How to make Unison kernel application for any STM32 board

1. Creating STM32CubeMX application

1.1 Description

This step describes how to create an application for Unison Kernel Expansion Package for any STM32 board.

1.2 Prepare basic files for an application

1.2.1 Create your board directory under “Projects”. For example it will be “Projects/STM32F429I-DISCO”.

1.2.2 Create your application directory under “Projects/{board_name}/Applications/UnisonKernel/”. For example it will be “Projects/STM32F429I-DISCO/Applications/UnisonKernel/UnisonKernel_myapp”.

1.2.3 Copy all files from the directory “{Expansion_path}/GenericResources/” to the directory created in the previous item.

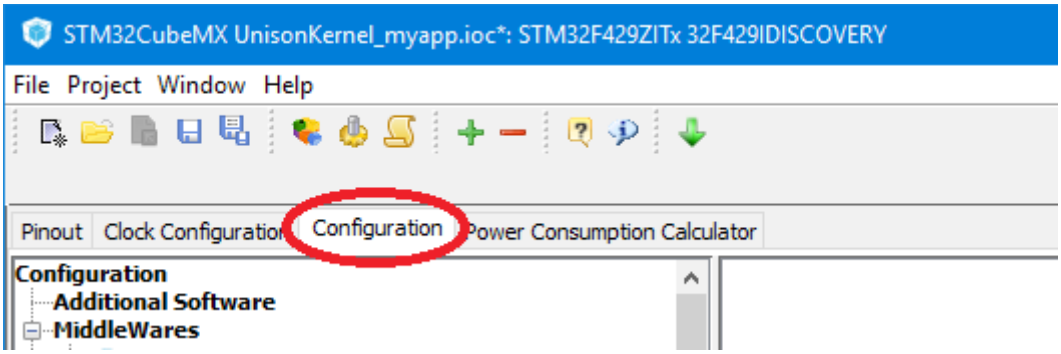
1.2.4 Open the file “.extSettings” and change the define “__ARMCORTEXM7__” to appropriate your used CPU:

- Cortex-M0/M0+: __ARMCORTEXM0__
- Cortex-M3: __ARMCORTEXM3__
- Cortex-M4: __ARMCORTEXM4__
- Cortex-M4 with FPU: __ARMCORTEXM4F__
- Cortex-M7: __ARMCORTEXM7__

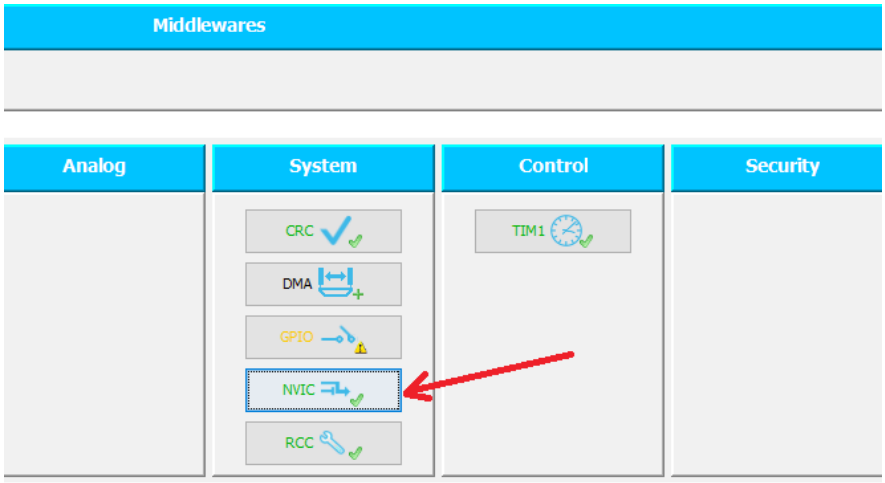
1.3 Make STM32CubeMX application

1.3.1 Open STM32CubeMX software, create and setup an application for your board using STM32CubeMX manual. If you want to indicate Unison kernel scheduler work, setup a LED. If you want to see output messages on serial port, setup an UART.

1.3.2 Open the tab “Configuration”

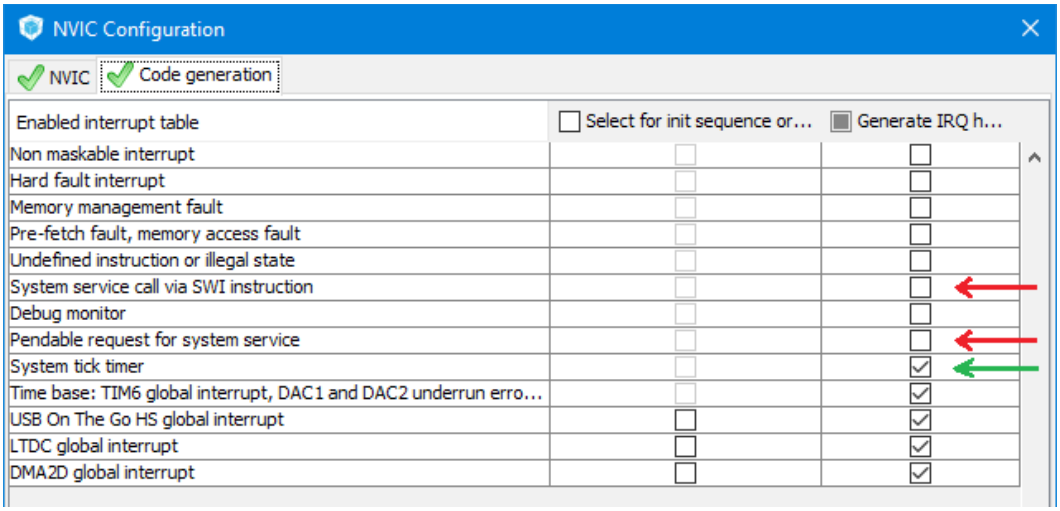


and in the right window under “Middlewares->System” click the “NVIC” button.



Then open the tab “Code generation” and make sure that check-boxes are enabled/disabled for the following interrupts:

- System service call via SWI instruction (SVCall_IRQn) disabled
- Pendable request for system service (PendSV_IRQn) disabled
- System tick timer (SysTick_IRQn) enabled



1.3.3 Save the STM32CubeMX project to prepared above directory. In our example it will be “Projects/STM32F429I-DISCO/Applications/UnisonKernel/UnisonKernel_myapp”.

1.3.4 Generate the application by STM32CubeMX and open it in the chosen IDE. You will see the project with all necessary files: generated by STM32CubeMX and Unison kernel as a third party middleware.

1.3.5 Open the file “unison_app_config.h” and change HAL include header to appropriate your used CPU. Setup the other parameters in this file if you need.

1.3.6 Open the file “unison_kernel_start.c” and set LED from item #1.3.1 as scheduler indicator in the function HAL_SYSTICK_Callback() (or comment out LED control, if you won't use it).

1.3.7 In the file “unison_kernel_start.c” setup in “Unison debug output section” appropriate UART from item #1.3.1 for output messages (or set define DEBUG_TO_BUF to “1” in the file “unison_app_config.h”, if you won't use it).

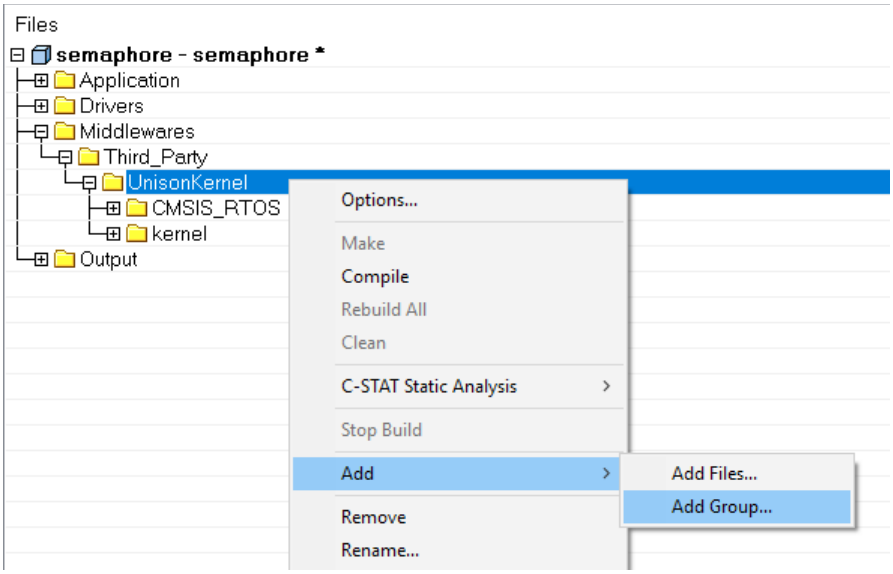
2. Tuning an application for IDE

2.1 Description

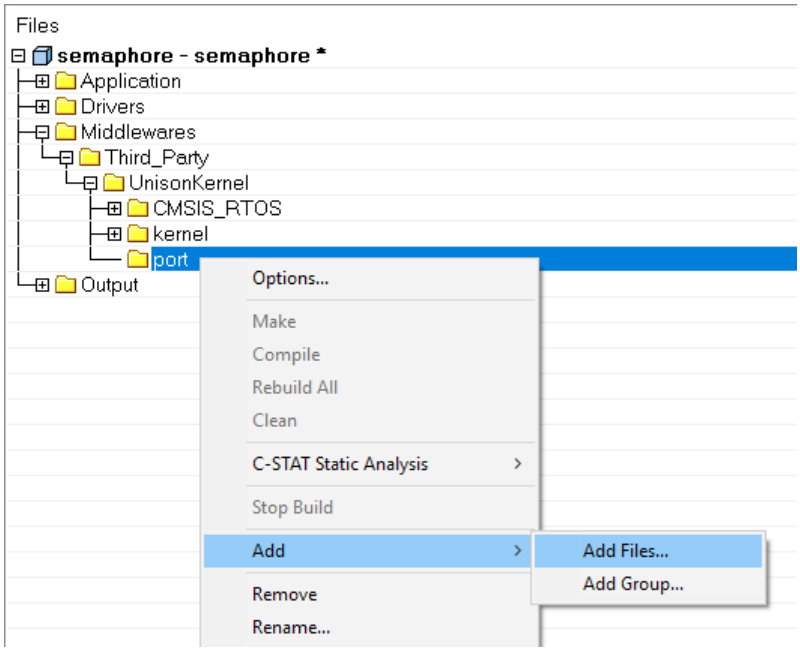
This step describes additional actions which user needs to make after creation an application by STM32CubeMX. These actions are depended on used IDE.

2.2 IAR (EWARM)

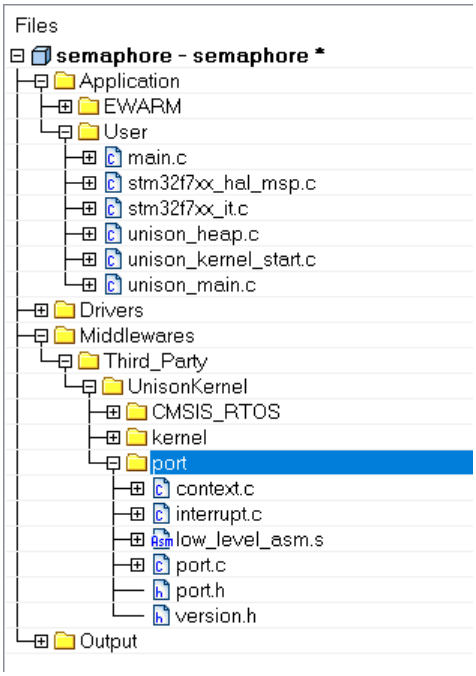
2.2.1 Create the group “port” under “Middlewares/Third_Party/UnisonKernel”.



2.2.2 Add all source files to this new group from the path “{Expansion_path}/Middlewares/Third_Party/UnisonKernel/port/ewarm/”



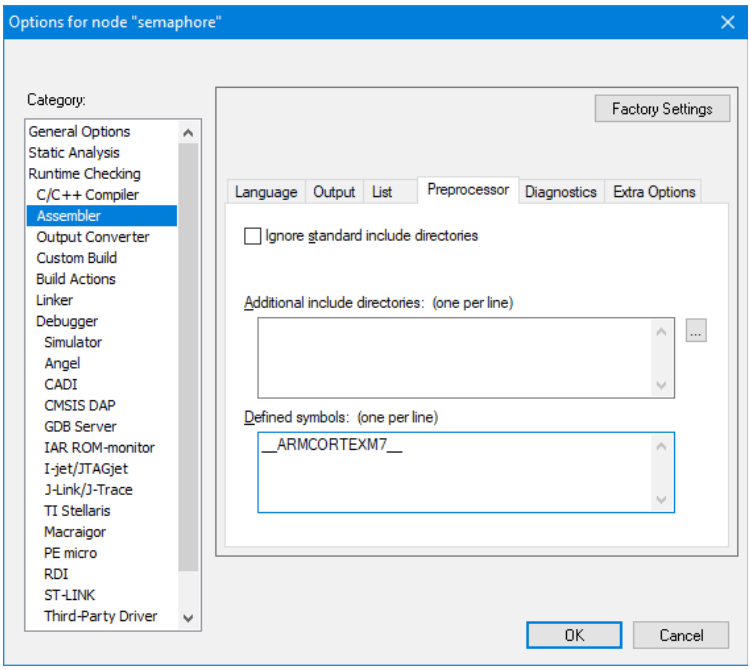
2.2.3 You should get the following structure:



2.2.4 Open the file “main.c” and add Unison header and kernel start function:

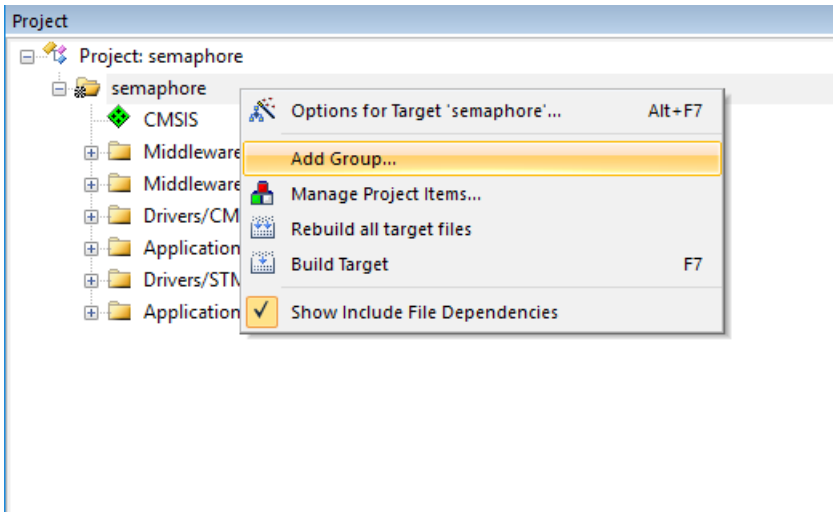
```
. . .
/* USER CODE BEGIN Includes */
#include "unison_kernel_start.h"
/* USER CODE END Includes */
. . .
/* USER CODE BEGIN 2 */
UnisonKernel_Start();
/* USER CODE END 2 */
. . .
```

2.2.5 Open the project options, then select the Category “Assembler” and open the tab “Preprocessor”. Add a build core define to the “Defined symbols” field. This define should be the same as in “Defined symbol” field from “Preprocessor” tab in “C/C++ Compiler” category. In the picture below this define is `"__ARMCORTEXM7__"`.

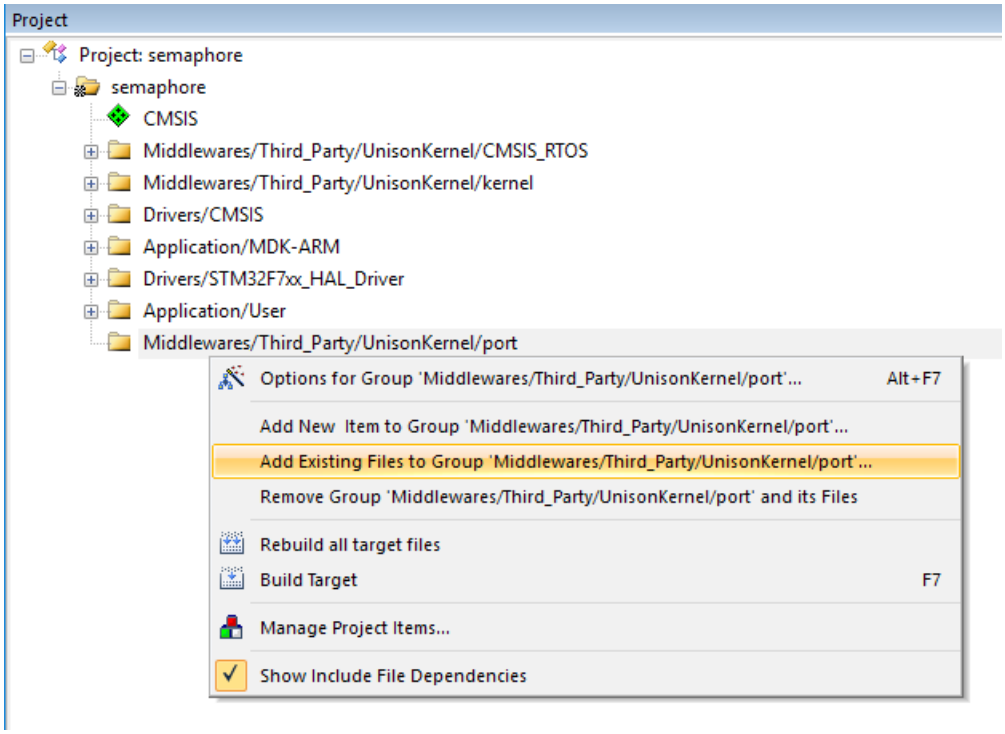


2.3 KEIL (MDK-ARM)

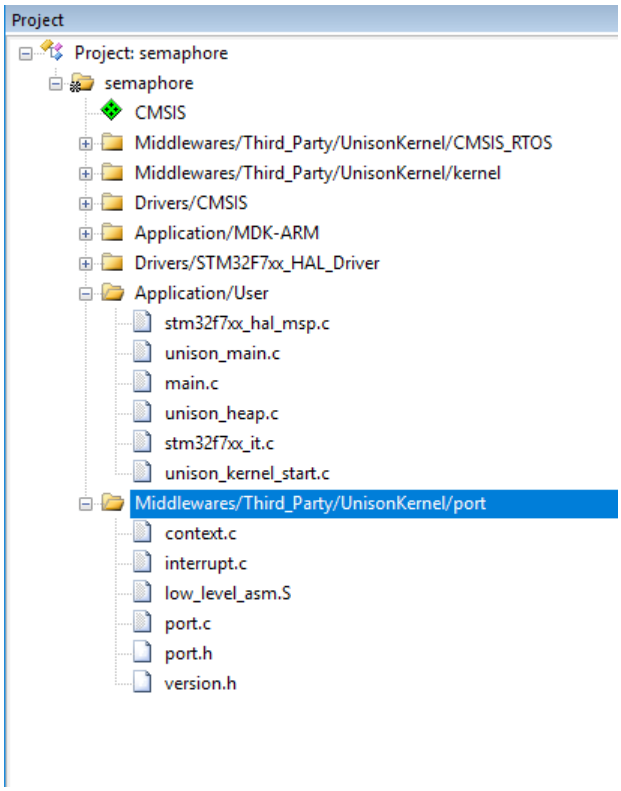
2.3.1 Create new group and rename it to “Middlewares/Third_Party/UnisonKernel/port”.



2.3.2 Add all source files to this new group from the path “{Expansion_path}/Middlewares/Third_Party/UnisonKernel/port/rwmdk/”



2.3.3 You should get the following structure:



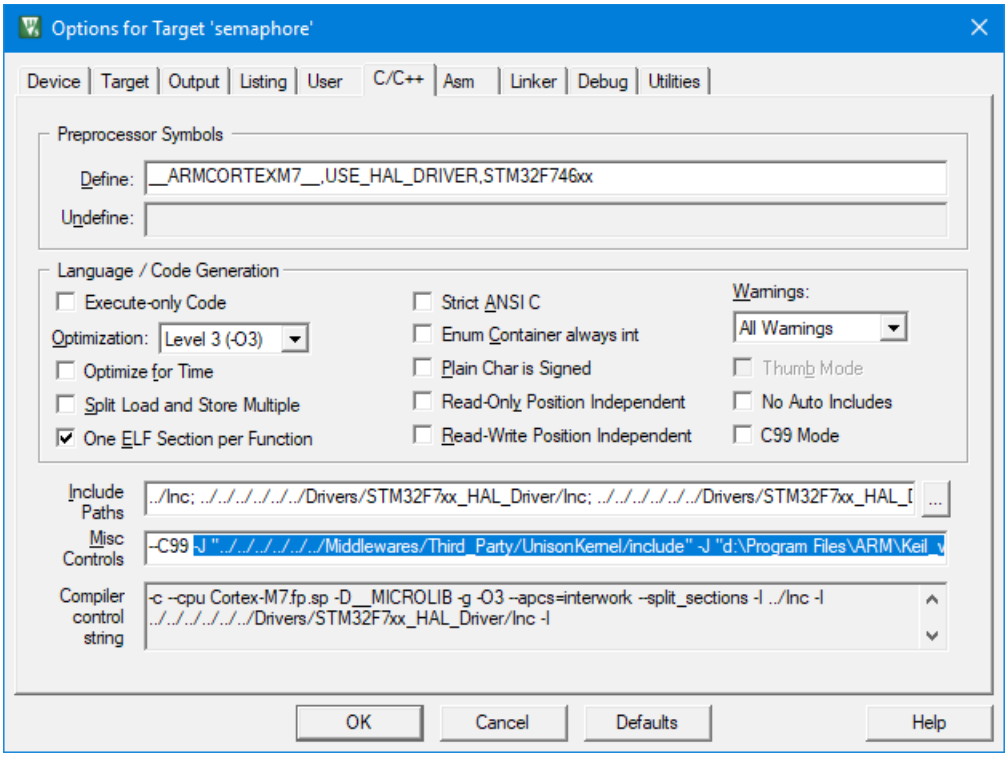
2.3.4 Open the file “main.c” and add Unison header and kernel start function:

```
. . .
/* USER CODE BEGIN Includes */
#include "unison_kernel_start.h"
/* USER CODE END Includes */
. . .
/* USER CODE BEGIN 2 */
UnisonKernel_Start();
/* USER CODE END 2 */
...
```

2.3.5 Keil MDK uses standard compiler headers include paths before any user's headers include paths. Unison has some headers with same names as standards ones. To change the headers search priority open the project options, then open tab “C/C++” and add the following string into the “Misc Controls” field:

```
  -J  "../../../../../../Middlewares/Third_Party/UnisonKernel/include"  -J  "d:\Program Files\ARM\Keil_v5.24a\ARM\ARMCC\include"
```

The second path should be according to your Keil install path. In this example this path is “`d:\Program Files\ARM\Keil_v5.24a\`”. Change it to yours one.



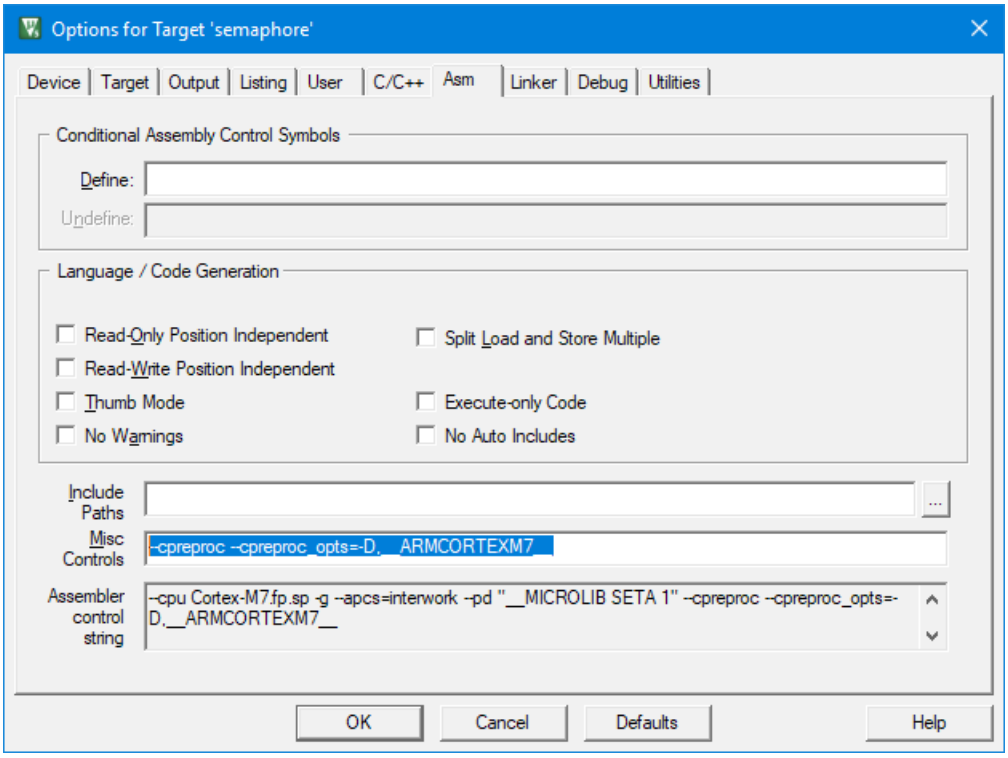
2.3.6 Open the project options, then open tab “Asm” and add the following string into the “Misc Controls” field:

```
--cpreproc
```

Also add to the same field a build core define option (for example “__ARMCORTEXM7__”):

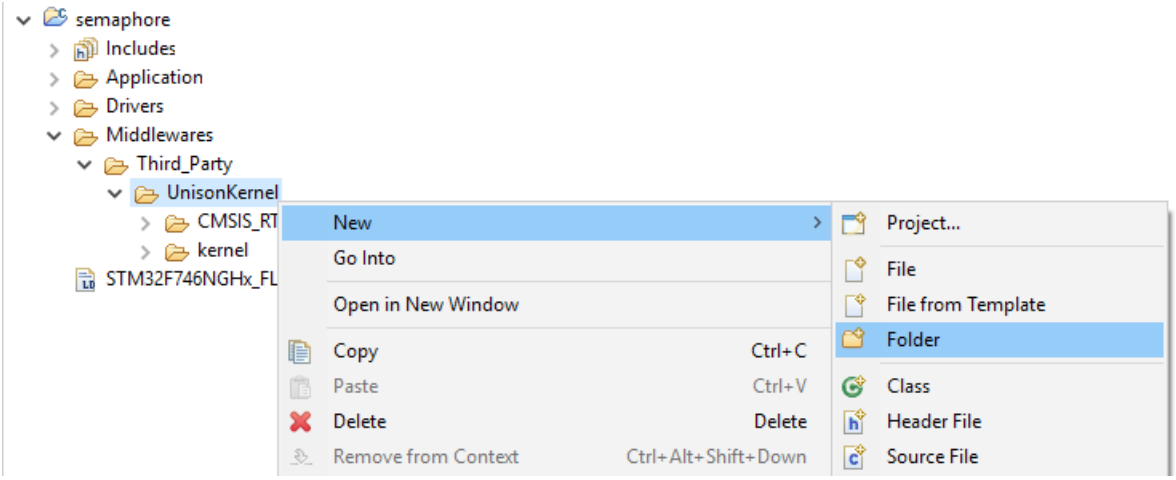
```
--cpreproc_opts=-D,__ARMCORTEXM7__
```

This define should be the same as in “Define” field from “C/C++” tab. In the picture below this define is "__ARMCORTEXM7__".

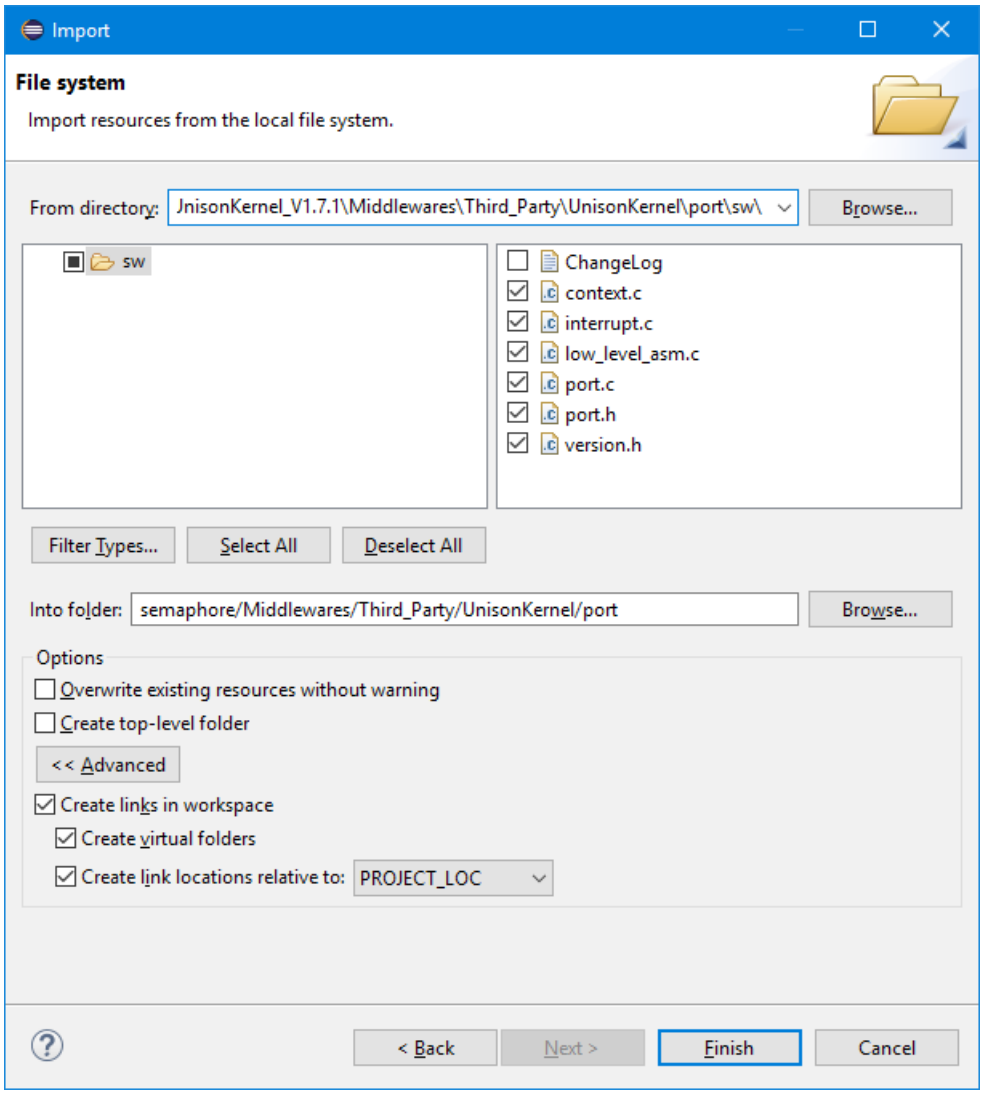


2.4 System Workbench (SW4STM32)

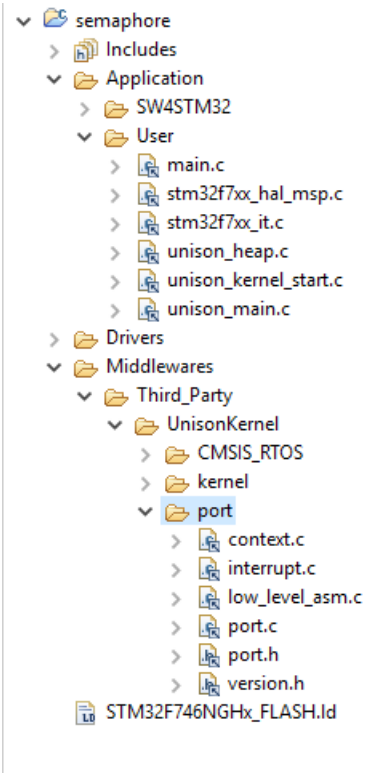
2.4.1 Create new folder “port” under the folder “Middlewares/Third_Party/UnisonKernel/”.



2.4.2 Import to this new folder as links all source files from the path “{Expansion_path}/Middlewares/Third_Party/UnisonKernel/port/sw/”



2.4.3 You should get the following structure:



2.4.4 Open the file “main.c” and add Unison header and kernel start function:

```
. . .
/* USER CODE BEGIN Includes */
#include "unison_kernel_start.h"
/* USER CODE END Includes */
. . .
/* USER CODE BEGIN 2 */
UnisonKernel_Start();
/* USER CODE END 2 */
. . .
```

2.4.5 Open project properties, then open “C/C++ Build -> Settings”. In the tab “Tool Settings” open menu “MCU GCC Compiler -> Dialect” and set C99 standard in the field “Language standard”