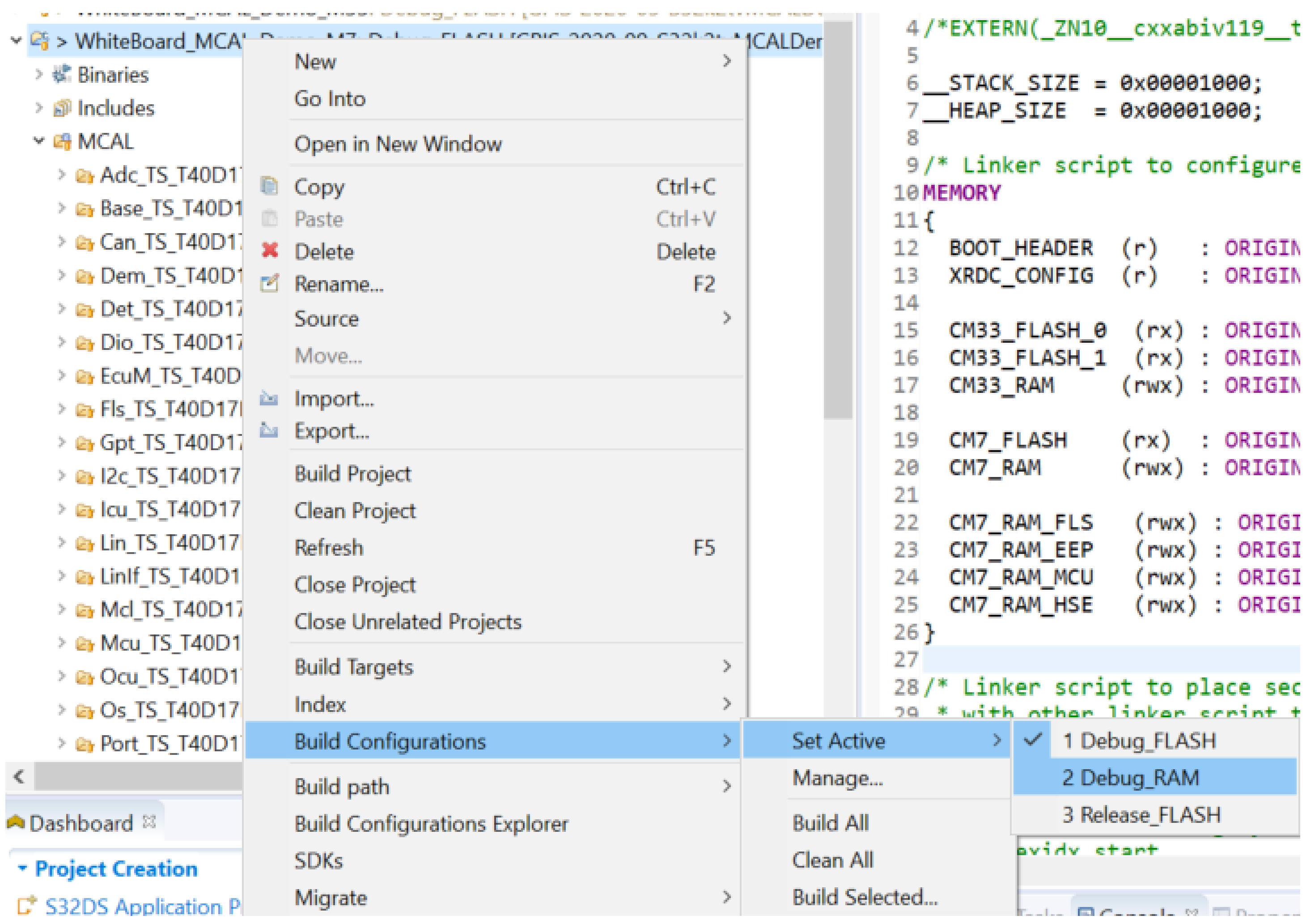
Then it will move to the tail and color will change to gray. You can select multiple folders once time. If you want to re-active the module, the method is same, under `Build path`, select `Add to` -> `Debug_FLASH`.



> Attention: There have three options, Debug_FLASH, Release_FLASH, Debug_RAM. Commonly, the project's default state is Debug_FLASH, you can check the state from the project tree. Please refer the following picture.



If you want to change to release, right click on project name, select `Build Configurations` -> `Set Active` -> `Release FLASH`. If you change this, the previous process (Add/Remove module) need synchronize.

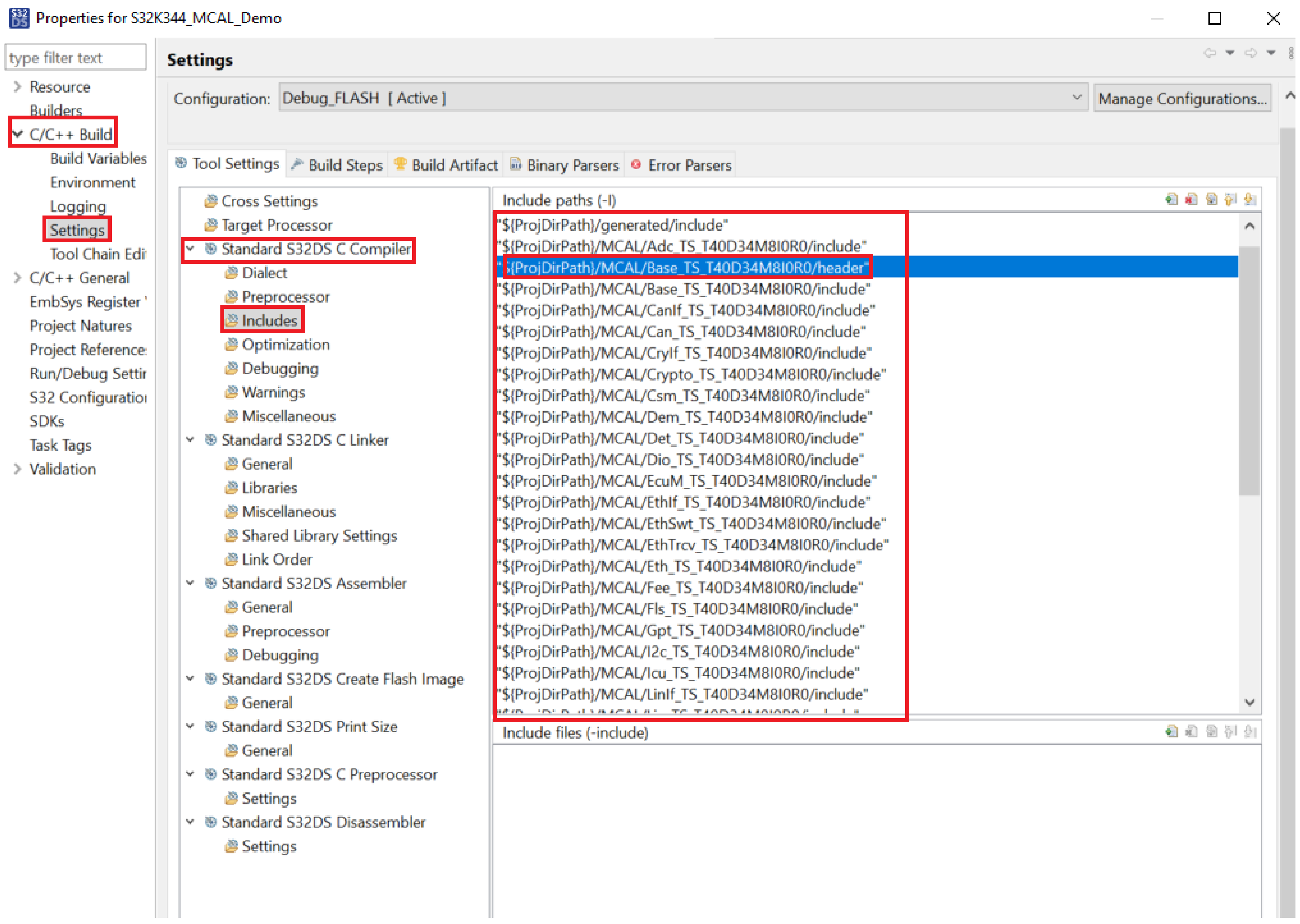


```
/*EXTERN(_ZN10__cxxabiv119__t

__STACK_SIZE = 0x00001000;
__HEAP_SIZE  = 0x00001000;

/* Linker script to configure
MEMORY
{
  BOOT_HEADER  (r)    : ORIGIN
  XRDC_CONFIG  (r)    : ORIGIN

  CM33_FLASH_0  (rx)  : ORIGIN
  CM33_FLASH_1  (rx)  : ORIGIN
  CM33_RAM     (rwx)  : ORIGIN

  CM7_FLASH    (rx)   : ORIGIN
  CM7_RAM      (rwx)  : ORIGIN

  CM7_RAM_FLS   (rwx) : ORIGI
  CM7_RAM_EEP   (rwx) : ORIGI
  CM7_RAM_MCU   (rwx) : ORIGI
  CM7_RAM_HSE   (rwx) : ORIGI
}

/* Linker script to place sec
```

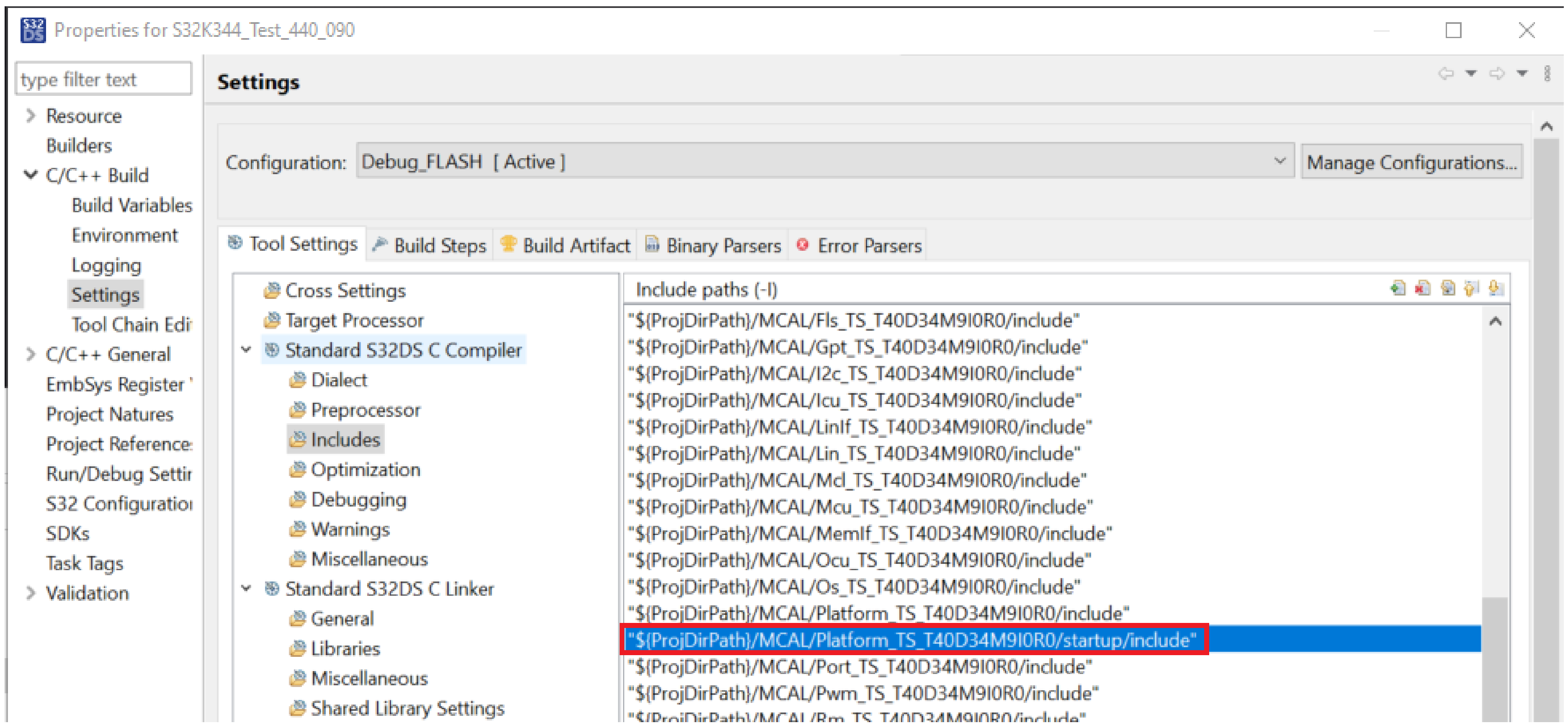
4. Property Setting

1. Now, we need include all modules' header file path in project's property. Right click on project name, click Properties. Then `C/C++ Build` -> `Standard S32DS C Compiler` -> `Includes`.

> - Pay attention to **Base** module, it need includes two paths. one is "include", another is "header".
> - And the **Platform** module also a specially module, you need set the startup header file path.
> - Don't forgot to add the MCAL generated codes' header file path.



```
"${ProjDirPath}/generated/include"
"${ProjDirPath}/MCAL/Adc_TS_T40D34M80I0R0/include"
"${ProjDirPath}/MCAL/Base_TS_T40D34M80I0R0/header"
"${ProjDirPath}/MCAL/Base_TS_T40D34M80I0R0/include"
"${ProjDirPath}/MCAL/CanIf_TS_T40D34M80I0R0/include"
"${ProjDirPath}/MCAL/Can_TS_T40D34M80I0R0/include"
"${ProjDirPath}/MCAL/CryIf_TS_T40D34M80I0R0/include"
"${ProjDirPath}/MCAL/Crypto_TS_T40D34M80I0R0/include"
"${ProjDirPath}/MCAL/Csm_TS_T40D34M80I0R0/include"
"${ProjDirPath}/MCAL/Dem_TS_T40D34M80I0R0/include"
"${ProjDirPath}/MCAL/Det_TS_T40D34M80I0R0/include"
"${ProjDirPath}/MCAL/Dio_TS_T40D34M80I0R0/include"
"${ProjDirPath}/MCAL/EcuM_TS_T40D34M80I0R0/include"
"${ProjDirPath}/MCAL/EthIf_TS_T40D34M80I0R0/include"
"${ProjDirPath}/MCAL/EthSwt_TS_T40D34M80I0R0/include"
"${ProjDirPath}/MCAL/EthTrcv_TS_T40D34M80I0R0/include"
"${ProjDirPath}/MCAL/Eth_TS_T40D34M80I0R0/include"
"${ProjDirPath}/MCAL/Fee_TS_T40D34M80I0R0/include"
"${ProjDirPath}/MCAL/Fls_TS_T40D34M80I0R0/include"
"${ProjDirPath}/MCAL/Gpt_TS_T40D34M80I0R0/include"
"${ProjDirPath}/MCAL/I2c_TS_T40D34M80I0R0/include"
"${ProjDirPath}/MCAL/Icu_TS_T40D34M80I0R0/include"
"${ProjDirPath}/MCAL/LinIf_TS_T40D34M80I0R0/include"
```


Properties for S32K344_Test_440_090

type filter text

- Resource
- Builders
- C/C++ Build
  - Build Variables
  - Environment
  - Logging
  - Settings
  - Tool Chain Edi
- C/C++ General
- EmbSys Register
- Project Natures
- Project Reference
- Run/Debug Settir
- S32 Configuratio
- SDKs
- Task Tags
- Validation

Settings

Configuration: Debug_FLASH [Active] — Manage Configurations...

Tool Settings | Build Steps | Build Artifact | Binary Parsers | Error Parsers

- Cross Settings
- Target Processor
- Standard S32DS C Compiler
  - Dialect
  - Preprocessor
  - Includes
  - Optimization
  - Debugging
  - Warnings
  - Miscellaneous
- Standard S32DS C Linker
  - General
  - Libraries
  - Miscellaneous
  - Shared Library Settings

Include paths (-I)

```
"${ProjDirPath}/MCAL/Fls_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Gpt_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/I2c_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Icu_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/LinIf_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Lin_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Mcl_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Mcu_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/MemIf_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Ocu_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Os_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Platform_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Platform_TS_T40D34M9I0R0/startup/include"
"${ProjDirPath}/MCAL/Port_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Pwm_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Rm_TS_T40D34M9I0R0/include"
```

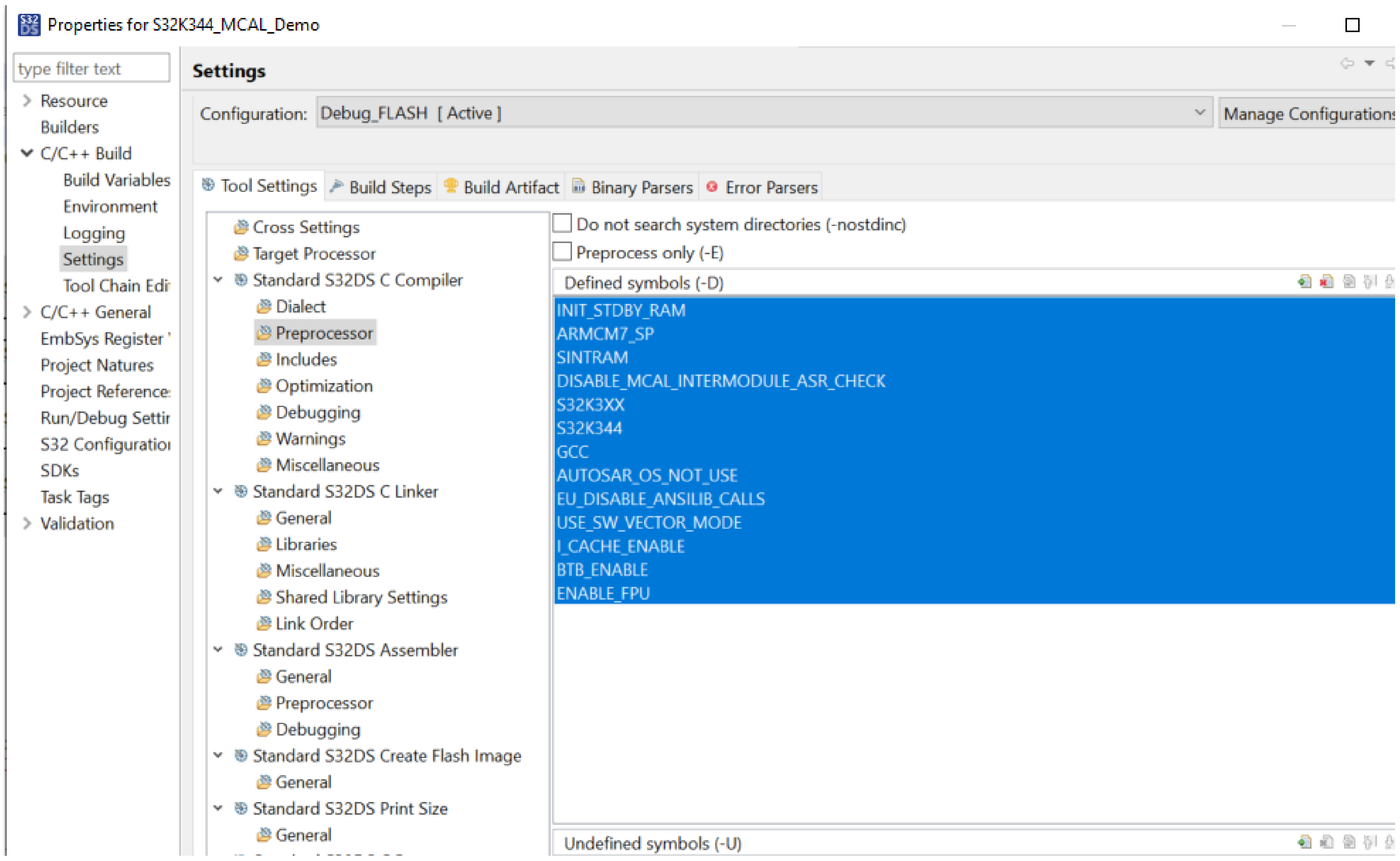
You'd better use the relative path here.

You can copy from the following list (Note that different versions of the software may be different):

```
"${ProjDirPath}/include"
"${ProjDirPath}/generate/include"
"${ProjDirPath}/MCAL/Adc_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Base_TS_T40D34M9I0R0/header"
"${ProjDirPath}/MCAL/Base_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/CanIf_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Can_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Crc_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/CryIf_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Crypto_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Csm_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Dem_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Det_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Dio_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/EcuM_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/EthIf_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/EthSwt_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/EthTrcv_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Eth_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Fee_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Fls_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Gpt_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/I2c_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Icu_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/LinIf_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Lin_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Mcl_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Mcu_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/MemIf_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Ocu_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Os_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Platform_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Platform_TS_T40D34M9I0R0/startup/include"
"${ProjDirPath}/MCAL/Port_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Pwm_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Rm_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Rte_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Sai_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Sent_TS_T40D34M9I0R0/include"
```

```
"${ProjDirPath}/MCAL/Spi_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Uart_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/WdgIf_TS_T40D34M9I0R0/include"
"${ProjDirPath}/MCAL/Wdg_TS_T40D34M9I0R0/include"
```

2. Define the necessary MACRO for project. Also in the property window. If your debugger is PE/Multilink and you didn't configure MPU, suggest you don't define: "I_CACHE_ENABLE" and "D_CACHE_ENABLE". It will affect you debug. If you plan to use FPU, you need define: "ENABLE_FPU" in here. That means the FPU is active by two conditions, the one is the item when you create the project, you need enable FPU; another condition is this macro definition.

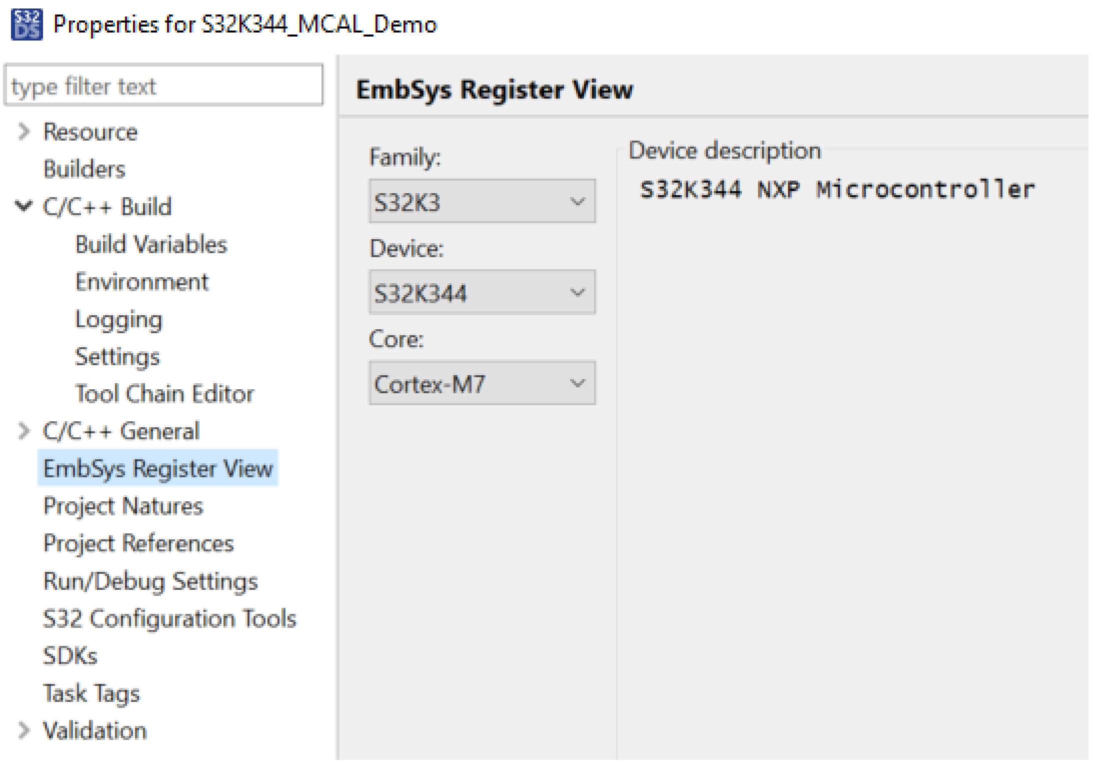


```
INIT_STDBY_RAM
ARMCM7_SP
SINTRAM
DISABLE_MCAL_INTERMODULE_ASR_CHECK
S32K3XX
S32K344
GCC
AUTOSAR_OS_NOT_USE
EU_DISABLE_ANSILIB_CALLS
USE_SW_VECTOR_MODE
I_CACHE_ENABLE
BTB_ENABLE
ENABLE_FPU
```

You can copy from the following list:

```
INIT_STDBY_RAM
ARMCM7_SP
SINTRAM
DISABLE_MCAL_INTERMODULE_ASR_CHECK
S32K3XX
S32K344
GCC
AUTOSAR_OS_NOT_USE
EU_DISABLE_ANSILIB_CALLS
USE_SW_VECTOR_MODE
I_CACHE_ENABLE
BTB_ENABLE
ENABLE_FPU
```

3. Check/Set device type for debug in property window.


Properties for S32K344_MCAL_Demo

EmbSys Register View

Family: S32K3

Device: S32K344

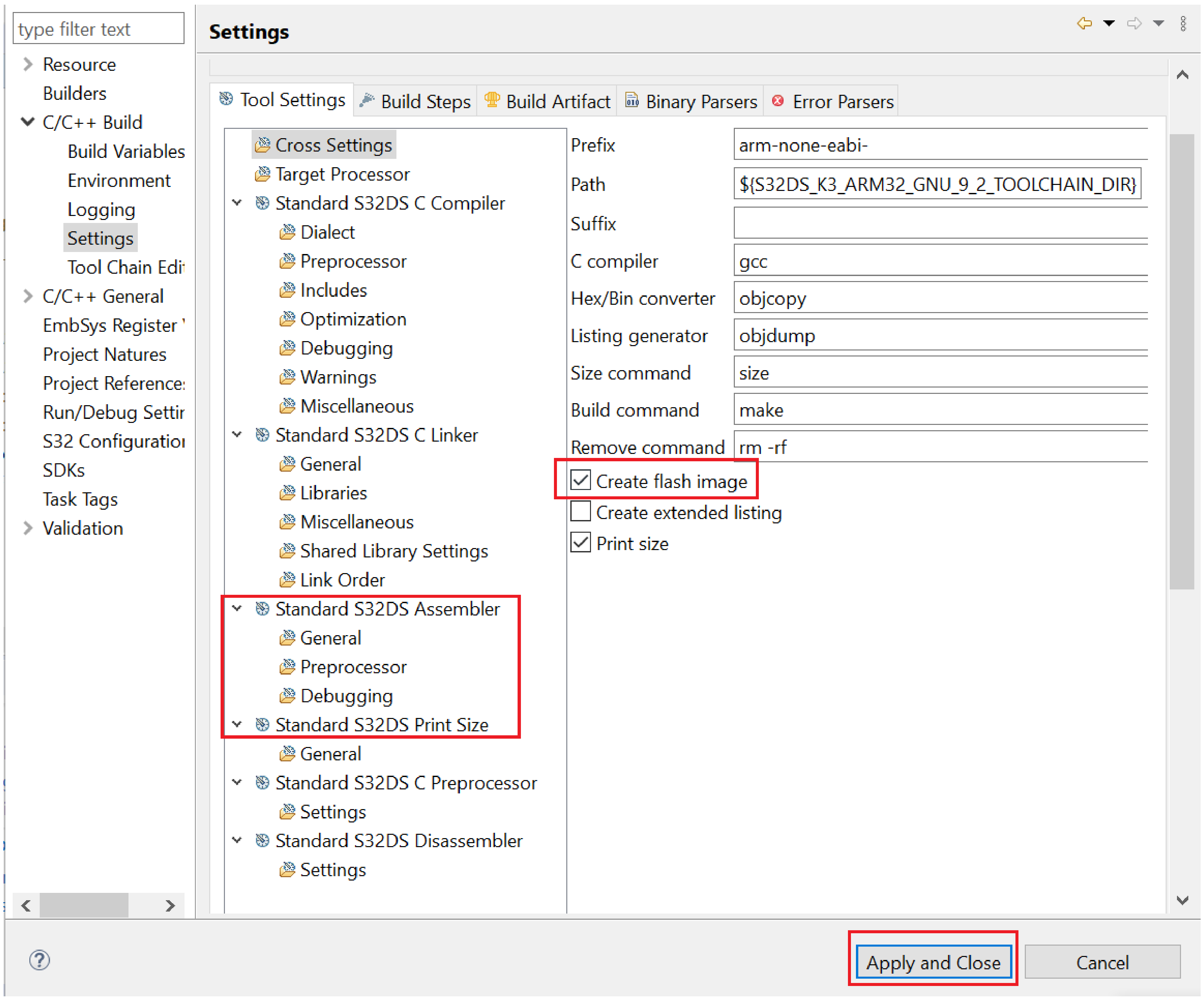
Core: Cortex-M7

Device description: S32K344 NXP Microcontroller

4. Set compiler parameters in property window.

   About the compiler parameters, please refer the "xxx_ReleaseNotes.pdf". Our document and demo project also take a reference from that file. Maybe there have some special requirements in your project, in such case, you need very clear the parameters what does it mean, this out of the scope of this document. But I think most settings apply to your project. For example, the startup file setting, the link file setting and so on.

   If you want to generate some other formats output file, like *.mot, *.s19, *.bin and so on, you can check the item: "Create flash image", please refer the following picture. After you check this item, you need click `Apply and Close` button, and re-open the property window, then you can find that the left side menu will appear a new feature, please see the following second picture.


Settings

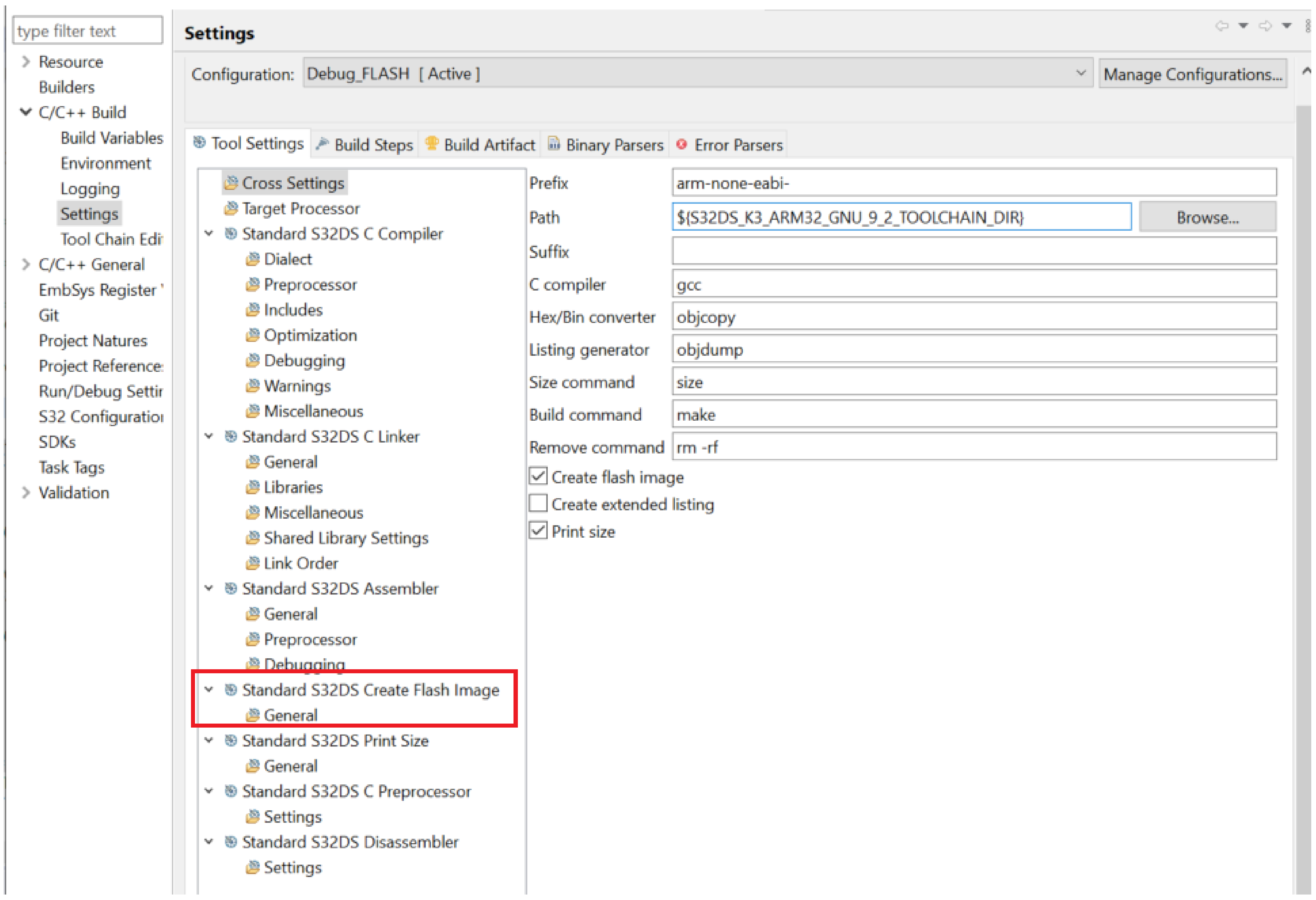
Tool Settings | Build Steps | Build Artifact | Binary Parsers | Error Parsers

Prefix	arm-none-eabi-
Path	${S32DS_K3_ARM32_GNU_9_2_TOOLCHAIN_DIR}
Suffix	
C compiler	gcc
Hex/Bin converter	objcopy
Listing generator	objdump
Size command	size
Build command	make
Remove command	rm -rf

☑ Create flash image

☐ Create extended listing

☑ Print size


Settings

Configuration: Debug_FLASH [Active]

Manage Configurations...

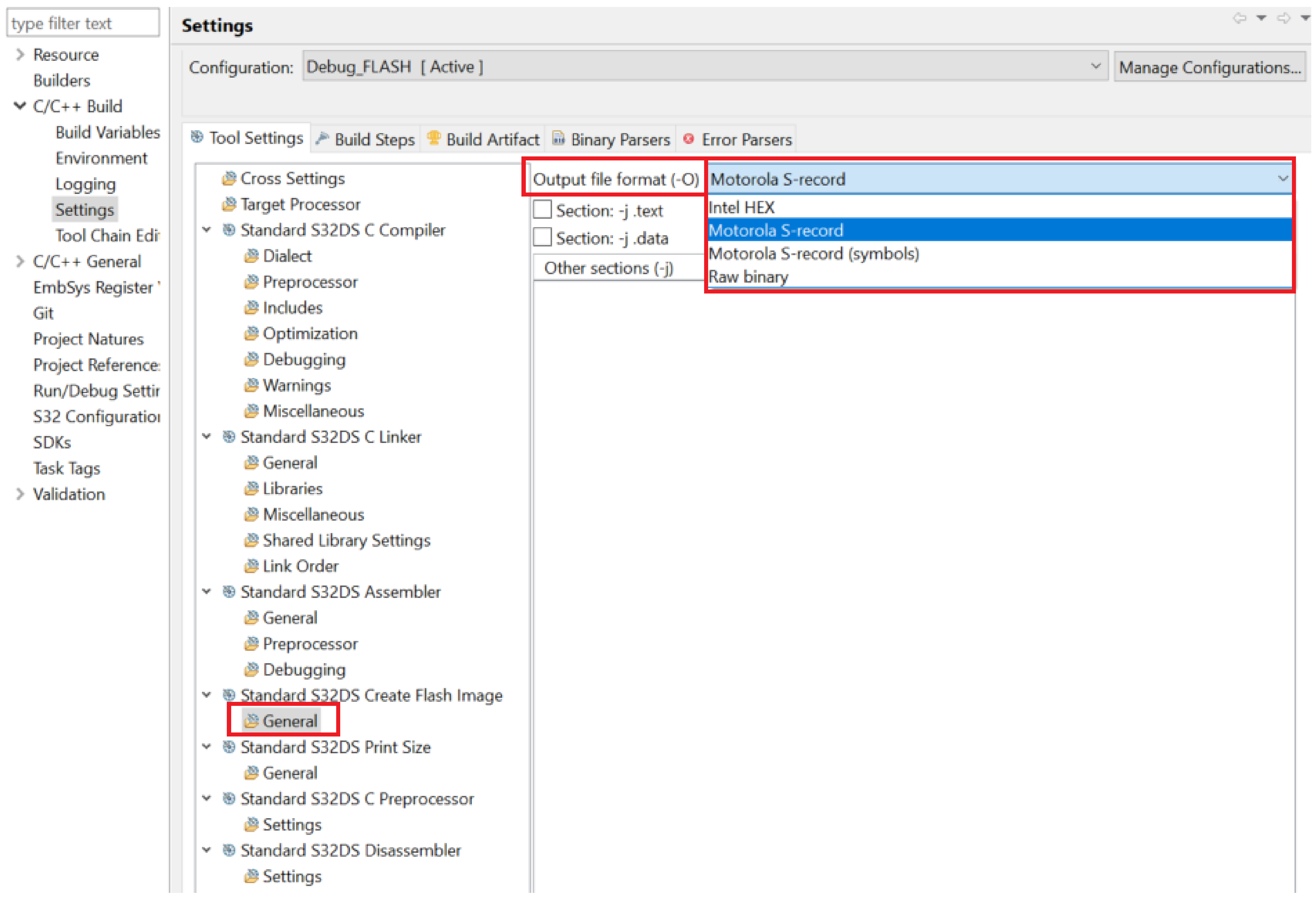
Tool Settings | Build Steps | Build Artifact | Binary Parsers | Error Parsers

- Cross Settings
- Target Processor
- Standard S32DS C Compiler
  - Dialect
  - Preprocessor
  - Includes
  - Optimization
  - Debugging
  - Warnings
  - Miscellaneous
- Standard S32DS C Linker
  - General
  - Libraries
  - Miscellaneous
  - Shared Library Settings
  - Link Order
- Standard S32DS Assembler
  - General
  - Preprocessor
  - Debugging
- Standard S32DS Create Flash Image
  - General
- Standard S32DS Print Size
  - General
- Standard S32DS C Preprocessor
  - Settings
- Standard S32DS Disassembler
  - Settings

Prefix	arm-none-eabi-
Path	${S32DS_K3_ARM32_GNU_9_2_TOOLCHAIN_DIR}
Suffix	
C compiler	gcc
Hex/Bin converter	objcopy
Listing generator	objdump
Size command	size
Build command	make
Remove command	rm -rf

[✓] Create flash image
[ ] Create extended listing
[✓] Print size

Then you can select the output file format in the new feature window. Please refer the following picture.


Settings

Configuration: Debug_FLASH [Active]

Manage Configurations...

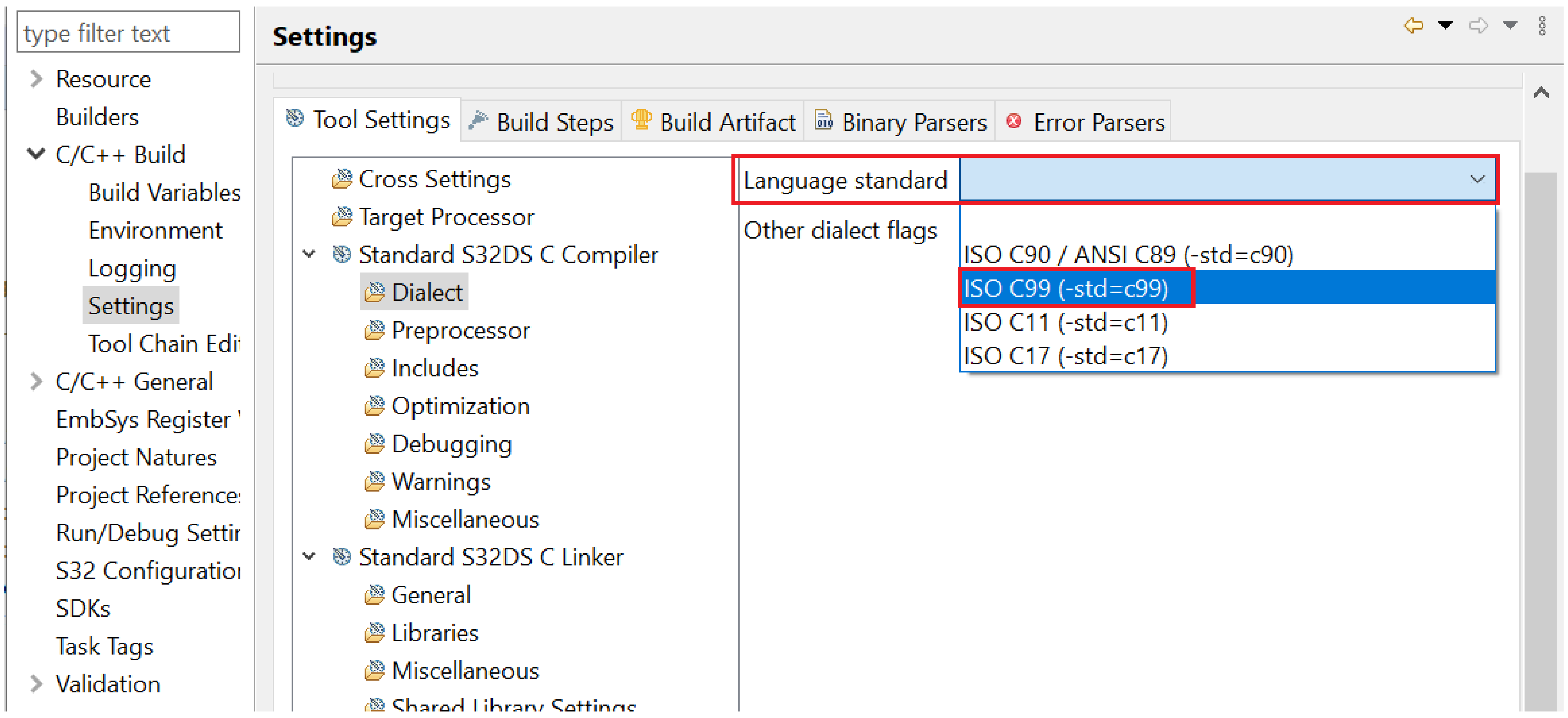
Tool Settings | Build Steps | Build Artifact | Binary Parsers | Error Parsers

- Standard S32DS Create Flash Image
  - General

Output file format (-O): Motorola S-record
- Intel HEX
- Motorola S-record
- Motorola S-record (symbols)
- Raw binary

[ ] Section: -j .text
[ ] Section: -j .data
Other sections (-j)

Set the language standard to: ISO C99



Language standard

Other dialect flags

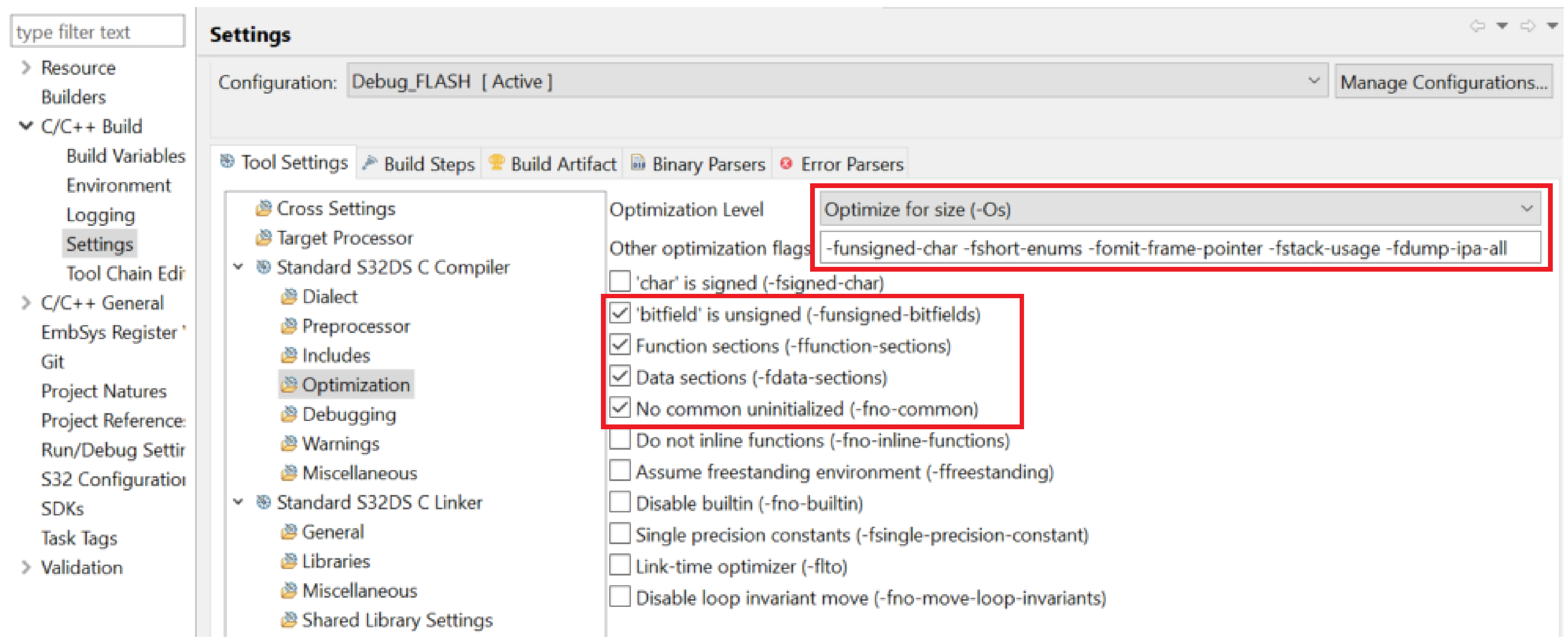
ISO C90 / ANSI C89 (-std=c90)

ISO C99 (-std=c99)

ISO C11 (-std=c11)

ISO C17 (-std=c17)

Set optimization level. Here we set to `-Os` . This level is refer from ReleaseNotes.pdf. This is a global setting, in the real project, you can set the level according to your project's requirement. S32DS also support set the level for each single file independently. And some other parameters need setting in this window. please refer the following picture.



Configuration: Debug_FLASH [Active]

Optimization Level: Optimize for size (-Os)

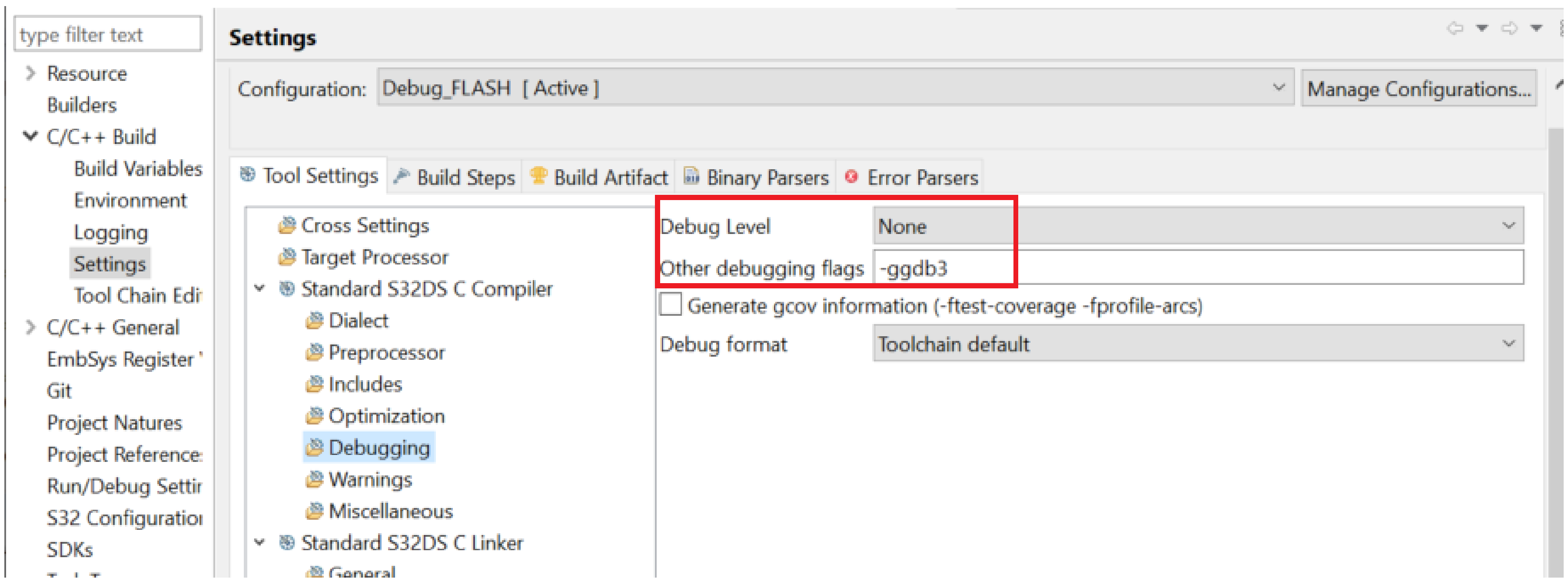
Other optimization flags: -funsigned-char -fshort-enums -fomit-frame-pointer -fstack-usage -fdump-ipa-all

- [ ] 'char' is signed (-fsigned-char)
- [x] 'bitfield' is unsigned (-funsigned-bitfields)
- [x] Function sections (-ffunction-sections)
- [x] Data sections (-fdata-sections)
- [x] No common uninitialized (-fno-common)
- [ ] Do not inline functions (-fno-inline-functions)
- [ ] Assume freestanding environment (-ffreestanding)
- [ ] Disable builtin (-fno-builtin)
- [ ] Single precision constants (-fsingle-precision-constant)
- [ ] Link-time optimizer (-flto)
- [ ] Disable loop invariant move (-fno-move-loop-invariants)

About the "Other optimization flags", you can copy from following list:

> -funsigned-char -fshort-enums -fomit-frame-pointer -fstack-usage -fdump-ipa-all

Set Debugging Level.



Configuration: Debug_FLASH [Active]

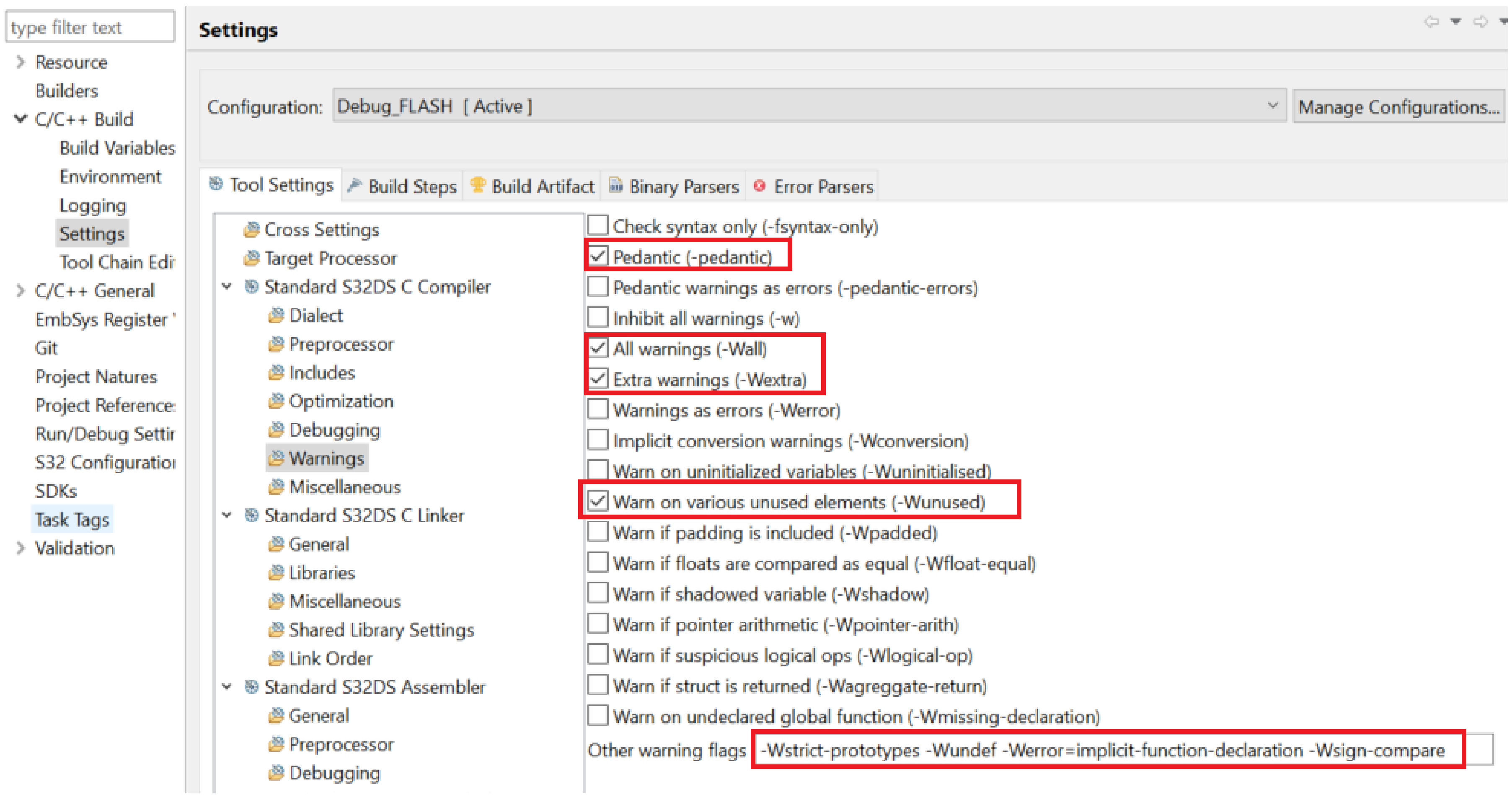
Debug Level: None

Other debugging flags: -ggdb3

- [ ] Generate gcov information (-ftest-coverage -fprofile-arcs)

Debug format: Toolchain default

Set Warnings parameters:


Settings

Configuration: Debug_FLASH [Active]

Tool Settings > Standard S32DS C Compiler > Warnings

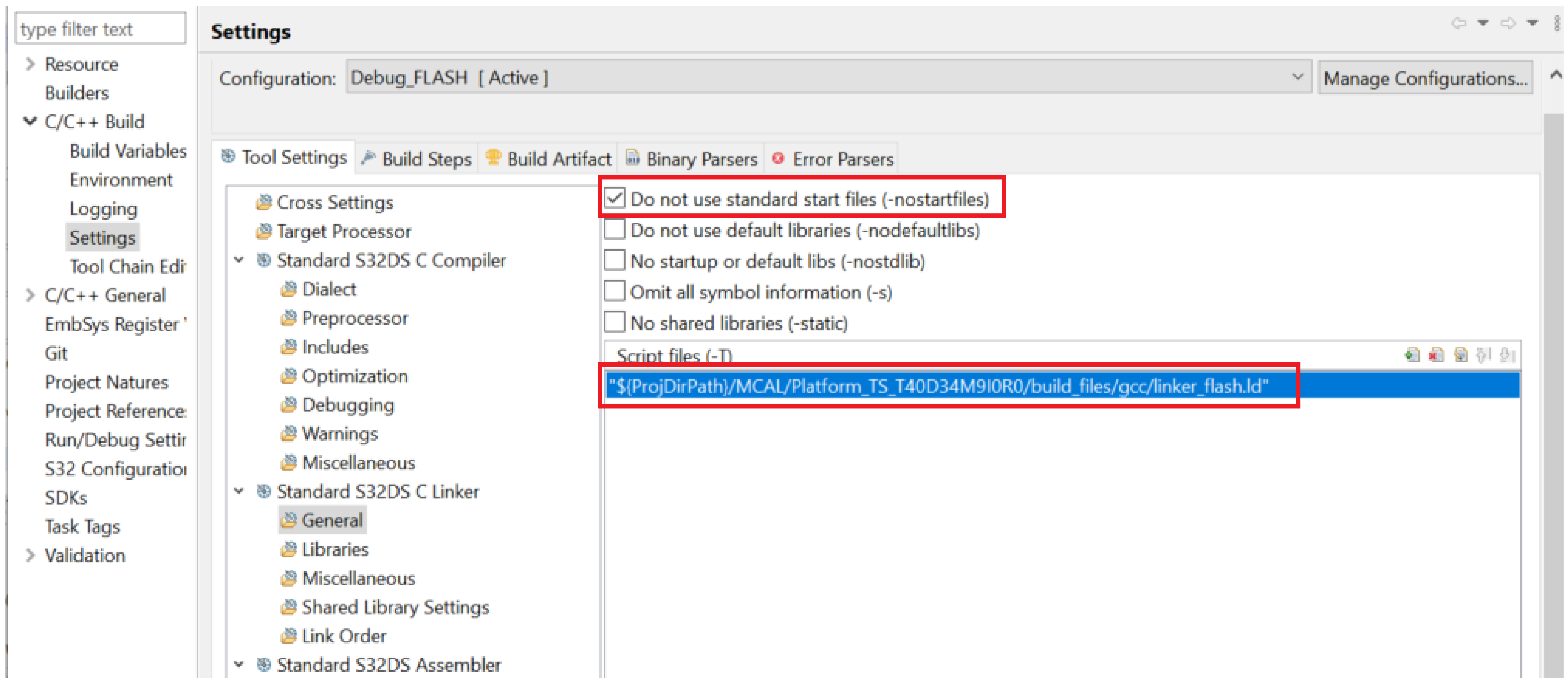
- [ ] Check syntax only (-fsyntax-only)
- [✓] Pedantic (-pedantic)
- [ ] Pedantic warnings as errors (-pedantic-errors)
- [ ] Inhibit all warnings (-w)
- [✓] All warnings (-Wall)
- [✓] Extra warnings (-Wextra)
- [ ] Warnings as errors (-Werror)
- [ ] Implicit conversion warnings (-Wconversion)
- [ ] Warn on uninitialized variables (-Wuninitialised)
- [✓] Warn on various unused elements (-Wunused)
- [ ] Warn if padding is included (-Wpadded)
- [ ] Warn if floats are compared as equal (-Wfloat-equal)
- [ ] Warn if shadowed variable (-Wshadow)
- [ ] Warn if pointer arithmetic (-Wpointer-arith)
- [ ] Warn if suspicious logical ops (-Wlogical-op)
- [ ] Warn if struct is returned (-Wagreggate-return)
- [ ] Warn on undeclared global function (-Wmissing-declaration)

Other warning flags: -Wstrict-prototypes -Wundef -Werror=implicit-function-declaration -Wsign-compare

About the "Other warning flags", you can copy from following list:

> -Wstrict-prototypes -Wundef -Werror=implicit-function-declaration -Wsign-compare

Set link file and remove the default start files.


Settings

Configuration: Debug_FLASH [Active]

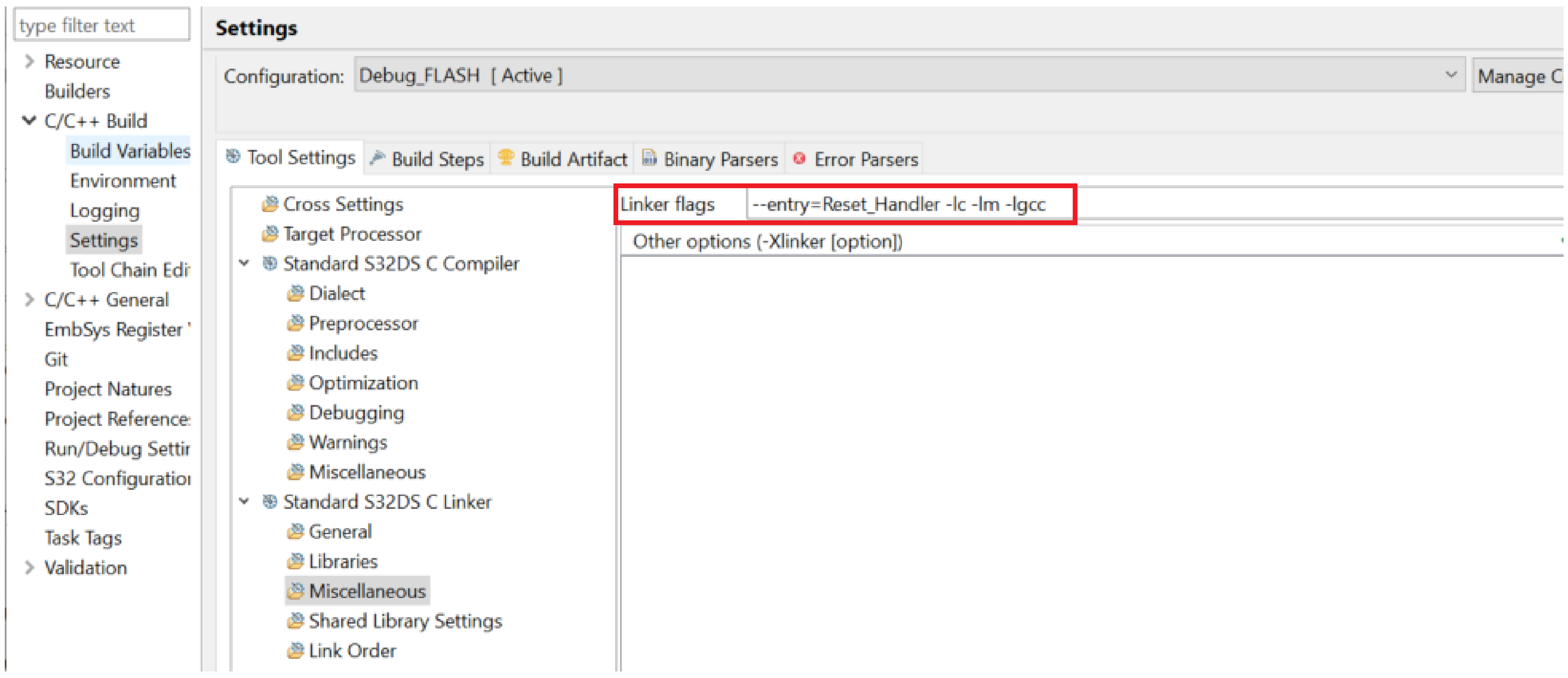
Tool Settings > Standard S32DS C Linker > General

- [✓] Do not use standard start files (-nostartfiles)
- [ ] Do not use default libraries (-nodefaultlibs)
- [ ] No startup or default libs (-nostdlib)
- [ ] Omit all symbol information (-s)
- [ ] No shared libraries (-static)

Script files (-T)

"${ProjDirPath}/MCAL/Platform_TS_T40D34M9I0R0/build_files/gcc/linker_flash.ld"

Set linker miscellaneous.


Settings

Configuration: Debug_FLASH [Active]

Tool Settings > Standard S32DS C Linker > Miscellaneous

Linker flags: --entry=Reset_Handler -lc -lm -lgcc

Other options (-Xlinker [option])

About the "Linker flags", you can copy from following list:

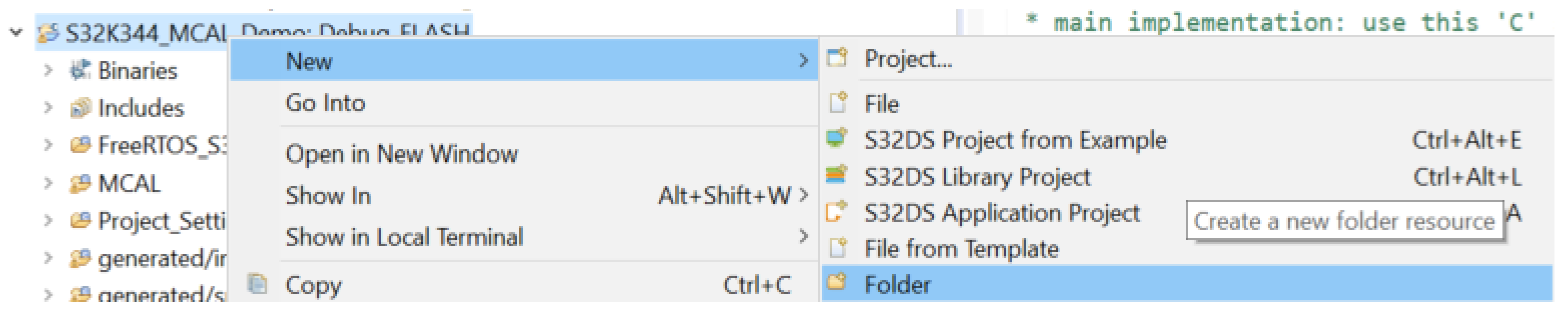
> --entry=Reset_Handler -lc -lm -lgcc

All these parameters which mentioned above, you can set it according to your requirement.

5. Link EB Tresos Generated Folder

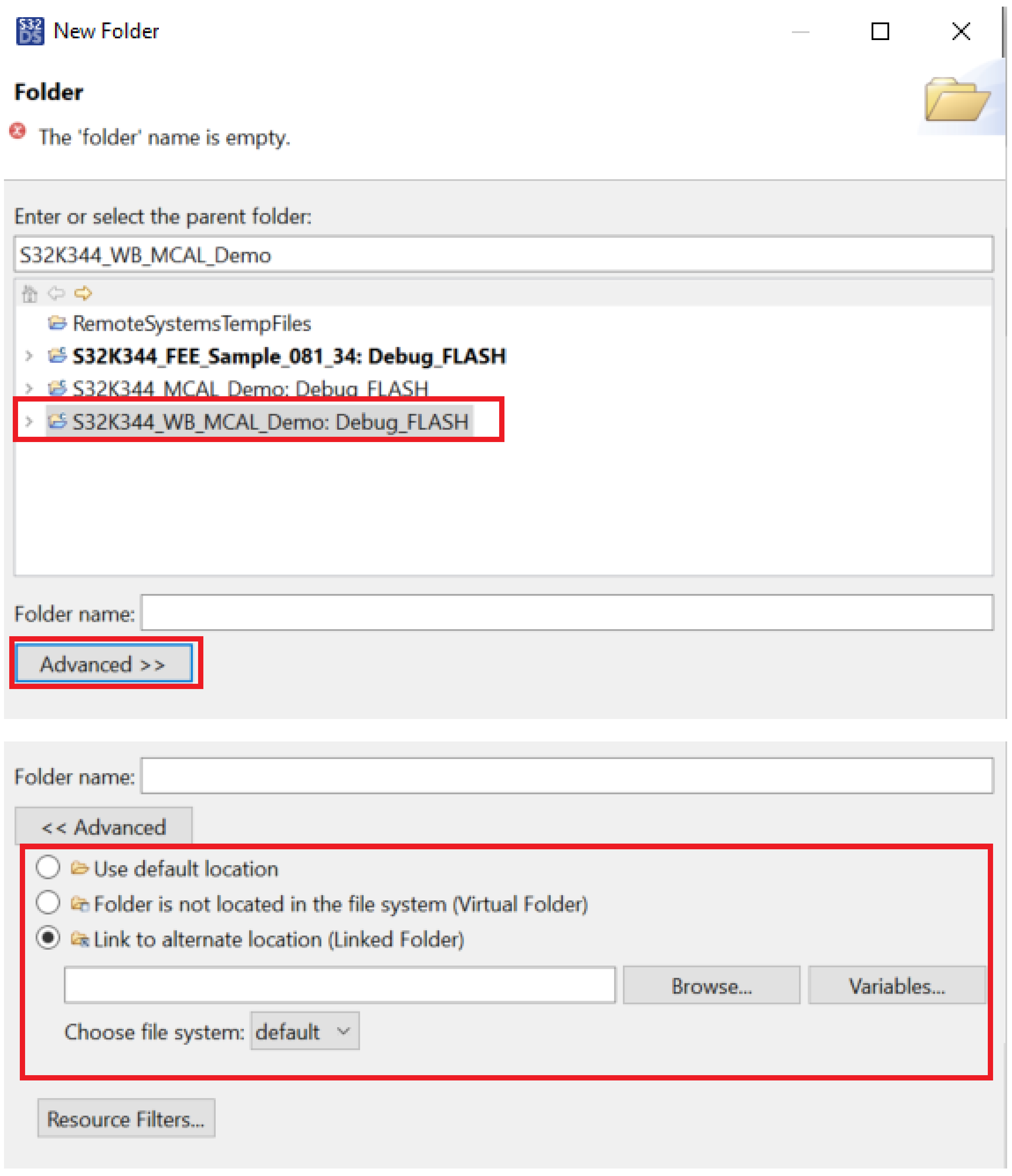
1. Link EB Tresos generated configure file for each module. About this, we have many solutions except link method. For example you can copy the generated folder to our project folder. You also can specify the generate path to our project directly in the EB Tresos setting. Link strategy can keep two projects (EB Tresos configure project and S32DS project) independently.

   Right click project name, `New` -> `Folder`.



S32K344_MCAL_Demo: Debug_FLASH
Binaries
Includes
FreeRTOS_S3
MCAL
Project_Setti
generated/ir
generated/s
New
Go Into
Open in New Window
Show In Alt+Shift+W
Show in Local Terminal
Copy Ctrl+C
Project...
File
S32DS Project from Example Ctrl+Alt+E
S32DS Library Project Ctrl+Alt+L
S32DS Application Project
File from Template
Folder
Create a new folder resource
* main implementation: use this 'C'

Select your project, Expand `Advances`, select `Link to alternate location(Linked Folder)`, browse the generate folder. Default path is under EB Tresos configure project, then `output\generated`.


New Folder

Folder

The 'folder' name is empty.

Enter or select the parent folder:

S32K344_WB_MCAL_Demo

RemoteSystemsTempFiles

S32K344_FEE_Sample_081_34: Debug_FLASH

S32K344_MCAL_Demo: Debug_FLASH

S32K344_WB_MCAL_Demo: Debug_FLASH

Folder name:

Advanced >>

Folder name:

<< Advanced

( ) Use default location

( ) Folder is not located in the file system (Virtual Folder)

[✓ Link to alternate location (Linked Folder)]

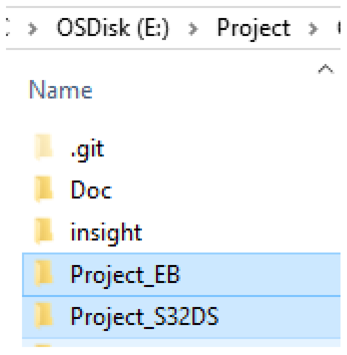
Browse... Variables...

Choose file system: default

Resource Filters...

After you browser, click `Finish`. Then you can find it under your project tree. The icon is different with others folder, the linked folder has a arrow icon.

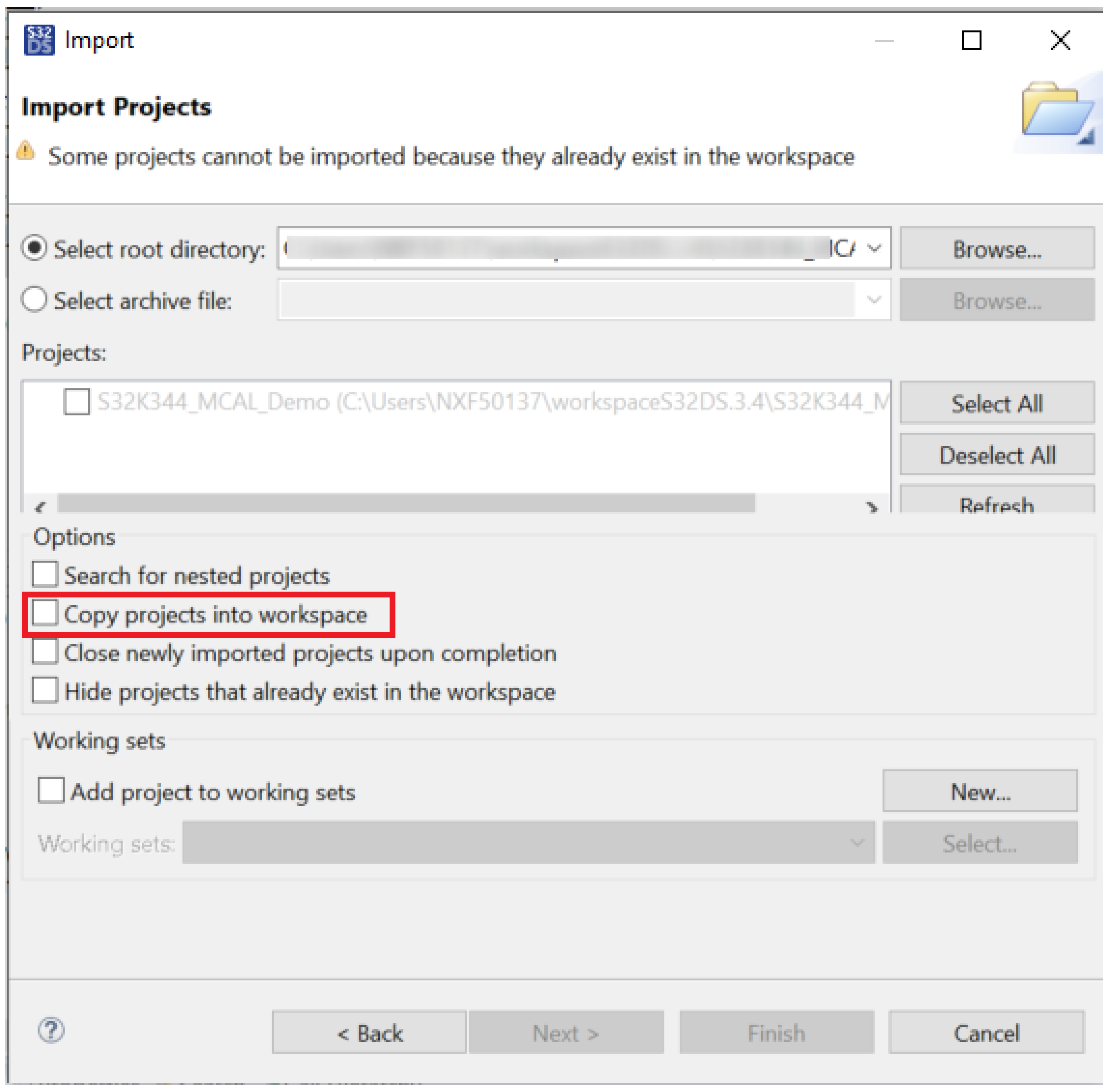
> I suggest you put the EB Tresos configure project at the same root path with S32DS project.


OSDisk (E:) > Project >

Name

.git
Doc
insight
Project_EB
Project_S32DS

> Attention, if you use link method and when you import the project after you create a new workspace, don't select `copy project`, otherwise, the linked folder will cause error, because it can't find the path.


Import

Import Projects

⚠ Some projects cannot be imported because they already exist in the workspace

● Select root directory: [redacted] IC/ — Browse...

○ Select archive file: — Browse...

Projects:

☐ S32K344_MCAL_Demo (C:\Users\NXF50137\workspaceS32DS.3.4\S32K344_M

Select All | Deselect All | Refresh

Options

☐ Search for nested projects

☐ Copy projects into workspace

☐ Close newly imported projects upon completion

☐ Hide projects that already exist in the workspace

Working sets

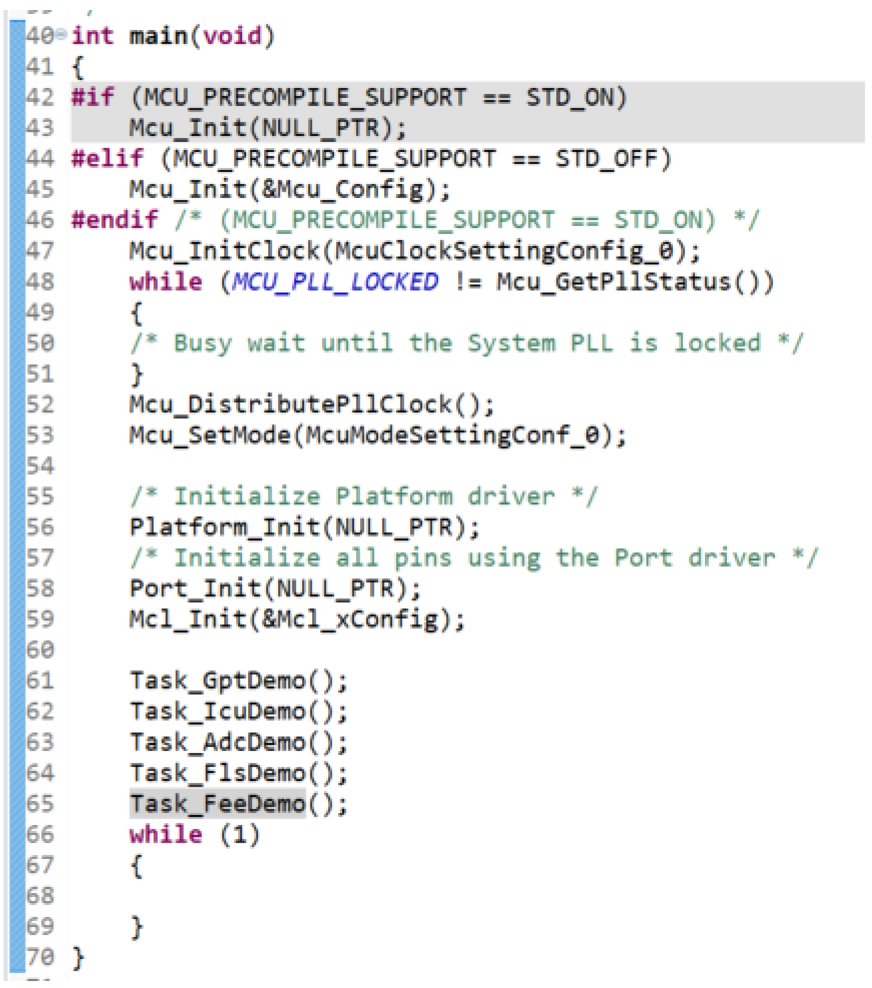
☐ Add project to working sets — New...

Working sets: — Select...

< Back | Next > | Finish | Cancel

6. Add Your Own Code

1. Add your own code in main.c or add intendent file/folder. Don't forget to add the header file path into project property.


```c
int main(void)
{
#if (MCU_PRECOMPILE_SUPPORT == STD_ON)
    Mcu_Init(NULL_PTR);
#elif (MCU_PRECOMPILE_SUPPORT == STD_OFF)
    Mcu_Init(&Mcu_Config);
#endif /* (MCU_PRECOMPILE_SUPPORT == STD_ON) */
    Mcu_InitClock(McuClockSettingConfig_0);
    while (MCU_PLL_LOCKED != Mcu_GetPllStatus())
    {
    /* Busy wait until the System PLL is locked */
    }
    Mcu_DistributePllClock();
    Mcu_SetMode(McuModeSettingConf_0);

    /* Initialize Platform driver */
    Platform_Init(NULL_PTR);
    /* Initialize all pins using the Port driver */
    Port_Init(NULL_PTR);
    Mcl_Init(&Mcl_xConfig);

    Task_GptDemo();
    Task_IcuDemo();
    Task_AdcDemo();
    Task_FlsDemo();
    Task_FeeDemo();
    while (1)
    {

    }
}
```

2. Done. Till now, you have completed the integration work, you can start compiling and debugging.