

Keil MDK Version 5 Component-based Software Development

Christopher Seidl
Technical Marketing Manager, DSG

Installing the Workshop Material

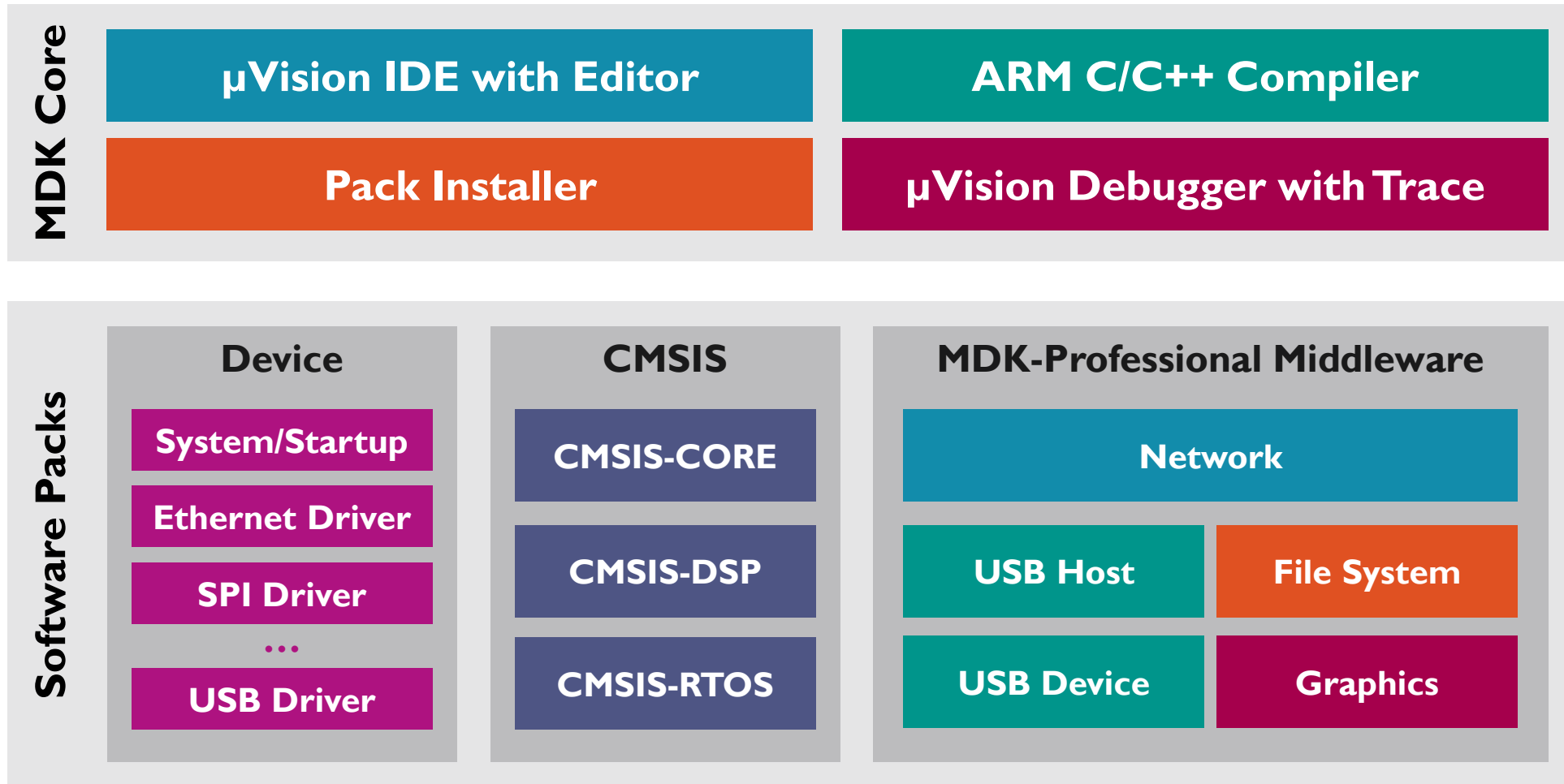
- Required software:

- DAVE™ 3.1.10 Including the latest Updates
- MDK Version 5.11: MDK511.exe
- MDK Add-On Installer: MDK-ARM_Pro_Eval_AddOn.exe
- Infineon XMC400 Device Family Pack: Infineon.XMC4500_DFP.2.0.0.pack

- DAVE Project:

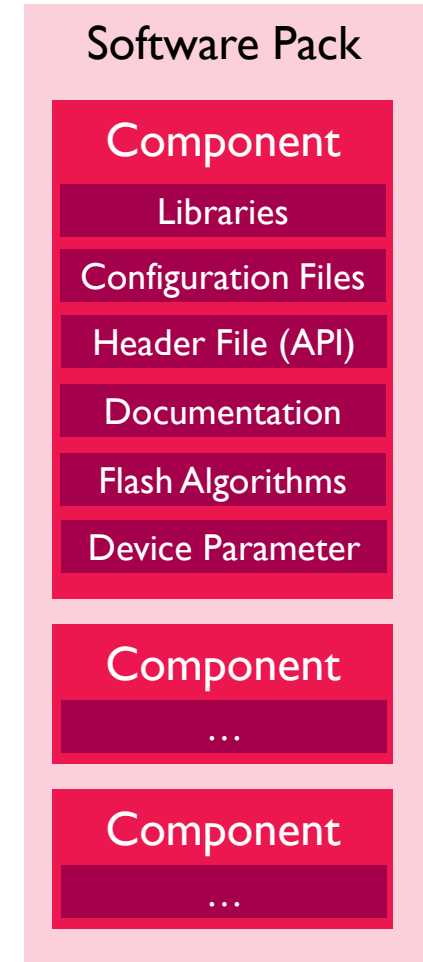
- Please use the one that has been created in the morning session. If you do not have it, use:
- Project\Relax_kit.zip (can be imported into DAVE™ using the DAVE™ import functionality: **File → Import → DAVE Project**; then browse to the project ZIP file)

Software Packs in Keil MDK Version 5



Software Packs

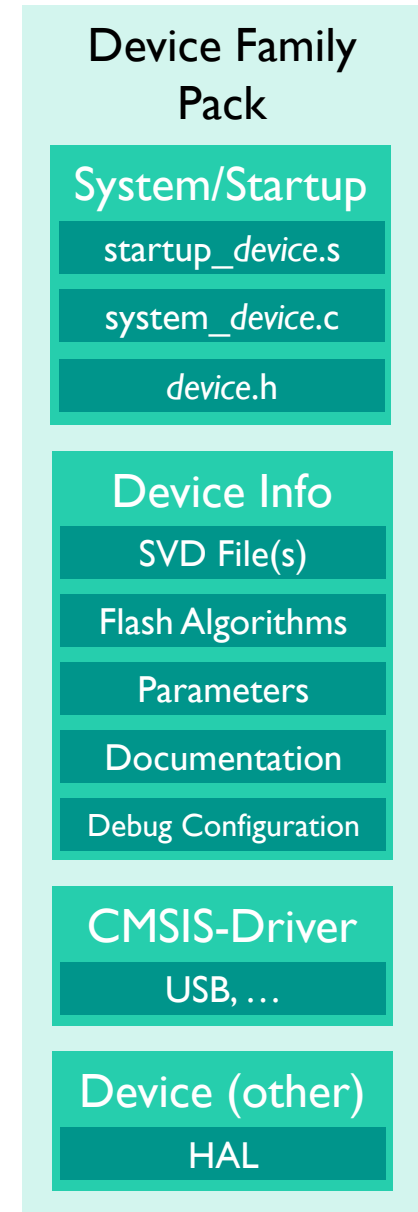
- Software components are delivered in one easy to install Software Pack
- A package description file (PDSC) contains:
 - Supplier information
 - Download URL
 - License
 - Release version
 - Dependencies on processors, devices, tool chains or other components



Device Family Pack

Example: Infineon.XMC4500_DFP

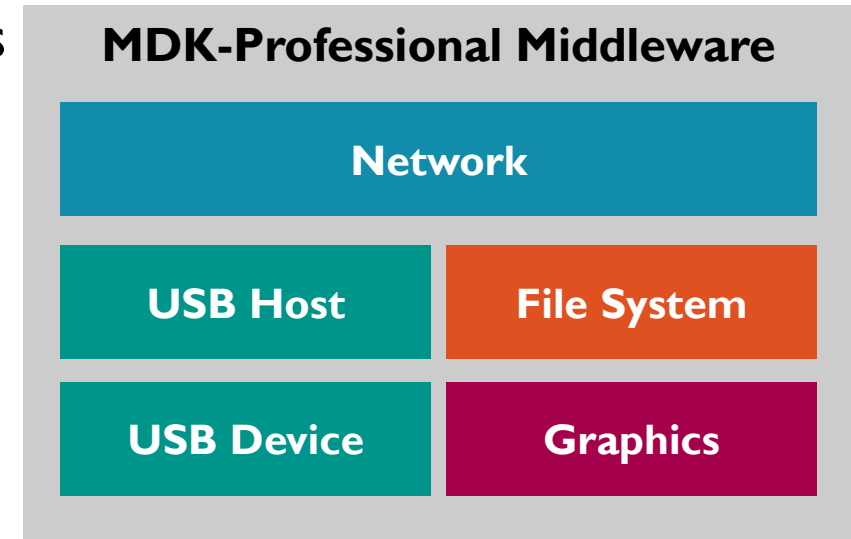
- Device family and devices
 - System and startup code, device header files
 - SVD files
 - Flash algorithms
- Functional grouping of files
- Development board information (if available)
- Usage of source code and library files for:
 - Specific processors
 - Tool chains
- Example Projects
- User Code Templates



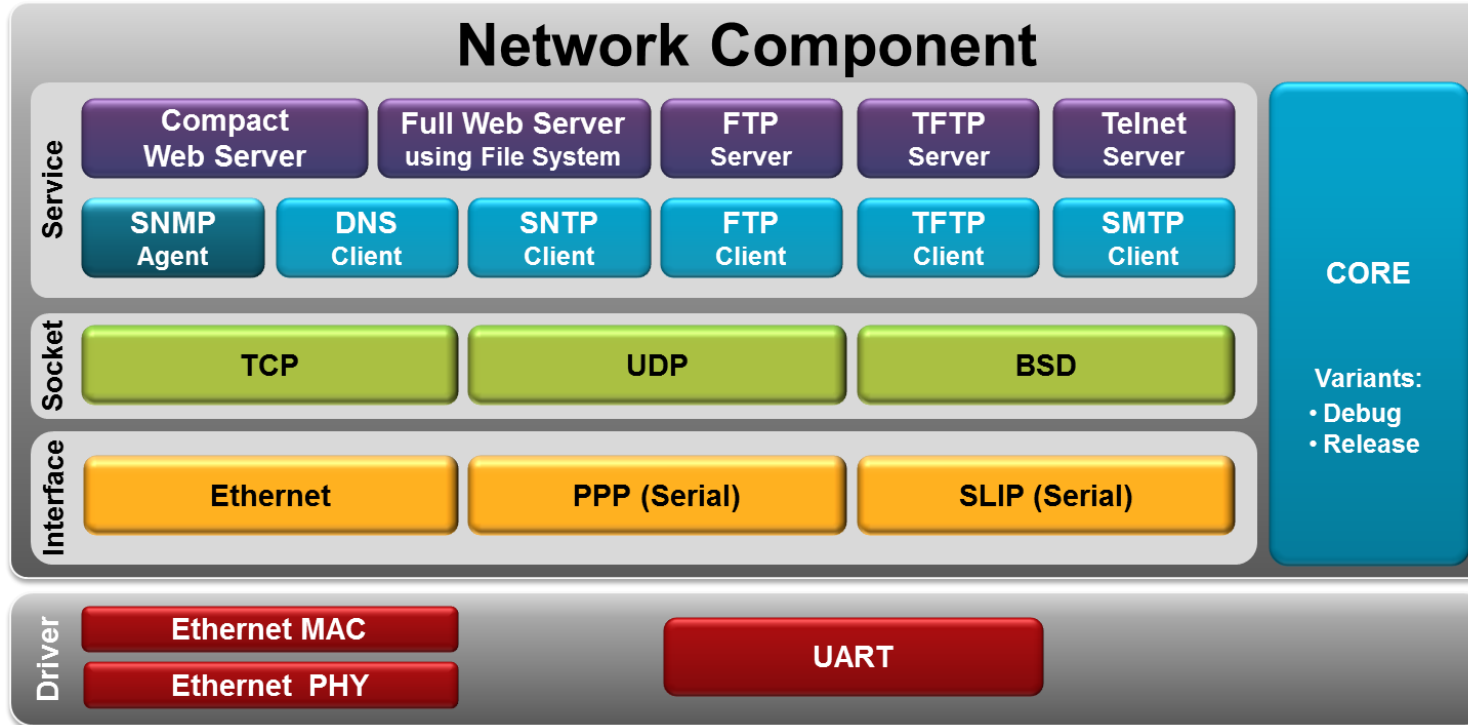
MDK-Professional Middleware



- Specifically designed for ARM Cortex-M devices
- Relies on CMSIS-Driver and CMSIS-RTOS
- Several components available:
 - Network
 - USB (Host & Device)
 - File System
 - Graphics
- Part of MDK-Professional

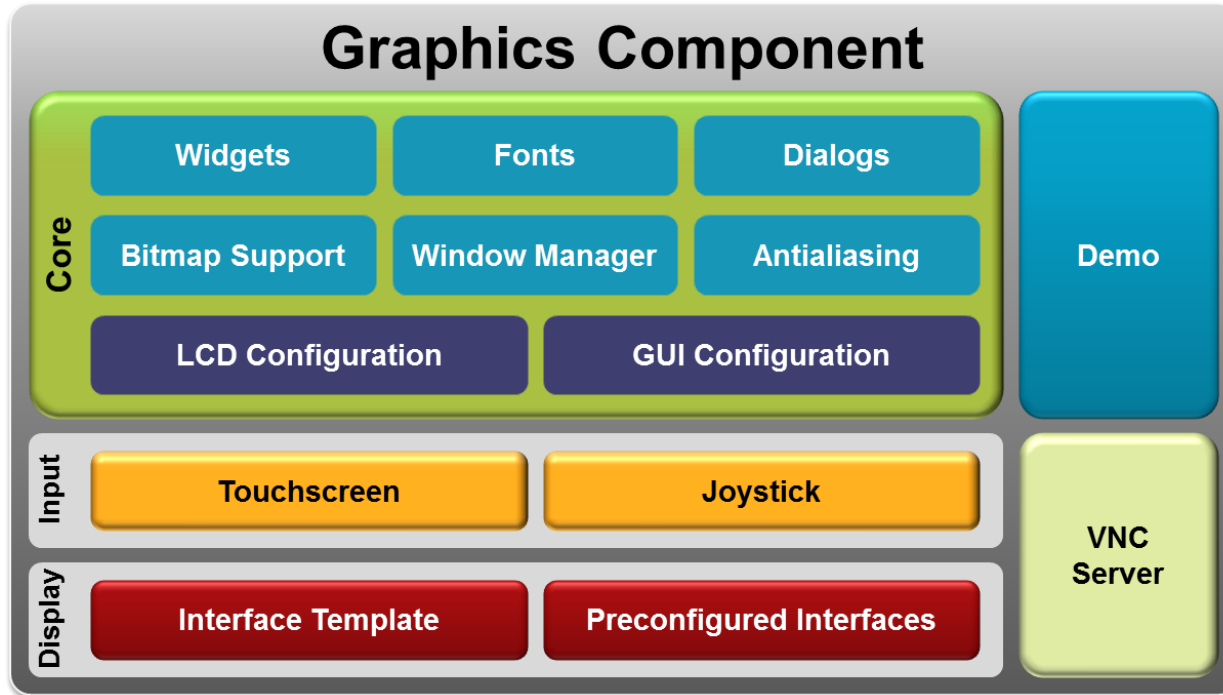


Middleware: Network



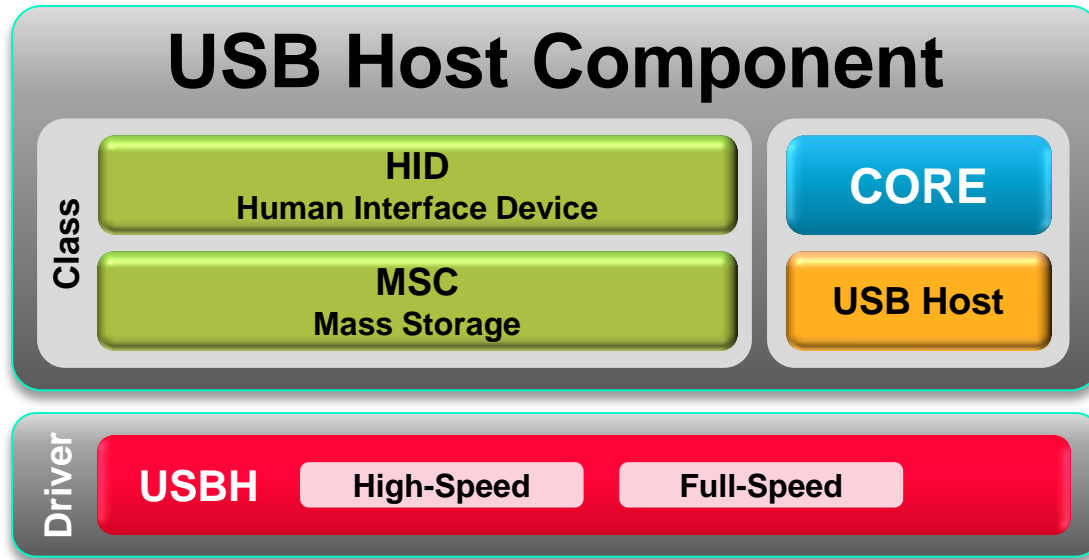
- Add TCP/IP network connectivity via Ethernet or Serial
- Extensive range of service applications included
- Designed for Cortex-M devices with small memory footprint

Middleware: Graphic



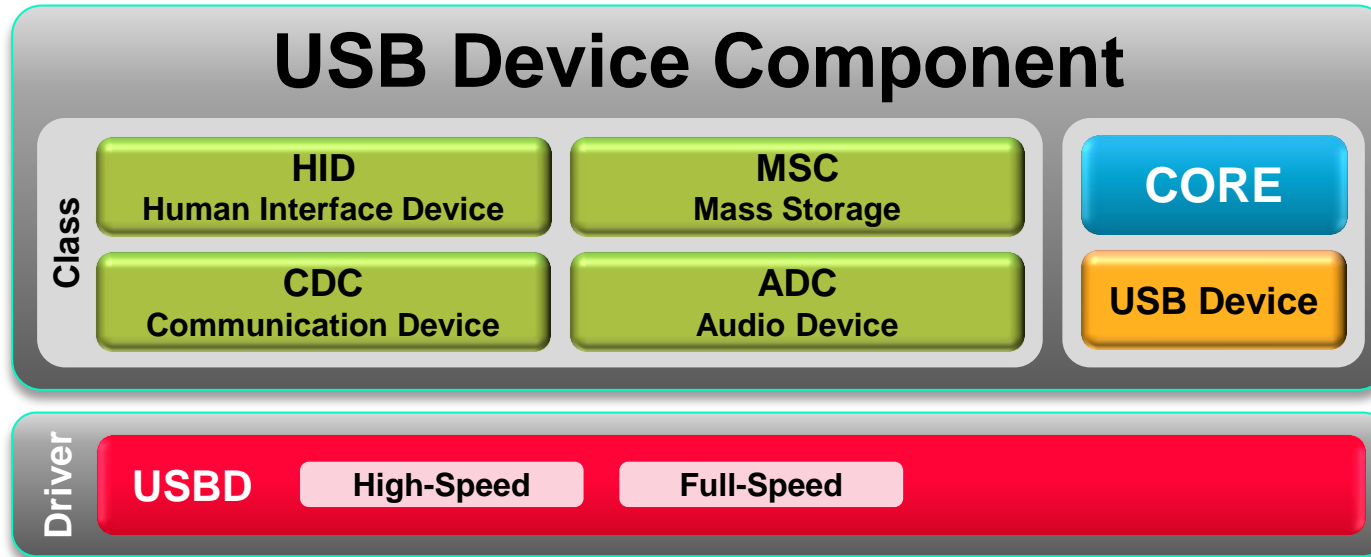
- Add a graphical user interface (GUI) to an application quickly
- Compatible with hundreds of display controllers
- Touch-screen support for many TFT LCDs

Middleware: USB Host



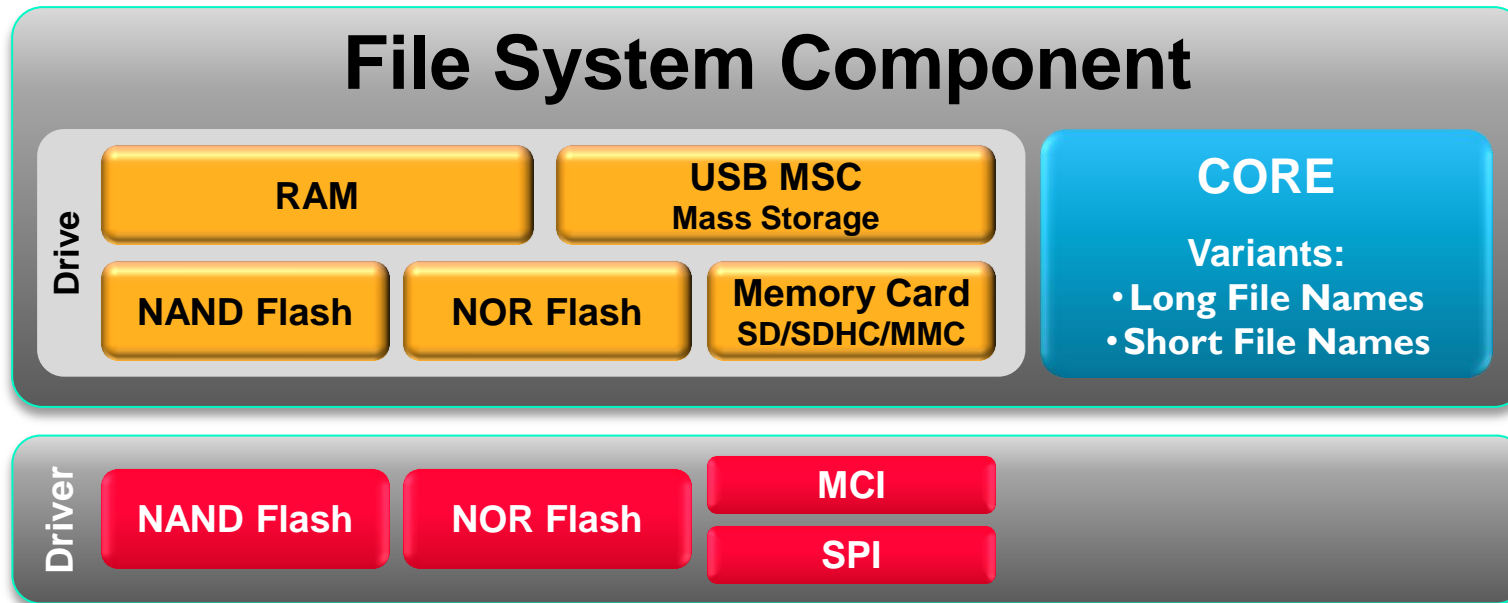
- Connect common USB devices easily
- Support for various popular microcontroller host controllers
- High performance and small footprint
- USB 1.1 Low Speed (1.5 Mbit/s) and Full Speed (12 Mbit/s)
- Supports USB Flash drives and card storage devices

Middleware: USB Device





- Standard USB Device classes supported
- Plug'N'Play: No need to develop a driver for Windows or Linux hosts
- USB 1.1 Low Speed (1.5 Mbit/s) and Full Speed (12 Mbit/s)

Middleware: File System



- Enable applications to work with locally stored data
- Storage devices: ROM, RAM, Flash, and SD/MMC/SDHC
- FAT32 file system supported
- Simultaneous access to multiple storage devices

Web Portal for Software Packs



[Home](#) [Products](#) [Download](#) [Events](#) [Support](#) [Videos](#) [Go](#)

Product Information

- Product Overview
- Supported Microcontrollers
- Shows and Seminars

Technical Support

- Support Knowledgebase
- Application Notes
- Discussion Forum

Software Downloads

- Product Downloads
- File Downloads

Other Information

- Books
- Consultants
- Links

Contact Information

Home / MDK5 Device List / Available Device Packs

Available Device Packs

ARM

- ✓ [Cortex Microcontroller Software Interface](#)

Keil

- ✓ [Device Family Package for ARM Cortex-M4](#)
- ✓ [Device Family Package for Energy Microcontrollers](#)
- ✓ [Device Family Package for Energy Microcontrollers](#)
- ✓ [Device Family Package for Freescale](#)
- ✓ [Device Family Package for Fujitsu Semiconductor FM3 devices](#)

Pack Installer

File Packs Window Help

Device:

Packs Examples

Pack	Action	Description
Keil::Kinetis_K70_DFP	Install	Freescall Semiconductor Kinetis K70 Series
Keil::Kinetis_KExx_DFP	Install	Freescall Semiconductor Kinetis KExx Serie
Keil::Kinetis_KLxx_DFP	Install	Freescall Semiconductor Kinetis KLxx Serie
Keil::Kinetis_KMxx_DFP	Install	Freescall Semiconductor Kinetis KMxx Seri
Keil::LPC1700_DFP	Install	NXP LPC1700 Series Device Support and Ex
Keil::LPC1800_DFP	Install	NXP LPC1800 Series Device Support, Driver
Keil::LPC800_DFP	Install	NXP LPC800 Series Device Support
Keil::MDK-Middleware	Up to date	Keil MDK-ARM Professional Middleware fo
Keil::SAM3_DFP	Install	Atmel SAM3 Series Device Support and Exa
Keil::SAM4_DFP	Install	Atmel SAM4 Series Device Support and Exa
Keil::SAMD20_DFP	Install	Atmel SAMD20 Series Device Support and i
Keil::STM32F0xx_DFP	Install	STMicroelectronics STM32F0 Series Device
Keil::STM32F1xx_DFP	Install	STMicroelectronics STM32F1 Series Device
Keil::STM32F2xx_DFP	Install	STMicroelectronics STM32F2 Series Device
Keil::STM32F3xx_DFP	Install	STMicroelectronics STM32F3 Series Device
Keil::STM32F4xx_DFP	Install	STMicroelectronics STM32F4 Series Device
Keil::STM32L1xx_DFP	Install	STMicroelectronics STM32L1 Series Device
Keil::V2M-MPS2_CMx_BSP	Install	ARM V2M-MPS2 Board Support PACK for C
Keil::XMC1000_DFP	Up to date	Infineon XMC1000 Series Device Support
Keil::XMC4000_DFP	Up to date	Infineon XMC4000 Series Device Support
1.0.2	Remove	Infineon XMC4000 Series Device Support
1.0.1	Remove	Infineon XMC4000 Series Device Support a
lwIP::lwIP	Install	lwIP is a light-weight implementation of th


Ready

ONLINE

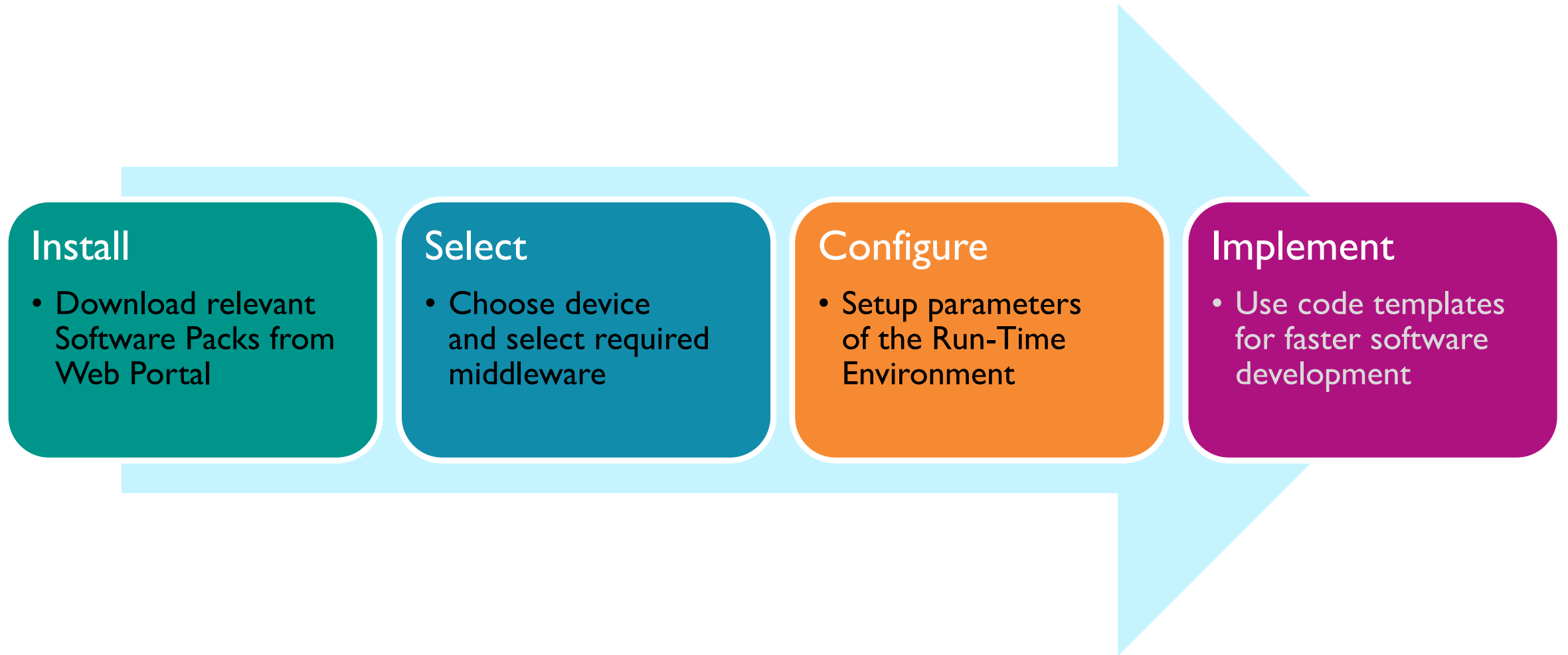
DFP 0.0.1

Search:

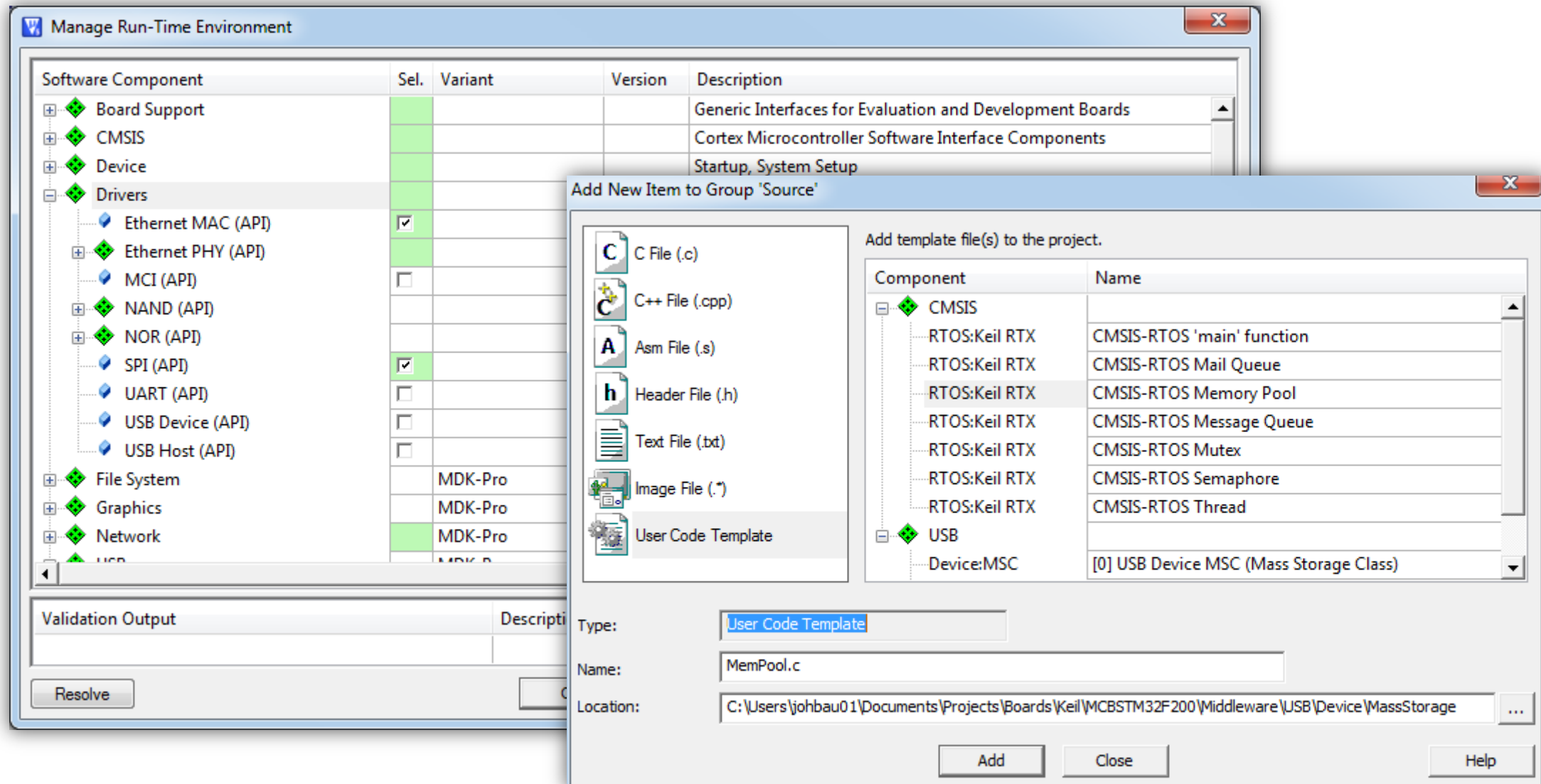
Device	Summary
ARM	10 Devices
Atmel	90 Devices
Energy Micro	235 Devices
Freescall	125 Devices
Infineon	29 Devices
XMC1000 Series	18 Devices
XMC4000 Series	11 Devices
XMC4100-128	ARM Cortex-M4, 80 MHz, 20 kB RAM, 256 kB ROM
XMC4104-64	ARM Cortex-M4, 80 MHz, 20 kB RAM, 128 kB ROM
XMC4104-128	ARM Cortex-M4, 80 MHz, 20 kB RAM, 256 kB ROM
XMC4200-256	ARM Cortex-M4, 80 MHz, 40 kB RAM, 512 kB ROM
XMC4400-256	ARM Cortex-M4, 120 MHz, 48 kB RAM, 512 kB ROM
XMC4400-512	ARM Cortex-M4, 120 MHz, 48 kB RAM, 1024 kB ROM
XMC4402-256	ARM Cortex-M4, 120 MHz, 48 kB RAM, 512 kB ROM
XMC4500-768	ARM Cortex-M4, 120 MHz, 128 kB RAM, 1536 kB ROM
XMC4500-1024	ARM Cortex-M4, 120 MHz, 128 kB RAM, 2048 kB ROM
XMC4502-768	ARM Cortex-M4, 120 MHz, 128 kB RAM, 1536 kB ROM
XMC4504-512	ARM Cortex-M4, 120 MHz, 128 kB RAM, 1024 kB ROM
NXP	40 Devices
Spanion	279 Devices
STMicroelectronics	287 Devices



Keil MDK Workflow using Software Packs



Component View and templates in Keil MDK Version 5



Advantages of Software Components in Keil MDK

Enhanced Productivity

- Convenient selection of software components
- Easy access to documentation
- Code templates and examples to kick-start development

Long-term Project Maintenance

- Software Packs with update facility and version management
- Simplifies the replacement of the target device

Improved Flexibility

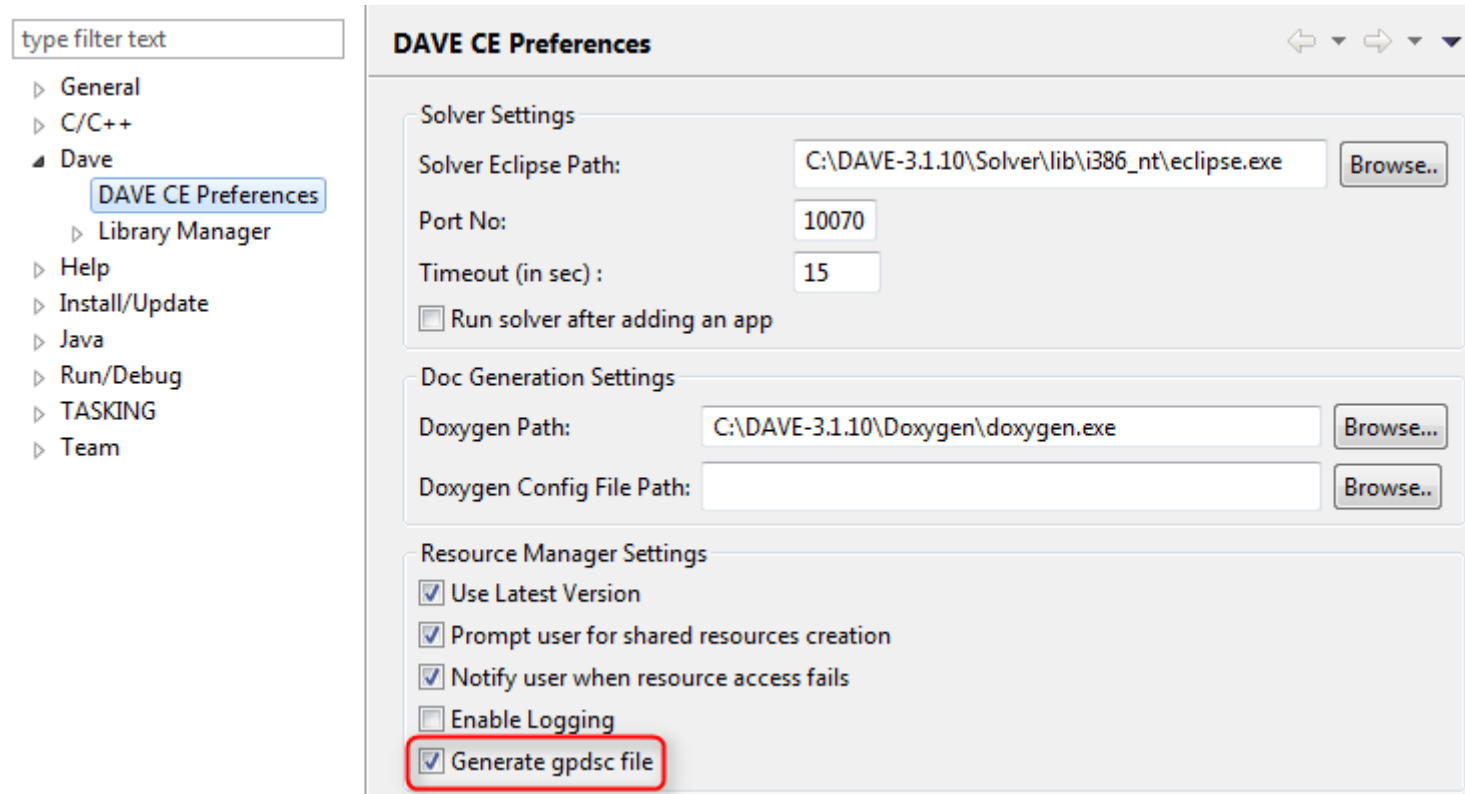
- Support for new devices is provided by Software Packs
- Open standard allows adding third party components

Importing DAVE™ projects in MDK Version 5

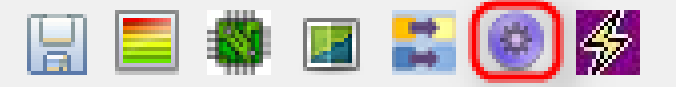

Hands-On

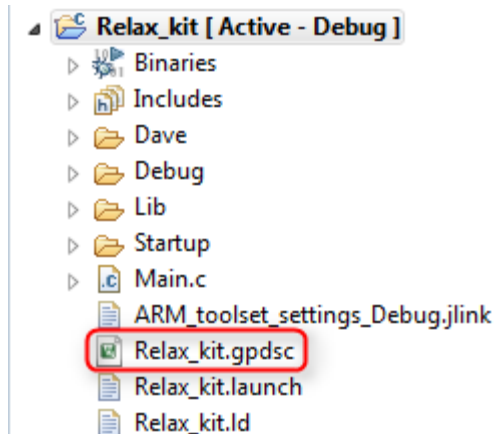
Required Setting for GPDSC Generation

- Go to **Window → Preferences** and **Dave → DAVE CE Preferences**
- Make sure that **Generate gpdsc file** is enabled:



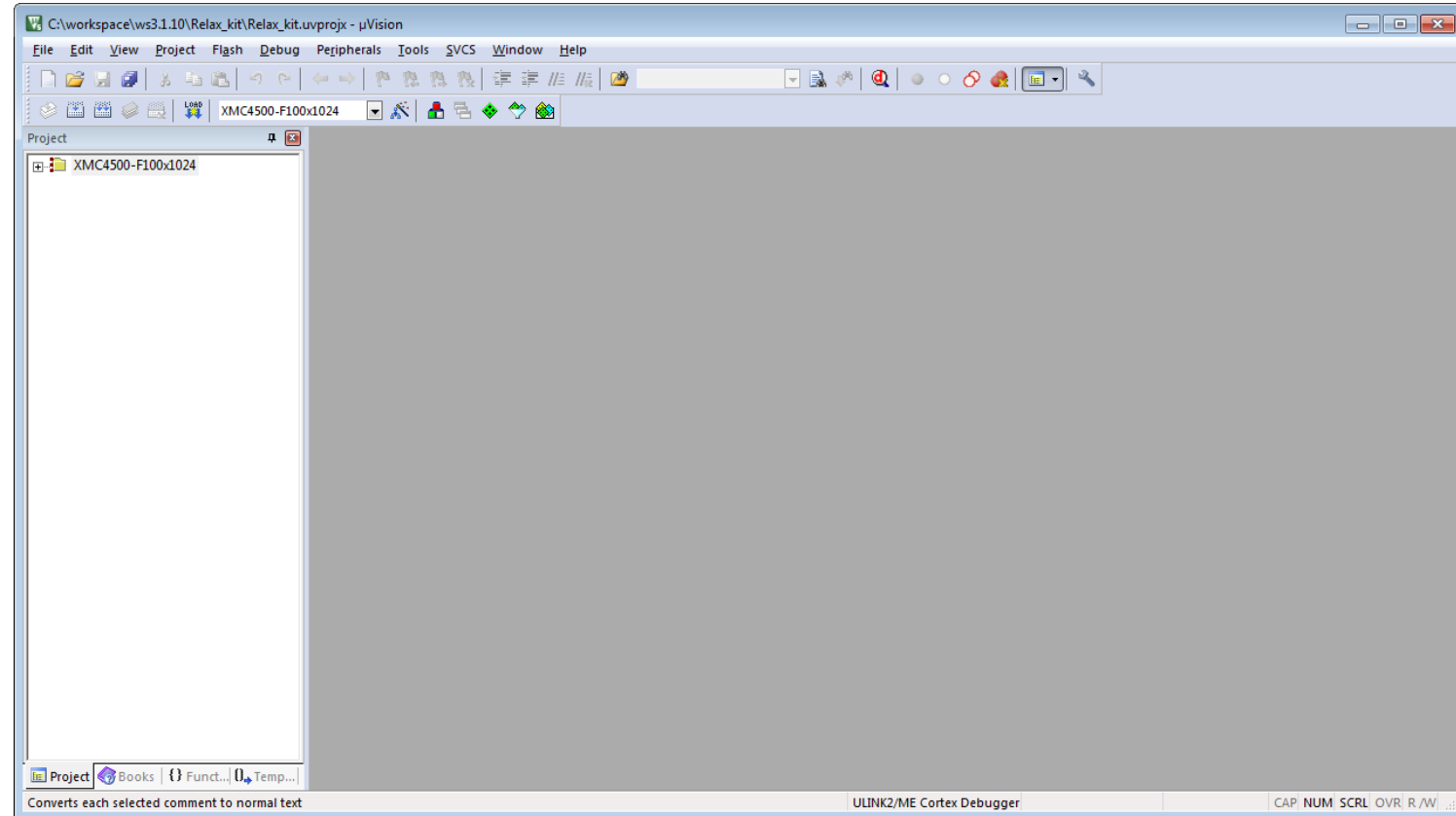
Resource Solving and Code Generation

- The project needs to be solved.
- Press the **Solver** button to run the resource solver: 
- Next, the library code of the DAVE™ Apps needs to be generated. This will also trigger the creation of the GPDSC file.
- Click the **Generate Code** button: 
- Check if the GPDSC file has been generated:



Invoking μ Vision[®] from DAVE[™]

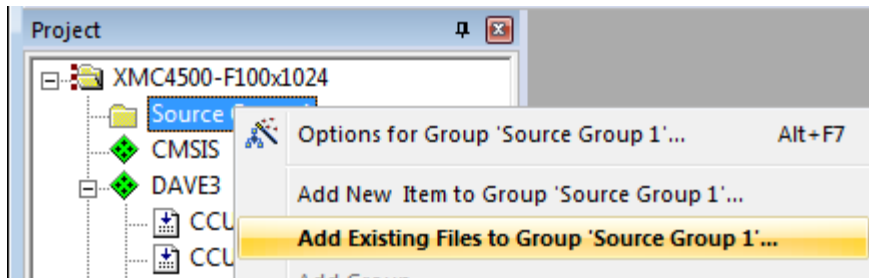
- To import the project to μ Vision, simply double-click the **Relax_kit.gpdsc** file in the **C/C++ Projects** view. μ Vision starts:



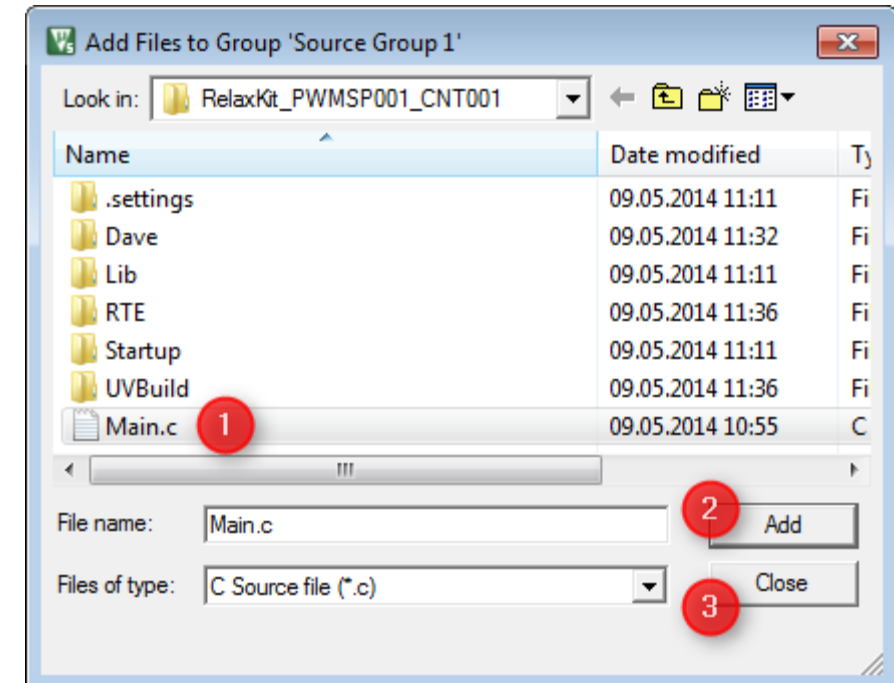
- **Note:** μ Vision will create a project in the same folder as the DAVE[™] project

Add a Main.c File to the μ Vision[®] Project

- There is no main file in the project, so we need to add one manually
- Right click on **Source Group 1**, select **Add Existing files to 'Source Group 1'...**



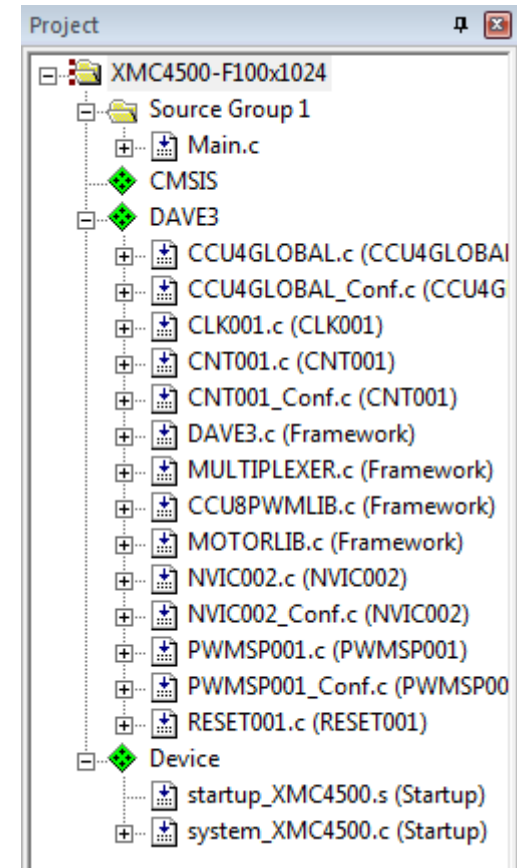
1. Click on **Main.c**
2. Click **Add**
3. Click **Close**



Explore the Project

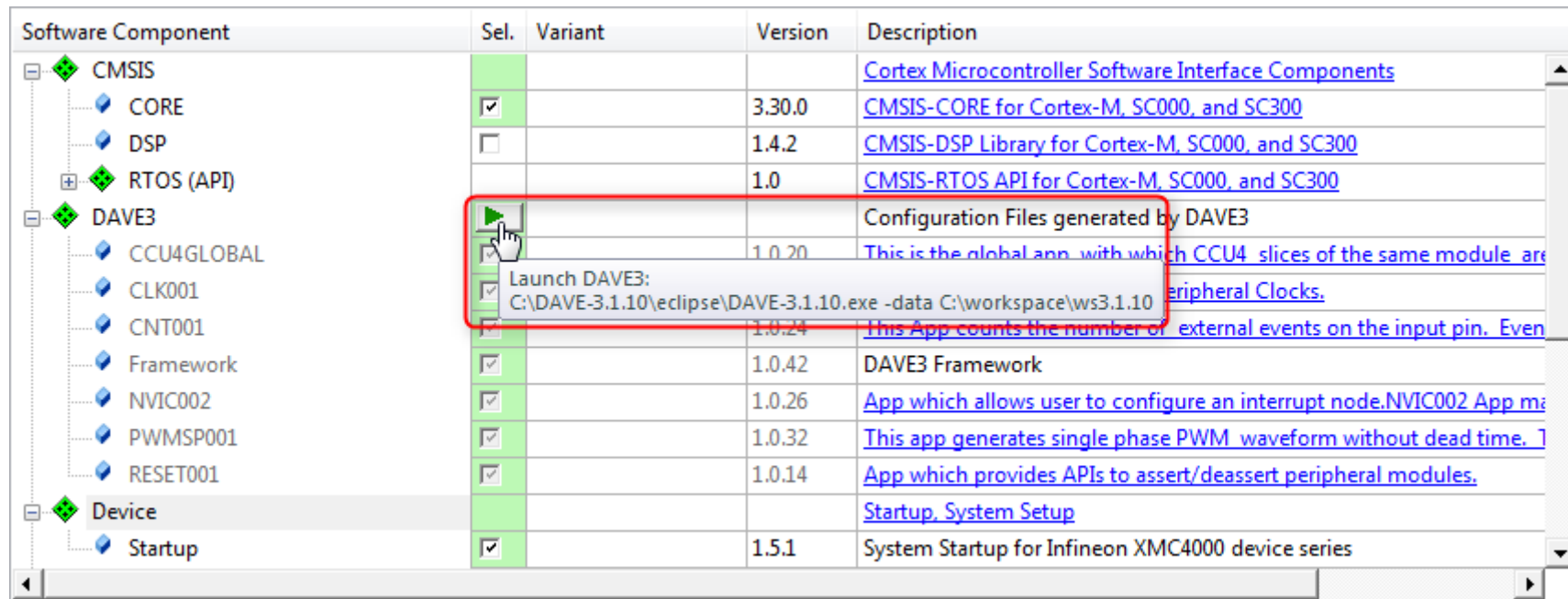
- The **Project** window shows you the software components:
 - **CMSIS** is representing the CMSIS-CORE framework
 - **DAVE3** contains generated files which are part of the GPDSC
 - **Device** incorporates files for device support
- The **Manage Run-Time Environment** window reflects this:

Software Component	Sel.	Variant	Version	Description
[-] CMSIS				Cortex Microcontroller Software Interface Components
[-] CORE	<input checked="" type="checkbox"/>		3.30.0	CMSIS-CORE for Cortex-M, SC000, and SC300
[-] DSP	<input type="checkbox"/>		1.4.2	CMSIS-DSP Library for Cortex-M, SC000, and SC300
[+] RTOS (API)			1.0	CMSIS-RTOS API for Cortex-M, SC000, and SC300
[-] DAVE3	<input checked="" type="checkbox"/>			Configuration Files generated by DAVE3
[-] CCU4GLOBAL	<input checked="" type="checkbox"/>		1.0.20	This is the global app with which CCU4 slices of the sam
[-] CLK001	<input checked="" type="checkbox"/>		1.0.42	App to configure System and Peripheral Clocks.
[-] CNT001	<input checked="" type="checkbox"/>		1.0.24	This App counts the number of external events on the in
[-] Framework	<input checked="" type="checkbox"/>		1.0.42	DAVE3 Framework
[-] NVIC002	<input checked="" type="checkbox"/>		1.0.26	App which allows user to configure an interrupt node.NV
[-] PWMSP001	<input checked="" type="checkbox"/>		1.0.32	This app generates single phase PWM waveform without
[-] RESET001	<input checked="" type="checkbox"/>		1.0.14	App which provides APIs to assert/deassert peripheral mc
[+] Debug				
[-] Device				Startup, System Setup
[-] Startup	<input checked="" type="checkbox"/>		1.5.1	System Startup for Infineon XMC4000 device series



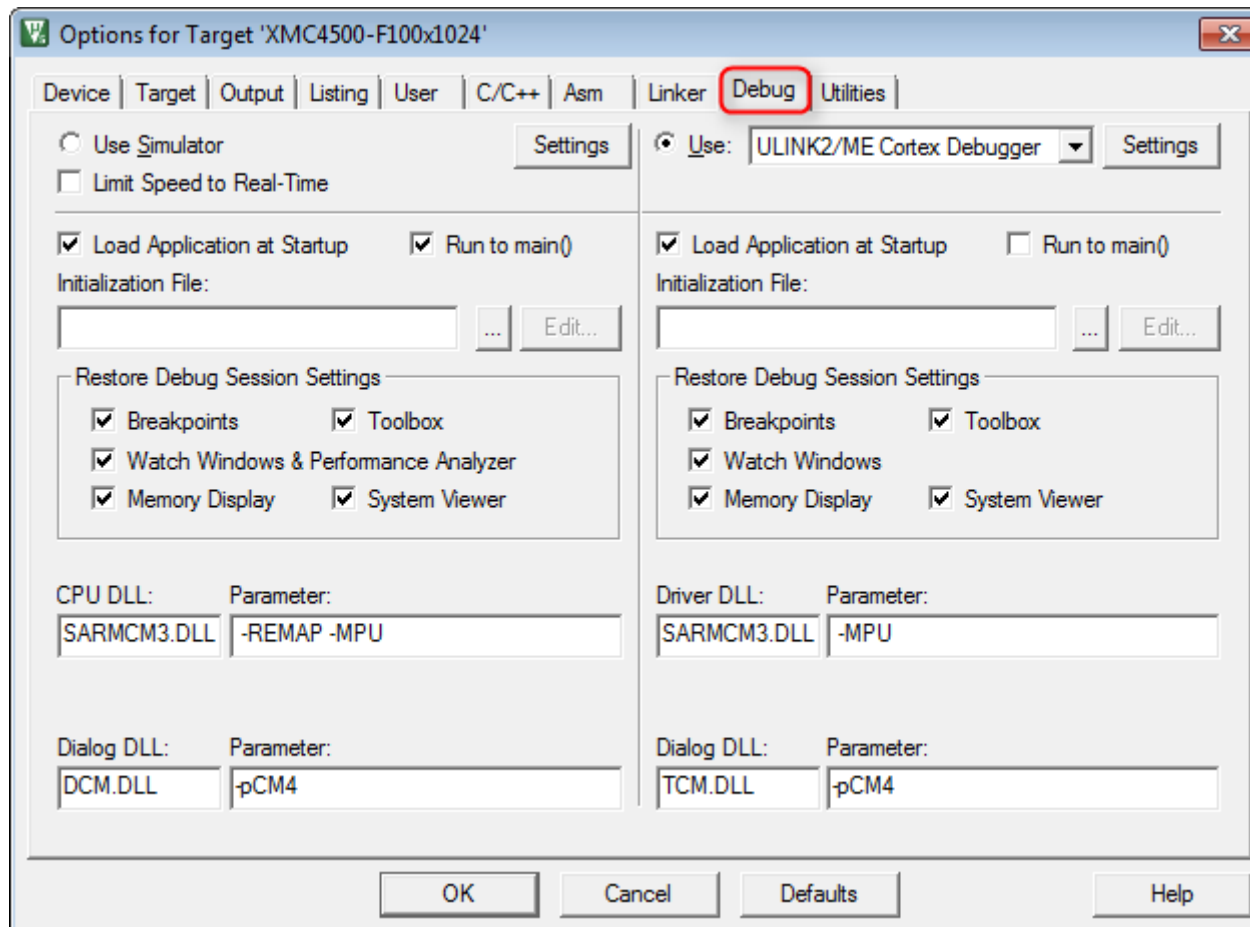
Going Back to DAVE™

- Changing the DAVE™ project (add/remove Apps or adjust App settings) needs to be done in DAVE™.
- The **Play** button in the Manage Run-Time Environment window invokes DAVE™ with the correct project:



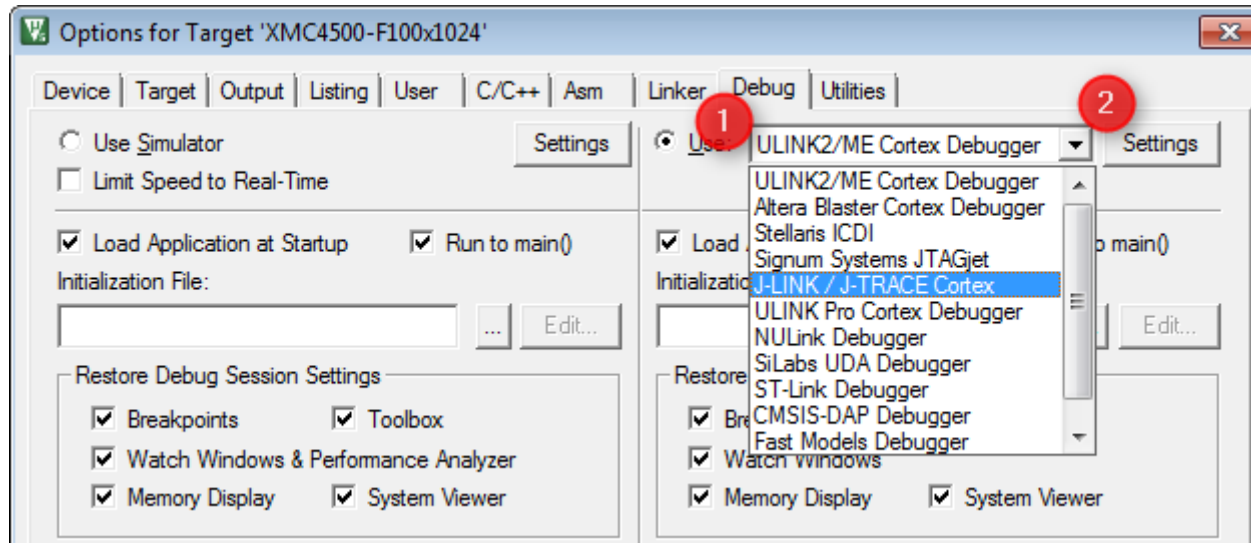
Open Target Options Dialog

- Go to **Project** → **Options for Target 'XMC4500-F100x1024'** (or press ALT+F7) and click on the **Debug** tab:



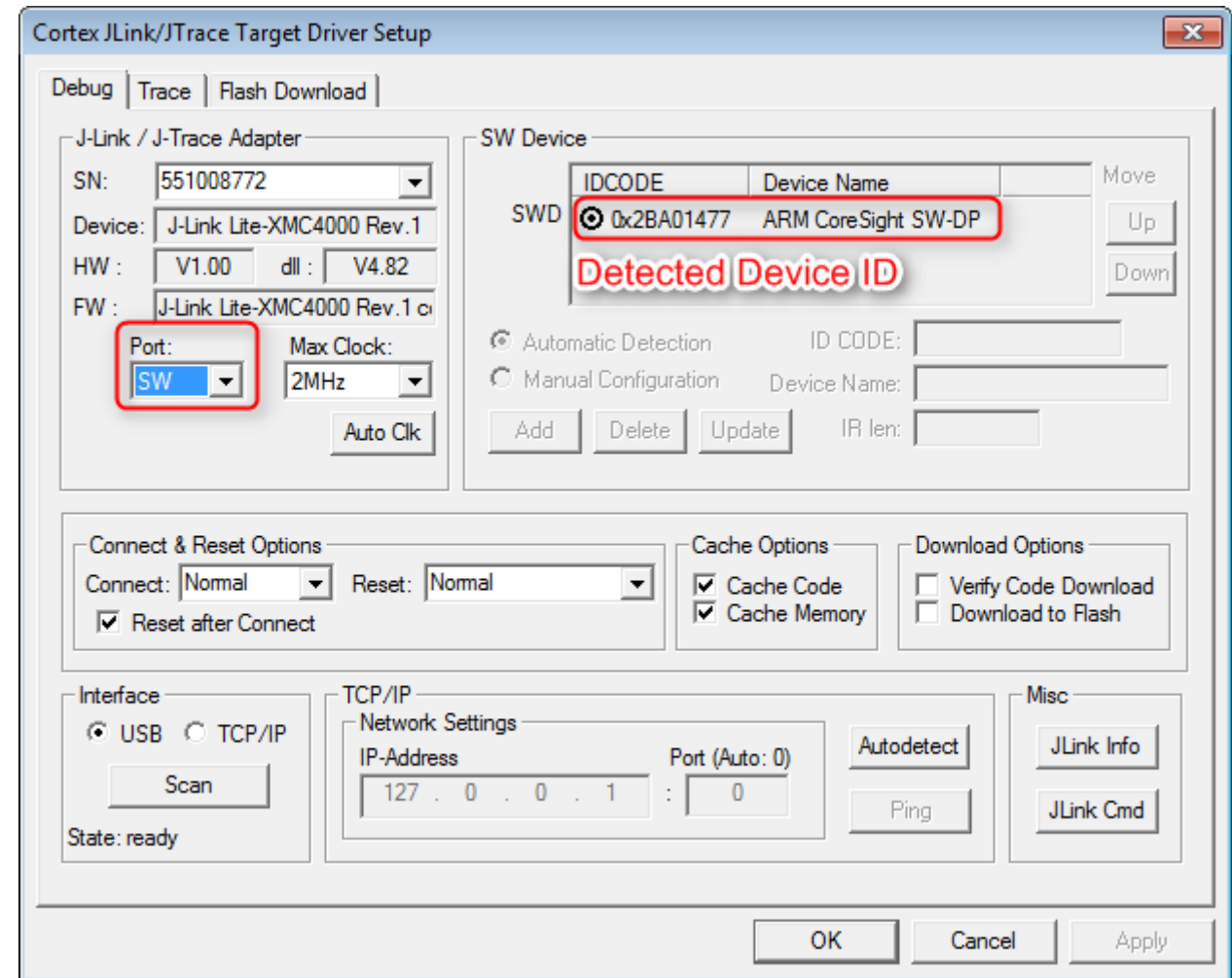
Select the J-LINK Debugger

- The XMC4500 Relax Kit has an integrated J-LINK debug adapter
 1. To change default adapter, click on **ULINK2/ME Cortex Debugger** and scroll down until **J-LINK/J-TRACE Cortex**
 2. Click on **Settings**, to check the connectivity between the target and your PC



Enable the SW Port


- µVision will try to connect to the J-LINK using a JTAG port. This is not available on the Relax Kit
- Click on **Port: JTAG** and set to **SW**
- The connected device will be detected automatically
- Click **OK** twice




Build the Project and Download to Target

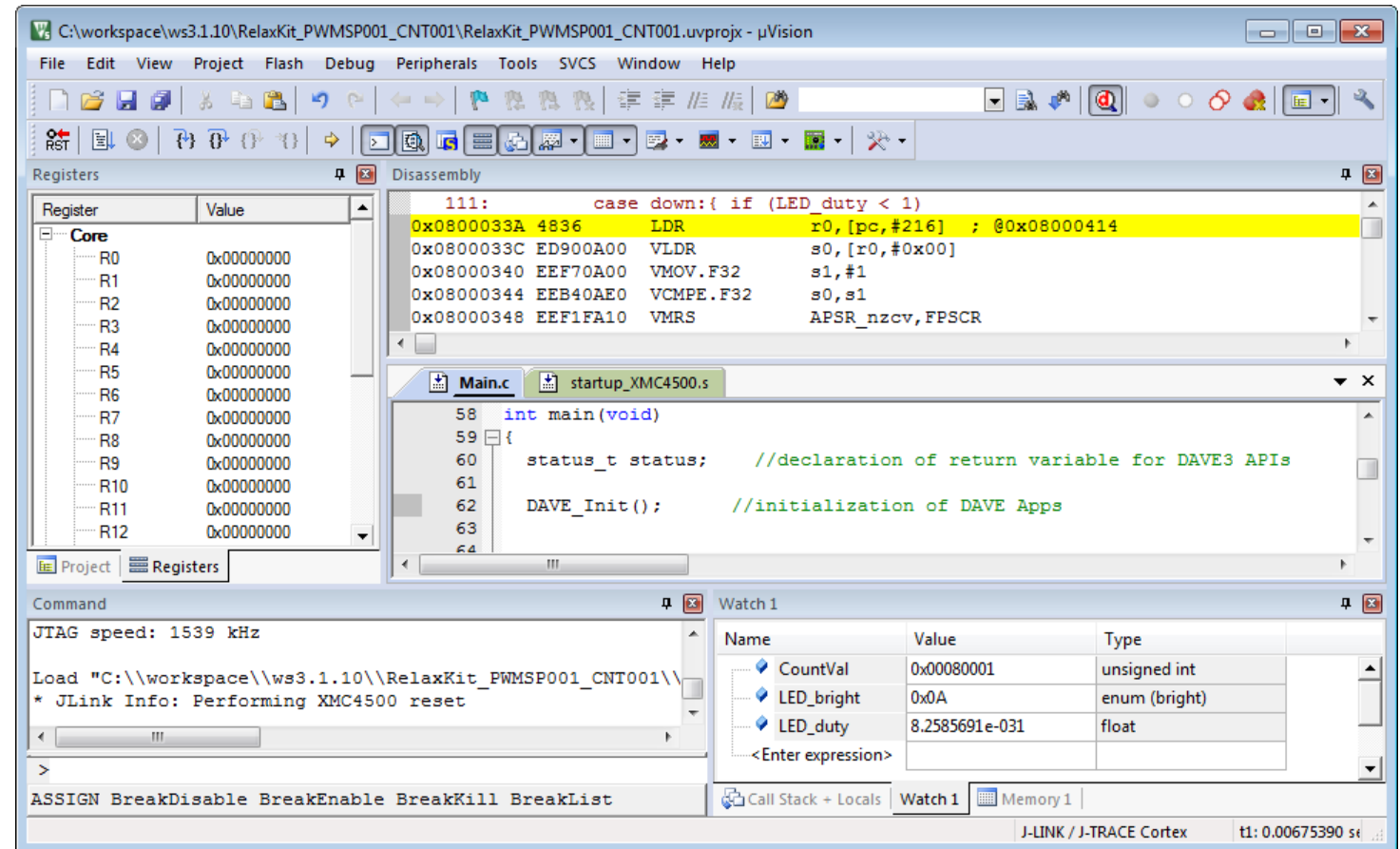
- Go to **Project → Build** (or press F7) to start the build process
- You will see a lot of warnings (#188-D) which can be safely ignored. If you want to suppress these warnings, press **ALT+F7** again, click on the **C/C++** tab and enter in the **Misc Controls** field: `--diag_suppress=188`
- Rebuilding the project (**Project → Rebuild all target files**) will show no warnings

```
Build Output
Build target 'XMC4500-F100x1024'
compiling Main.c...
linking...
Program Size: Code=9800 RO-data=668 RW-data=52 ZI-data=1660
".\UVBuild\RelaxKit_PWMSP001_CNT001.axf" - 0 Error(s), 0 Warning(s).
```

- Go to **Flash → Download** (or click on ) to download the program into the target's Flash memory

Start a Debug Session

- Go to **Debug** → **Start/Stop Debug Session** (or press CTRL+F5) to switch to the μ Vision debugger.
- During the start of the debug session, μ Vision loads the application, executes startup code, and stops at the main C function
- Click **Run**  on the toolbar. The LED connected to P1.1 will start flashing



printf Debugging using the Instrumentation Trace Macrocell (ITM)

CoreSight

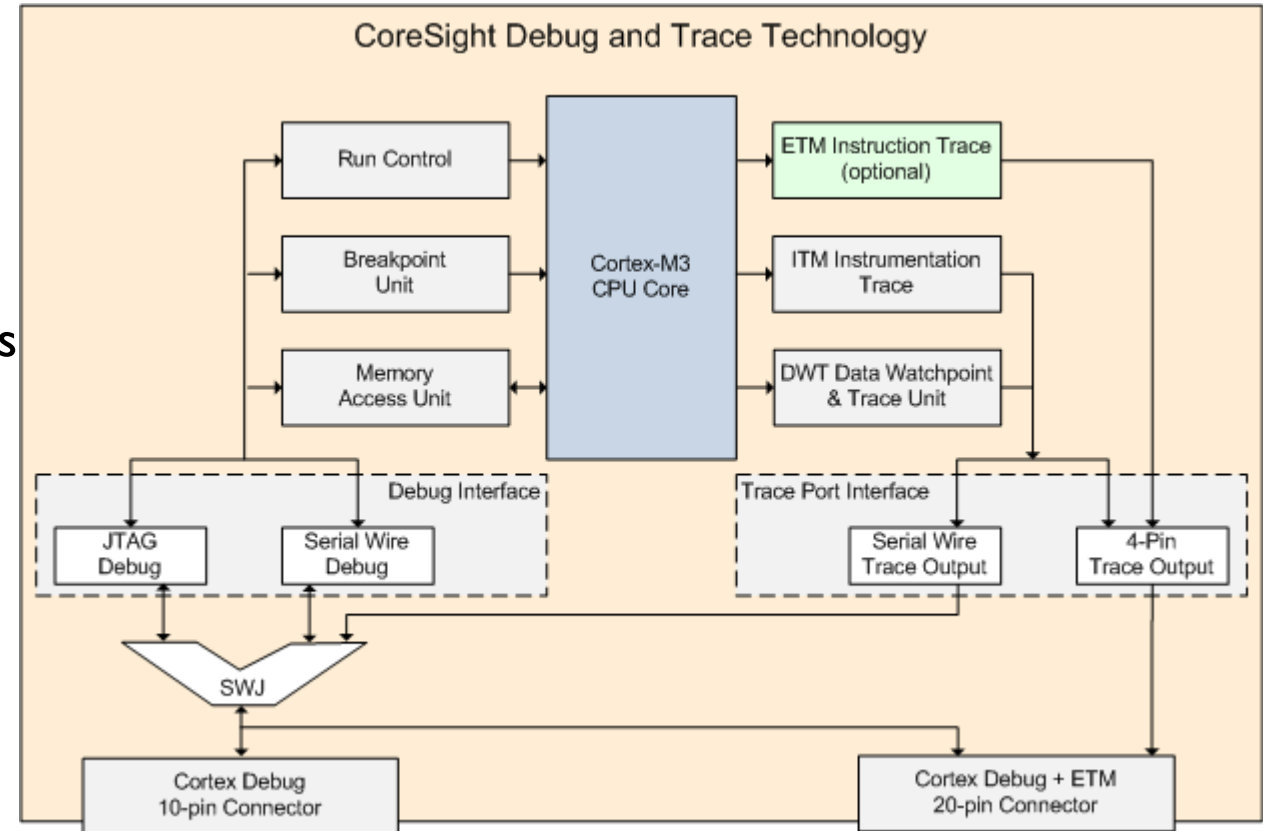
On-chip Debug and Trace Technology

■ Debug Interface

- JTAG/Serial Wire Debug (SWD)
- Start/Stop/Single Step Debug
- Set/Reset Breakpoints
- Read/Write Memory and Peripheral Registers

■ Trace Port Interface

- Serial Wire Trace Output (SWO) only in SWD debug mode
- Instrumentation Trace Macrocell (ITM) provides:
 - Printf debugging
 - RTOS information
 - Interrupt execution information

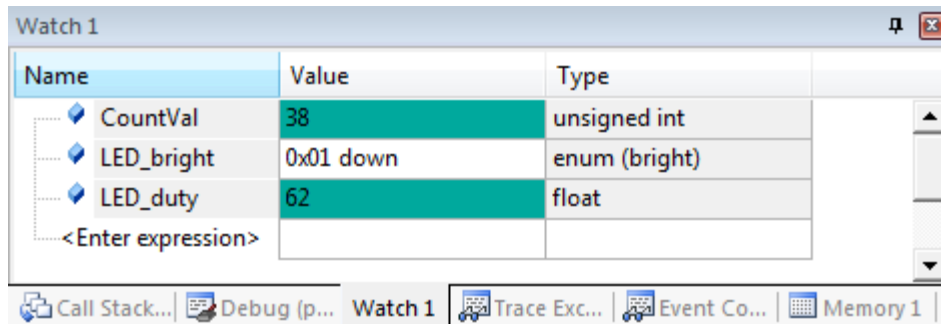


Serial Wire Viewer (SWV) – Real-Time Trace

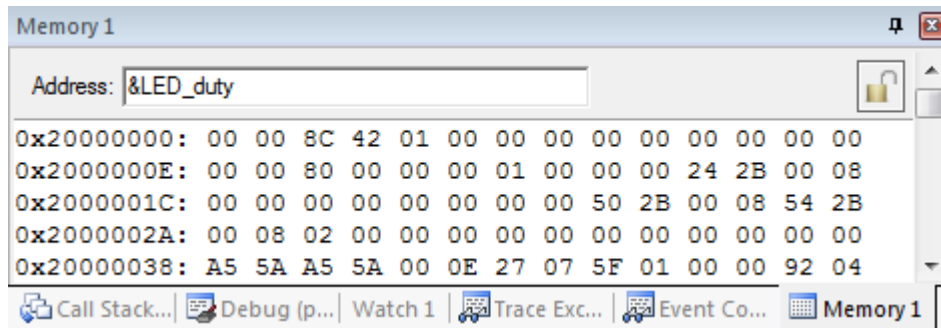
- The Serial Wire Viewer (SWV) provides real-time data trace information from various sources within the Cortex-M3/M4 device. This is output via the single SWO pin while your system processor continues running at full speed.
- Diagnostic system information is available from the ITM and DWT:
 - Data read/write of selected variables
 - PC values
 - Exception and interrupt execution
 - Timing statistics
 - Periodic samples of the program counter
- Event counters display CPU statistics
- Indicates required device wait states or the idle time

SWV: Three Ways to View Variables

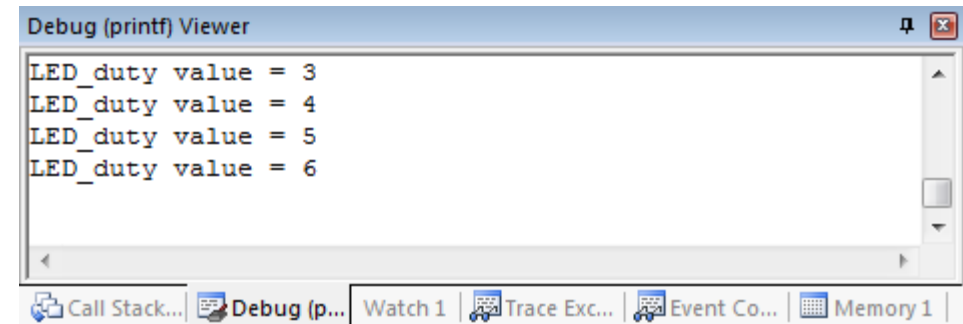
1. In the **Watch** window:



2. In the **Memory** window:



3. In the **Debug (printf) Viewer** window:



SWV Trace Windows

- Event Counters
 - Real-time values of event counters
- Trace Exceptions
 - Statistical information about program exceptions
- Trace Data (with ETM Trace only)
 - Time stamps, PC sample, r/w accesses
 - Updated while target system is running

The screenshot displays two windows from the SWV (System Workbench for Vortex) interface. The top window, titled 'Event Counters', shows a table of event counters with columns for Name, Value, and Enable. The bottom window, titled 'Trace Exceptions', shows a table of exception statistics with columns for Num, Name, Count, Total Time, Min Time In, Max Time In, First Time [s], and Last Time [s].

Event Counters

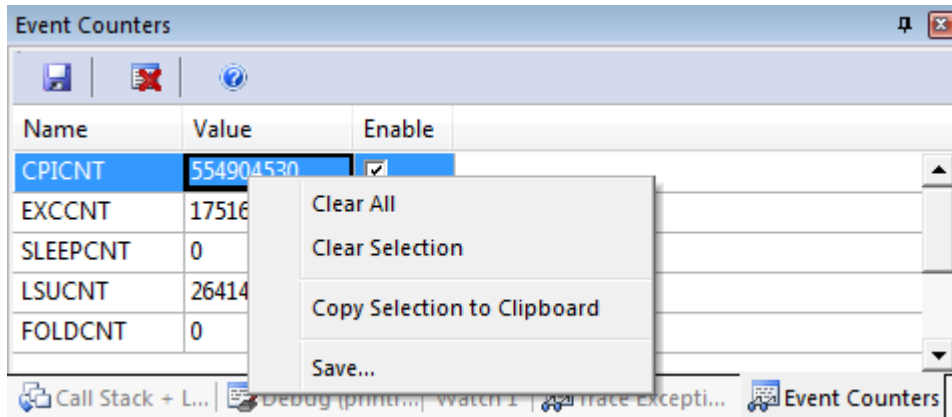
Name	Value	Enable
CPICNT	415332359	<input checked="" type="checkbox"/>
EXCCNT	64820	<input checked="" type="checkbox"/>
SLEEPNT	0	<input checked="" type="checkbox"/>
LSUCNT	98019800	<input checked="" type="checkbox"/>
FOLDNT	0	<input checked="" type="checkbox"/>

Trace Exceptions

Num	Name	Count	Total Time	Min Time In	Max Time In	First Time [s]	Last Time [s]
58	ExtIRQ 42	0	0 s				
59	ExtIRQ 43	0	0 s				
60	ExtIRQ 44	0	0 s				
61	ExtIRQ 45	0	0 s				
62	ExtIRQ 46	28	0 s			295.15431463	307.66498463
63	ExtIRQ 47	0	0 s				
64	ExtIRQ 48	0	0 s				
65	ExtIRQ 49	0	0 s				
66	ExtIRQ 50	0	0 s				
67	ExtIRQ 51	0	0 s				
68	ExtIRQ 52	0	0 s				
69	ExtIRQ 53	0	0 s				
70	ExtIRQ 54	0	0 s				

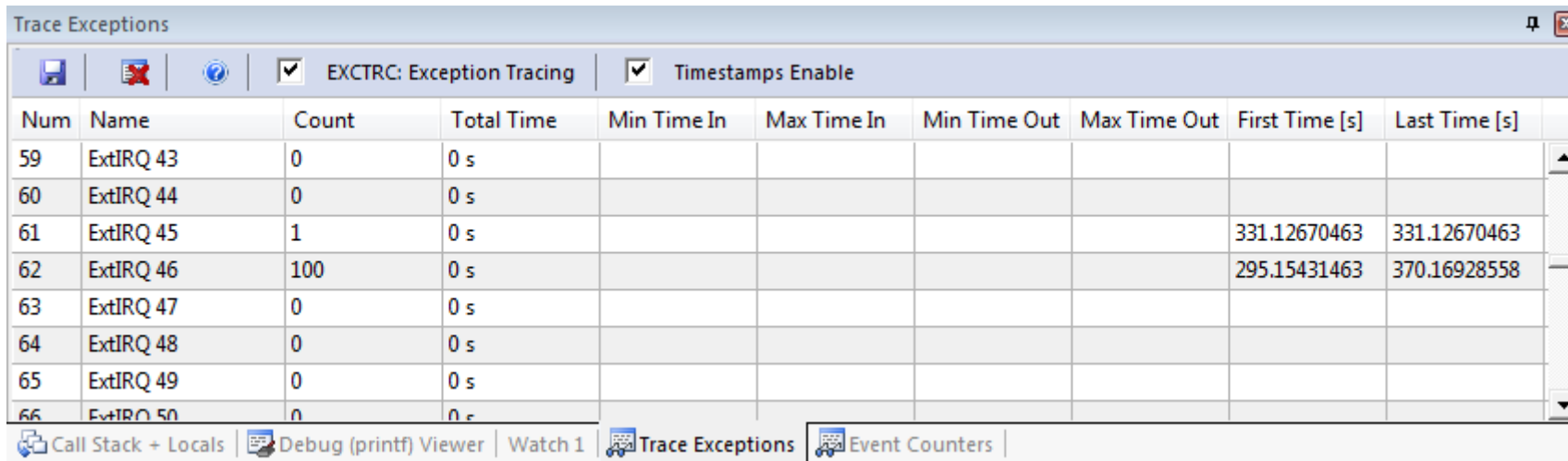
Event Counters

- Displays real-time values of the specific event counters
- Shows the number of times counter 'rolls over'.
- Updated in real time
- Ability to Save Counter Data
- Right Click and select "Clear All" or "Clear Selection"



Trace Exceptions

- Displays statistical information about program exceptions
- Exception name and number, number of times entered
- Max and min time spent in and out of exceptions
- First and last time entered

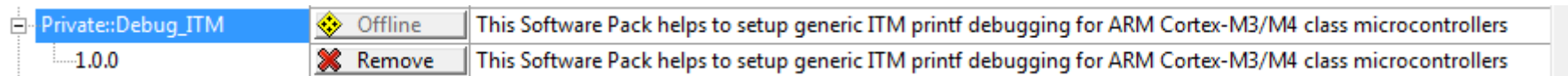


Num	Name	Count	Total Time	Min Time In	Max Time In	Min Time Out	Max Time Out	First Time [s]	Last Time [s]
59	ExtIRQ 43	0	0 s						
60	ExtIRQ 44	0	0 s						
61	ExtIRQ 45	1	0 s					331.12670463	331.12670463
62	ExtIRQ 46	100	0 s					295.15431463	370.16928558
63	ExtIRQ 47	0	0 s						
64	ExtIRQ 48	0	0 s						
65	ExtIRQ 49	0	0 s						
66	ExtIRQ 50	0	0 s						

ITM printf Debugging

Install Private.Debug_ITM.1.0.0.pack

- Double-click on **Private.Debug_ITM.1.0.0.pack** in the **Packs** directory of your USB stick
- The Pack will automatically be installed:



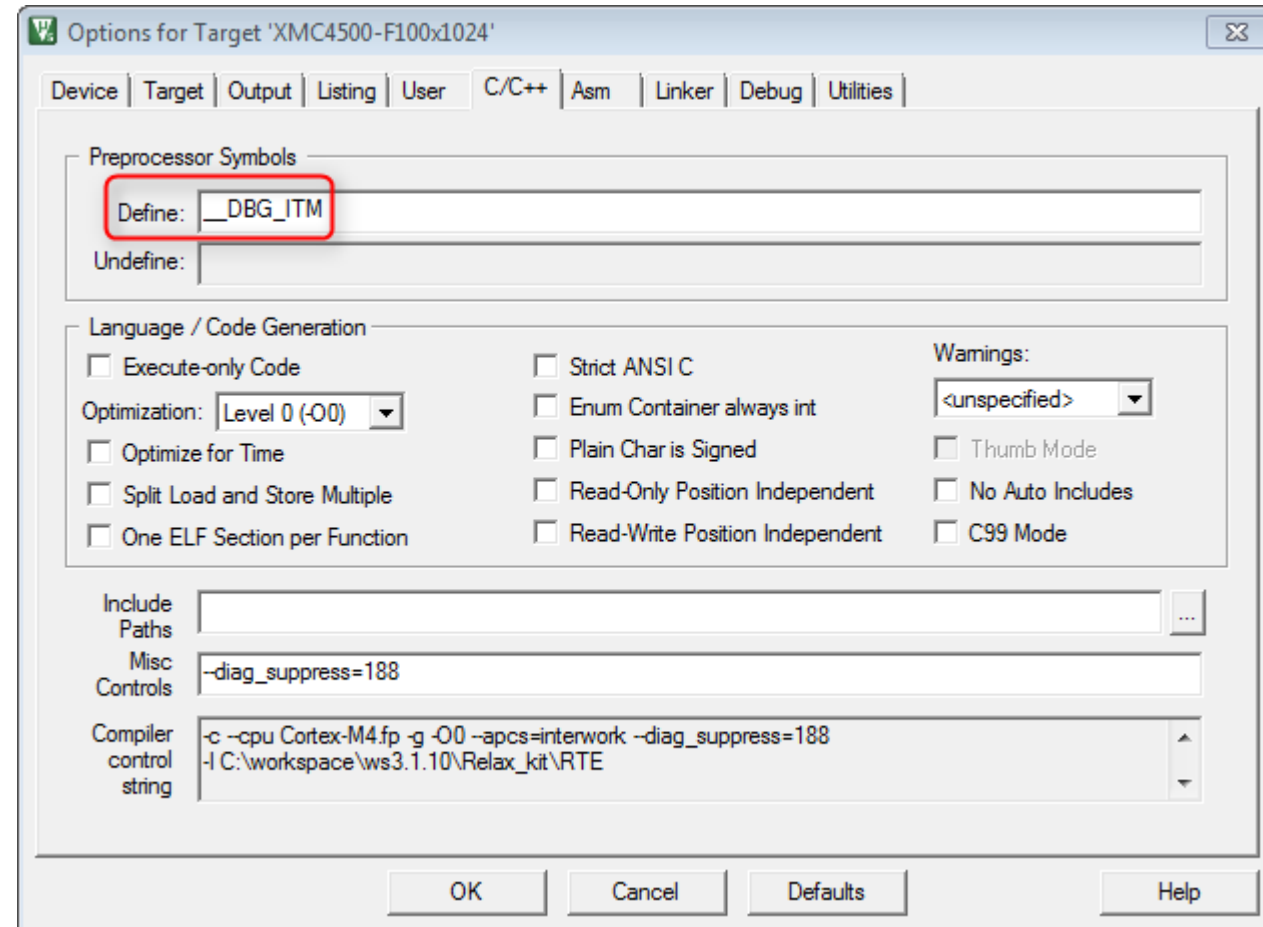
- Open the Manage Run-Time Environment window in μ Vision and browse to the Debug software component:

Software Component	Sel.	Variant	Version	Description
CMSIS	<input type="checkbox"/>			Cortex Microcontroller Software Interface Components
DAVE3	<input type="checkbox"/>			Configuration Files generated by DAVE3
Debug	<input type="checkbox"/>			
ITM	<input checked="" type="checkbox"/>		1.0.0	Support for printf debugging using the ITM in ARM Cortex-M3/M4 class microcontrollers
Device	<input type="checkbox"/>			Setup, System Setup
Drivers	<input type="checkbox"/>			Unified Device Drivers
File System	<input type="checkbox"/>	MDK-Pro	6.0.0	File Access on various storage devices
Graphics	<input type="checkbox"/>	MDK-Pro	5.24.0	User Interface on graphical LCD displays

Click on the link to open the help file

Setting the __DBG_ITM Define

- If not done, stop (⊗) debugging and leave the Debug view (🔍).
- Go to **Project → Options for Target 'XMC4500-F100x1024'** (or press ALT+F7) and click on the **C/C++** tab.
- Enter `__DBG_ITM` in the Preprocessor Symbols → Define box

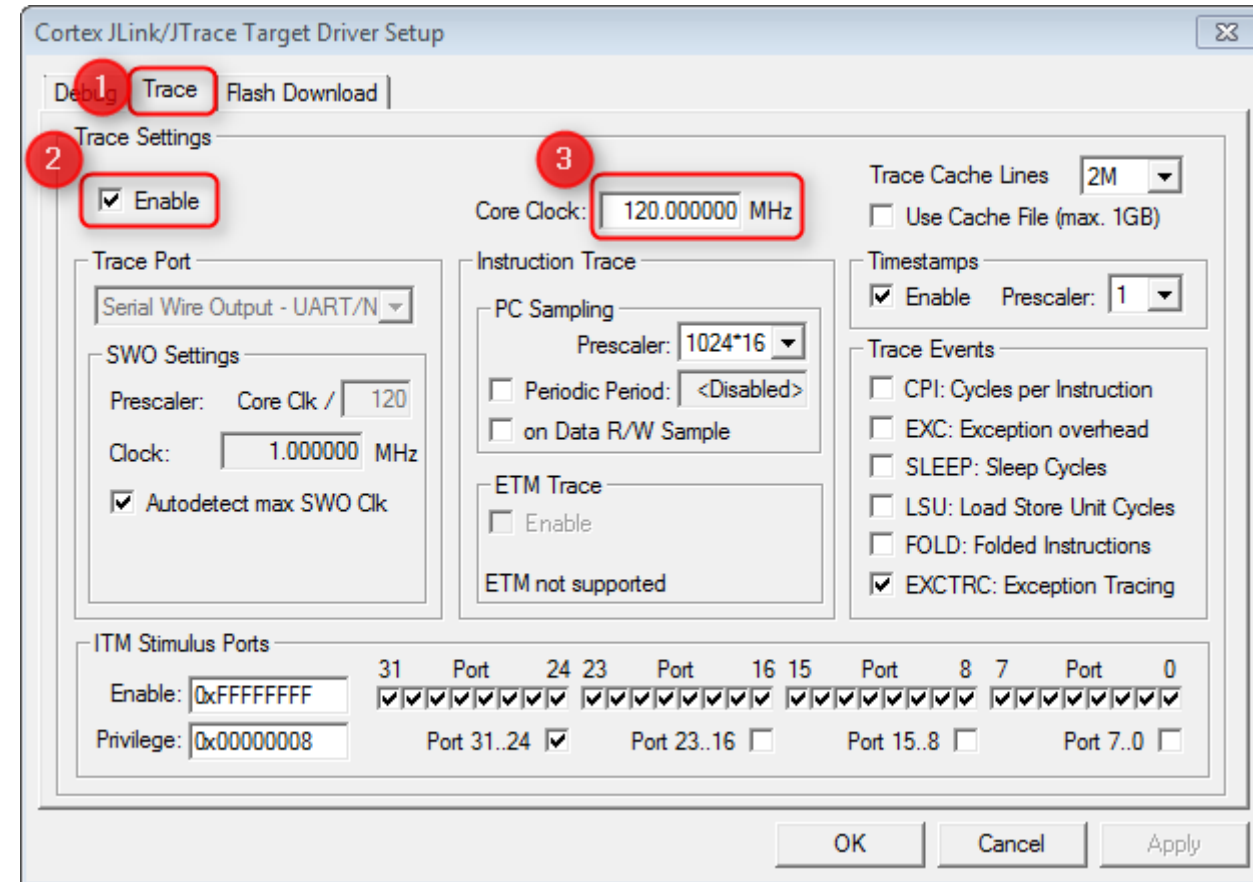


Enabling Trace in J-LINK

http://www.keil.com/support/man/docs/jlink/jlink_cortextrace.htm

- Click on the **Debug** tab.
- Click on **Settings**. A new window opens (Cortex JLink/JTrace Target Driver Setup)

1. Click on the **Trace** tab
2. Check **Enable**
3. Enter the correct Core Clock: **120 MHz**
4. Click **OK** twice



Add Code to Main.c

- At the top of the file add

```
#include <stdio.h>
```


- In the main function's while loop, after the

```
CNT001_GetEvtCountValue (&CNT001_Handle0, &CountVal);
```

- Add

```
printf("Count value = %d\n\r", CountVal);
```

Rebuild the Project and Download to Target

- Go to **Project → Build** (or press F7) to start the build process
- Go to **Flash → Download** (or click on ) to download the program into the target's Flash memory
- Go to **Debug → Start/Stop Debug Session** (or press CTRL+F5) to switch to the μ Vision debugger.
- Go to **View → Serial Windows → Debug (printf) Viewer** to open that window that shows the printf output
- Run the program (press F5)

Debug (printf) Viewer

The screenshot displays the uVision IDE interface for a project named "Relax_kit". The "Registers" window on the left shows the state of the Cortex-M4 registers, with R15 (PC) at 0x08001C98. The "Disassembly" window shows the current instruction at address 0x08001C96: "LDR R0, =_Vectors". The "Text Editor" window shows the source code for "startup_XMC4500.s", including the "Reset_Handler" function. The "Debug (printf) Viewer" window at the bottom right shows a list of printf output messages, all stating "Count value = 26". The "Command" window at the bottom left shows target information for the XMC4500-F100x1024 device. The status bar at the bottom indicates the trace is "SW Buffer Overrun" and the time is 7.69997717 sec.

Registers

Register	Value
R0	0x00000000
R1	0x00000000
R2	0x00000000
R3	0x00000000
R4	0x00000000
R5	0x00000000
R6	0x00000000
R7	0x00000000
R8	0x00000000
R9	0x00000000
R10	0x00000000
R11	0x00000000
R12	0x00000000
R13 (SP)	0x00000000
R14 (LR)	0x00000000
R15 (PC)	0x08001C98

Disassembly

```
0x08001C96 5AA5 DCW 0x5AA5
292: LDR R0, =_Vectors
0x08001C98 480A LDR r0, [pc, #40] ; @0x08001CC4
293: LDR R1, =0xE000ED08 ; *VTOR register
0x08001C9A 490B LDR r1, [pc, #44] ; @0x08001CC8
294: STR R0, [R1]
```

Text Editor

```
285 ;* Reset_Handler */
286 Reset_Handler PROC
287 EXPORT Reset_Handler [WEAK]
288 IMPORT SystemInit
289 IMPORT __main
290
291 ; Remap vector table
292 LDR R0, =_Vectors
```

Debug (printf) Viewer

```
Count value = 26
Count value = 26
Count value = 26
Count value = 26
Count value = 26
Count value = 26
Count value = 26
Count value = 26
Count value = 26
Count value = 26
```

Command

```
Target info:
-----
Device: XMC4500-F100x1024
VTarget = 3.300V
State of Pins: TCK: 0, TDI: 0, TDO: 0, TMS: 16, TRST: 1
Hardware-Breakpoints: 6
Software-Breakpoints: 8192
Watchpoints: 4
```

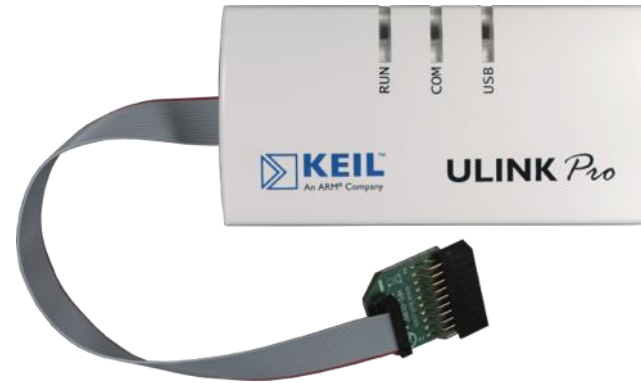
ASSIGN BreakDisable BreakEnable BreakKill BreakList BreakSet BreakAccess

Trace: SW Buffer Overrun J-LINK / J-TRACE Cortex t1: 7.69997717 sec L:292 C:1 CAP NUM SCRL OVR R/W

ULINKpro

Debug and Trace Adapter

- Programming + Run Control
- Memory + Breakpoint Access
- Serial Wire Trace (SWO)
 - 100 Mbps (Manchester Mode)
 - ITM and Data Trace @ CPU speed
- ETM Streaming Trace
 - Up to 800 Mbps
 - 100% Code Coverage and Performance Analysis



JTAG

SWD

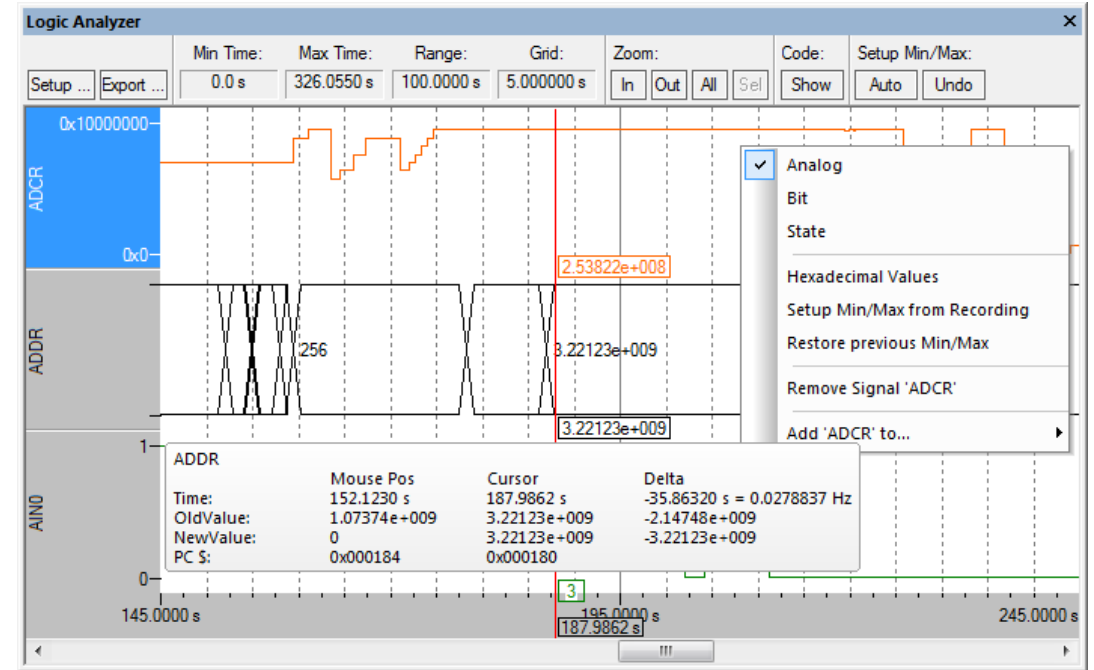
SWO
100 Mbps

ETM
Streaming

ULINKpro

Fastest Data Trace

- 100 x faster than most other MCU solutions
 - Real-Time data trace analysis
 - CPU operates at full speed
 - No overflows or lost data
- MDK gives clear visibility into application behaviour



JTAG

SWD

SWO
100 Mbps

ETM
Streaming

ULINKpro

Streaming Instruction Trace

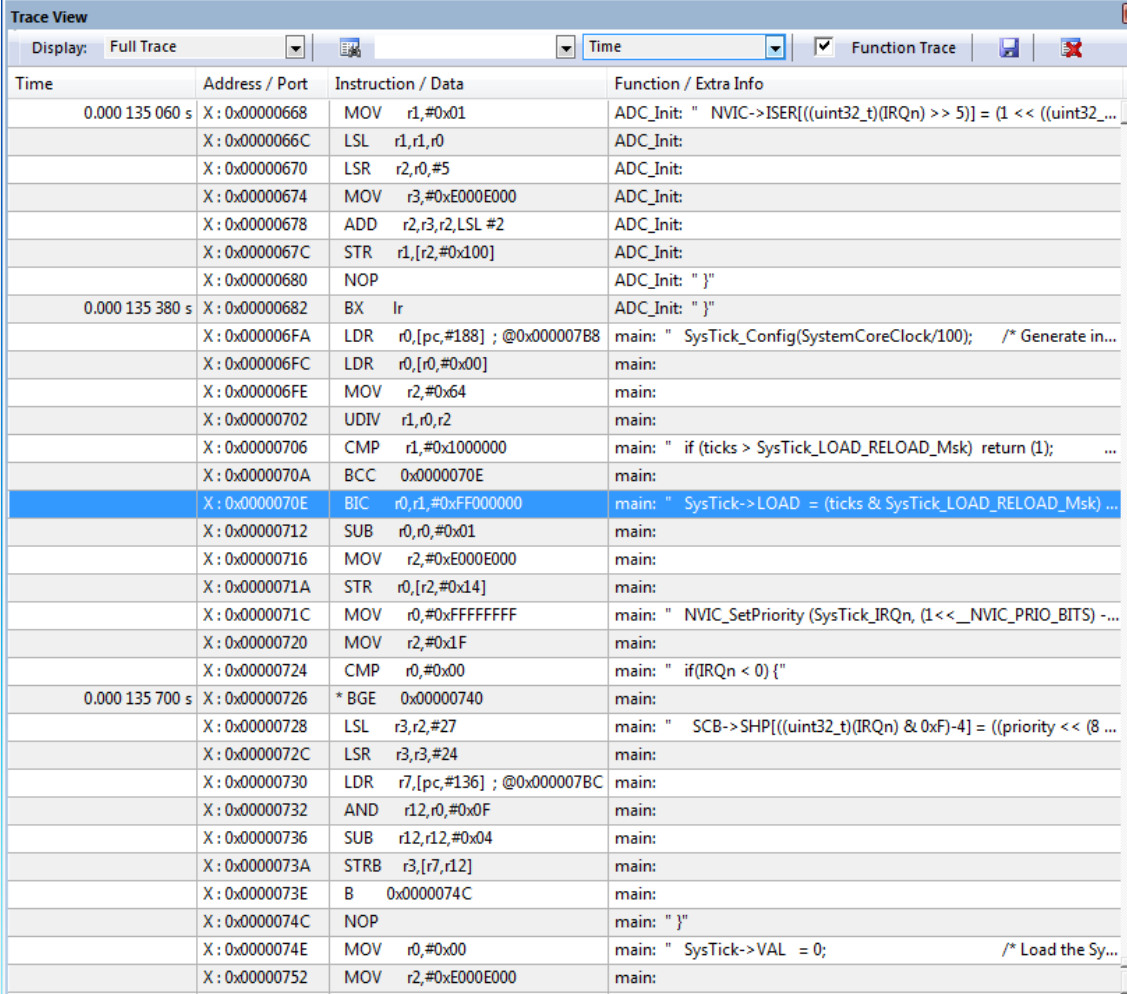
- What is Streaming Trace?
 - Trace data transferred in real-time to debug host
 - Trace for minutes, hours, or longer
 - Required for full code-coverage and timing analysis
- ULINKpro
 - Only solution to stream trace directly to PC
 - Search trace data
 - Save trace data
 - Function trace
 - Synchronized to source code

JTAG

SWD

SWO
100 Mbps

ETM
Streaming



Time	Address / Port	Instruction / Data	Function / Extra Info
0.000 135 060 s	X: 0x00000668	MOV r1, #0x01	ADC_Init: " NVIC->ISER[(((uint32_t)(IRQn) >> 5)] = (1 << (((uint32_t)...
	X: 0x0000066C	LSL r1, r1, #0	ADC_Init:
	X: 0x00000670	LSR r2, r0, #5	ADC_Init:
	X: 0x00000674	MOV r3, #0xE000E000	ADC_Init:
	X: 0x00000678	ADD r2, r3, LSL #2	ADC_Init:
	X: 0x0000067C	STR r1, [r2, #0x100]	ADC_Init:
	X: 0x00000680	NOP	ADC_Init: " }"
0.000 135 380 s	X: 0x00000682	BX lr	ADC_Init: " }"
	X: 0x000006FA	LDR r0, [pc, #188] ; @0x000007B8	main: " SysTick_Config(SystemCoreClock/100); /* Generate in...
	X: 0x000006FC	LDR r0, [r0, #0x00]	main:
	X: 0x000006FE	MOV r2, #0x64	main:
	X: 0x00000702	UDIV r1, r0, r2	main:
	X: 0x00000706	CMP r1, #0x1000000	main: " if (ticks > SysTick_LOAD_RELOAD_Msk) return (1); ...
	X: 0x0000070A	BCC 0x0000070E	main:
	X: 0x0000070E	BIC r0, r1, #0xFF000000	main: " SysTick->LOAD = (ticks & SysTick_LOAD_RELOAD_Msk) ...
	X: 0x00000712	SUB r0, r0, #0x01	main:
	X: 0x00000716	MOV r2, #0xE000E000	main:
	X: 0x0000071A	STR r0, [r2, #0x14]	main:
	X: 0x0000071C	MOV r0, #0xFFFFFFFF	main: " NVIC_SetPriority (SysTick_IRQn, (1 << __NVIC_PRIO_BITS) - ...
	X: 0x00000720	MOV r2, #0x1F	main:
	X: 0x00000724	CMP r0, #0x00	main: " if (IRQn < 0) {"
0.000 135 700 s	X: 0x00000726	* BGE 0x00000740	main:
	X: 0x00000728	LSL r3, r2, #27	main: " SCB->SHP[(((uint32_t)(IRQn) & 0xF)-4)] = ((priority << (8 ...
	X: 0x0000072C	LSR r3, r3, #24	main:
	X: 0x00000730	LDR r7, [pc, #136] ; @0x000007BC	main:
	X: 0x00000732	AND r12, r0, #0x0F	main:
	X: 0x00000736	SUB r12, r12, #0x04	main:
	X: 0x0000073A	STRB r3, [r7, r12]	main:
	X: 0x0000073E	B 0x0000074C	main:
	X: 0x0000074C	NOP	main: " }"
	X: 0x0000074E	MOV r0, #0x00	main: " SysTick->VAL = 0; /* Load the Sy...
	X: 0x00000752	MOV r2, #0xE000E000	main:

Instruction (ETM)

Function Name : "main"
Src Module : Blinky.c
Src Line : 1141

ULINKpro

Advanced Debug Capability

- Streaming Instruction Trace
 - Debug historical sequences
 - Full details of execution history
 - Application Soak testing over long periods of time
- Performance Analysis
 - Optimize and Profile Applications
 - Identify hotspots quickly
- Code Coverage
 - Implement 100% accurate Code Coverage on silicon
 - Essential for validation and verification
- Fastest Data Trace
 - 100 times faster than any other solution
 - CPU at full speed
 - No overflows or lost data

JTAG

SWD

SWO
100 Mbps

ETM
Streaming

Creating Applications using Software Packs

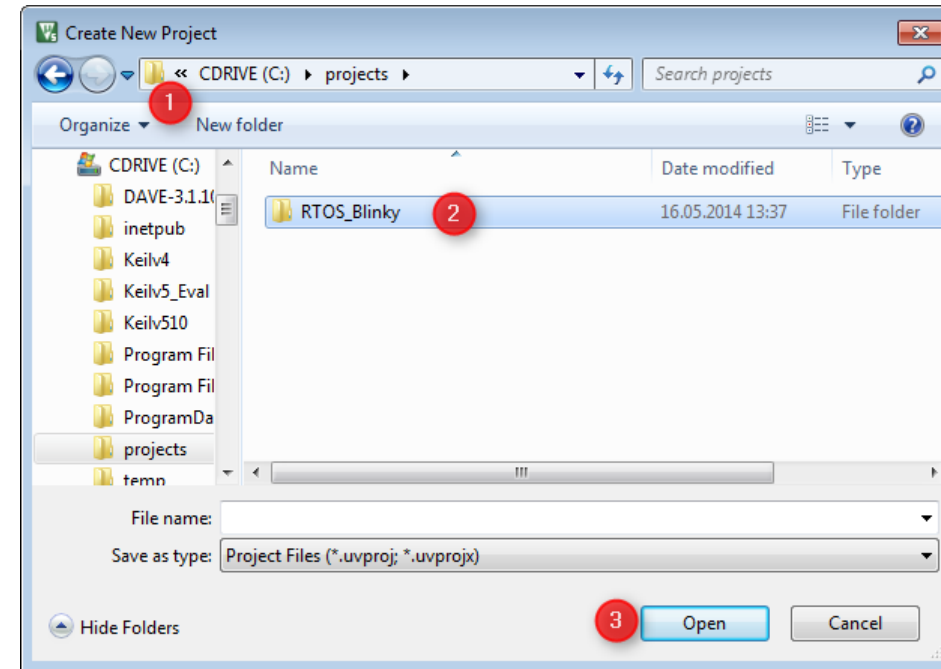
Objective: Blinky Project with RTOS

- We will create a new project from scratch, using:
 - CMSIS-CORE
 - Startup Files for CMSIS
 - CMSIS-RTOS compliant Keil RTX
- Later, we will add a USB driver and Middleware to set and reset the LEDs from a PC

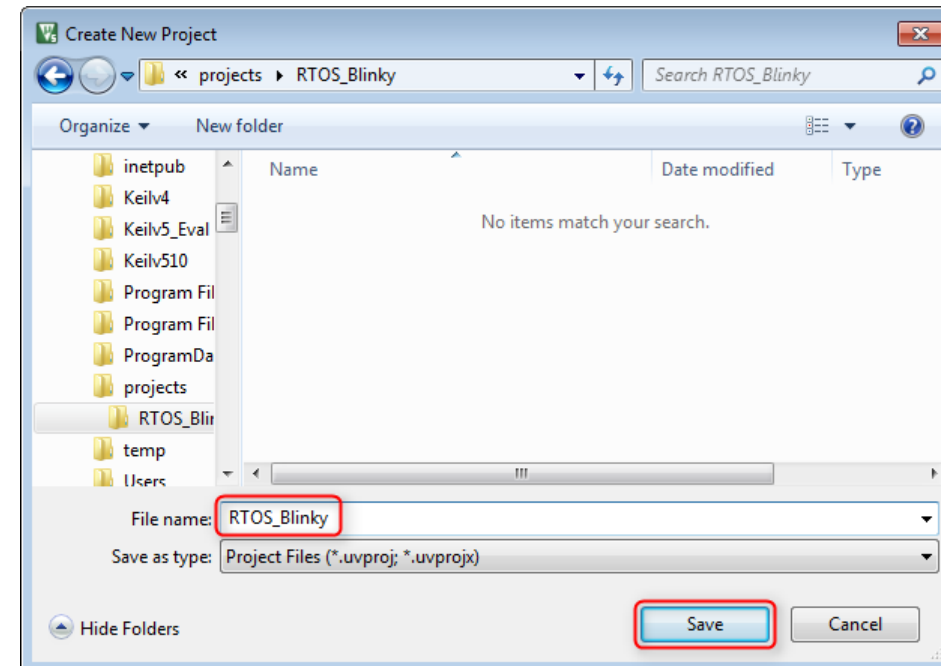
Create a New Project

- Open μ Vision and click **Project → New μ Vision Project...**

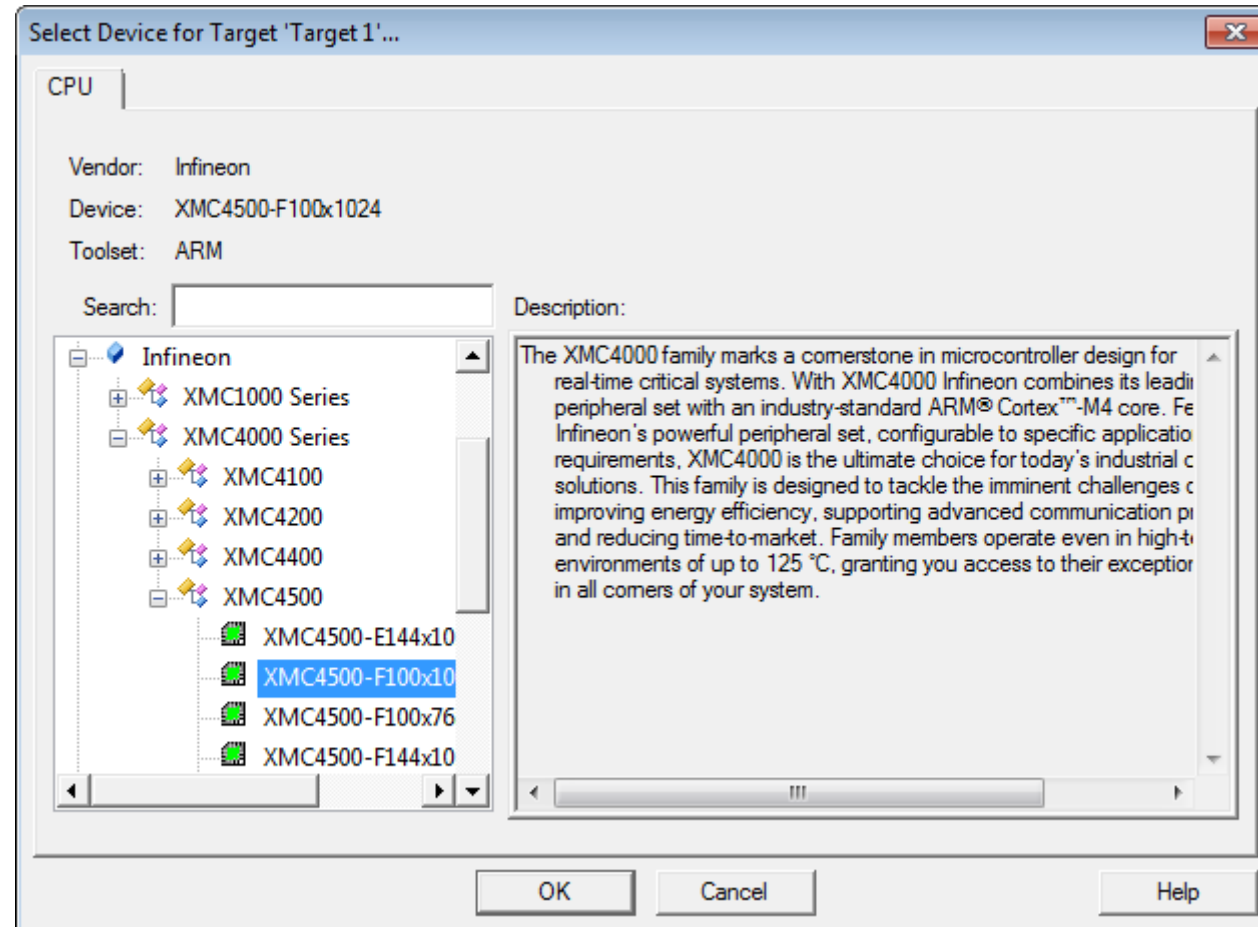
- Click **New folder**
- Enter Name: **RTOS_Blinky**
- Click **Open**



- Enter File Name: **RTOS_Blinky**
- Click **Save**



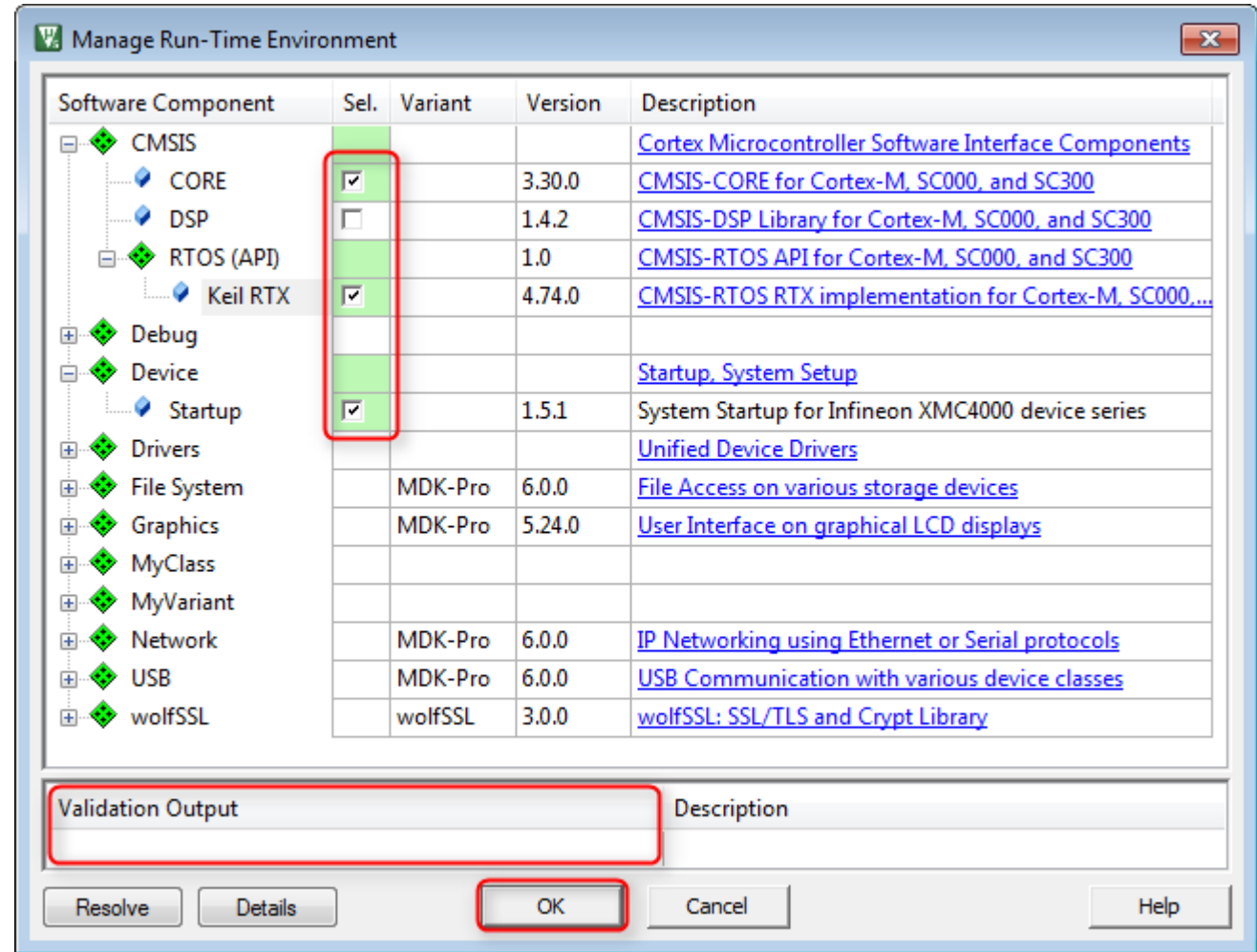
Select Device for Target: XMC4500-F100x1024



Manage Run-Time Environment

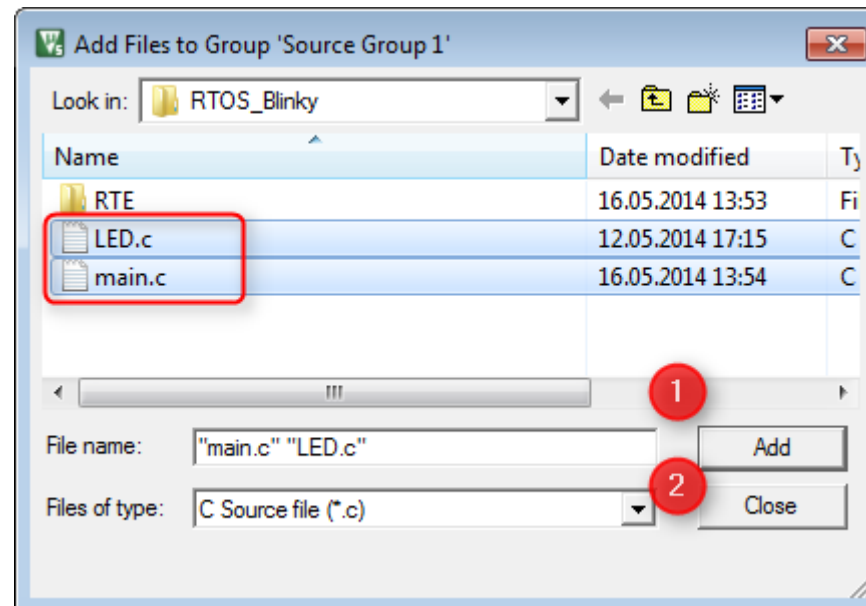
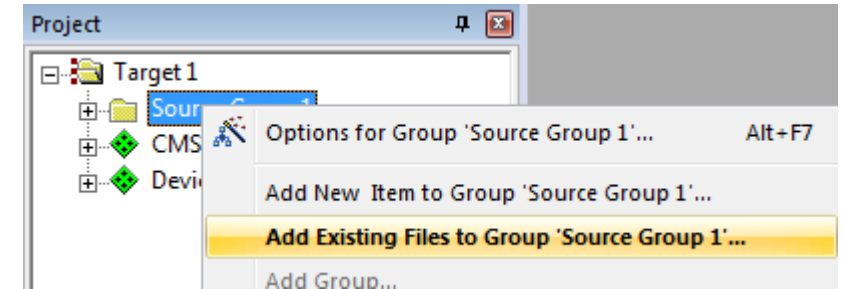
Select Software Components

- Expand **CMSIS**
 - Select **CORE**
- **Validation Output** shows required software components
- In Validation Output click on Infineon::Device:Startup
 - You will be automatically directed to the required component
 - Select the component
- Expand **CMSIS:RTOS (API)**
 - Select **Keil RTX**
- Click **OK**



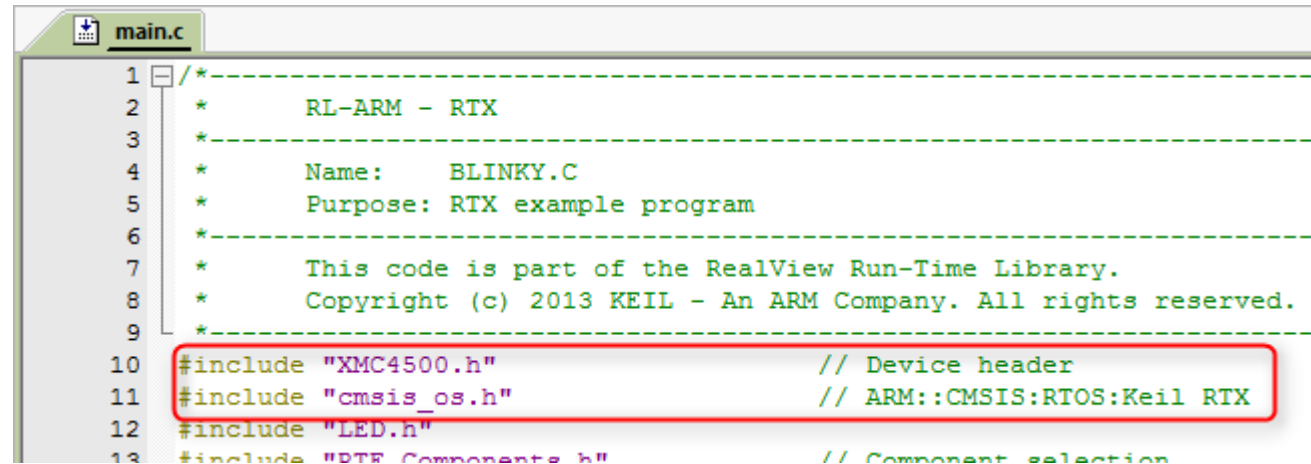
Add Code

- Copy the files **main.c**, **LED.c**, and **LED.h** from the Code directory on the USB stick to your **RTOS_Blinky** project directory
- Expand **Target 1** in the **Project** view
- Right click on **Source Group 1** and select **Add Existing Files to Group 'Source Group 1'...**
- Click on **main.c** and **LED.c**
 1. Click **Add**
 2. Click **Close**



Work with the Code

- Double-click **main.c**. The file opens in the editor
- You'll notice a lot of red x's and warning signs
- Add required include files by right-clicking in the code (around line 10)
 - Click on **Insert '#include file'** and select **XMC4500.h**
 - Repeat for **cmsis_os.h**

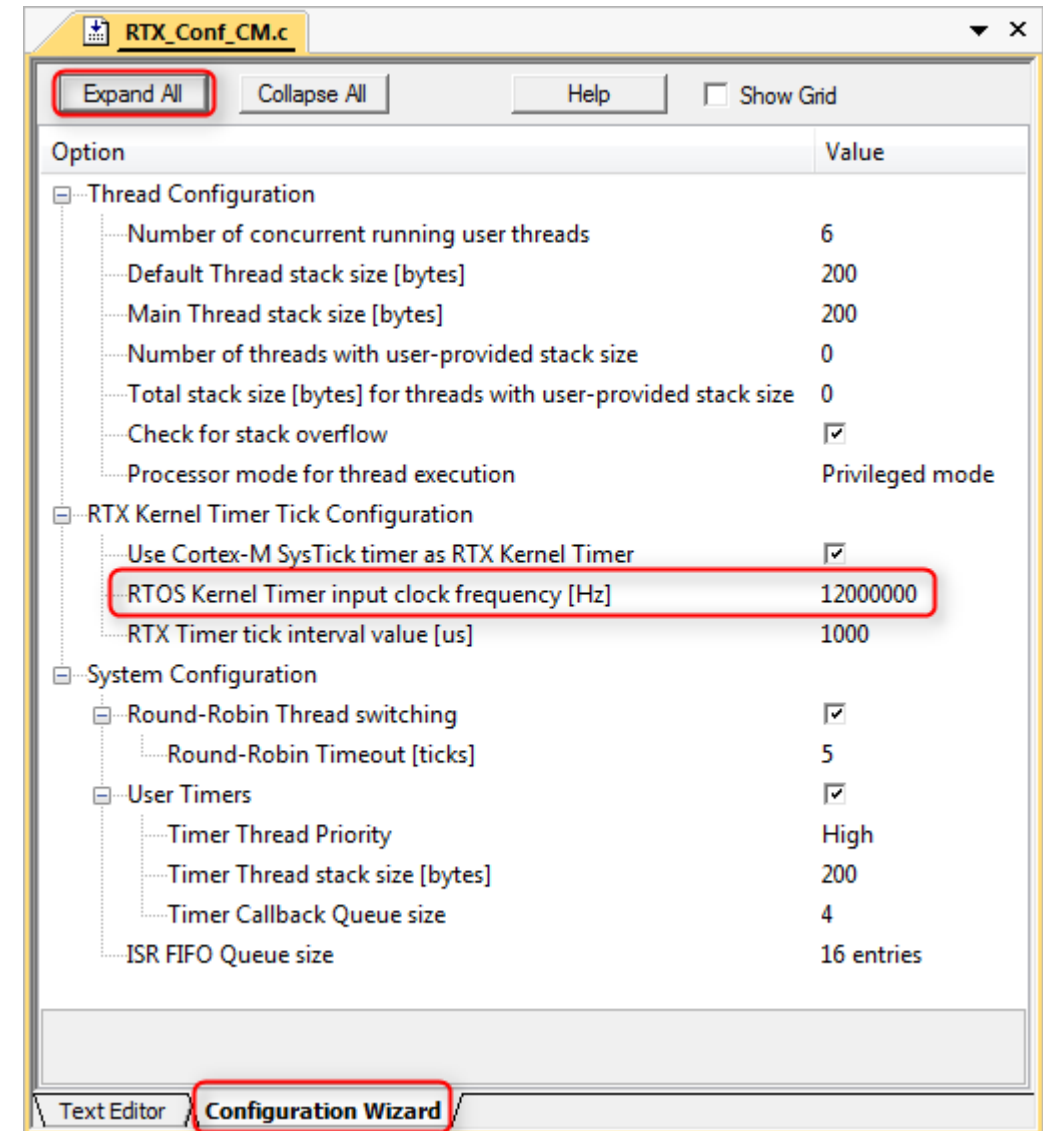


```
1  /*  
2  *      RL-ARM - RTX  
3  *  
4  *      Name:      BLINKY.C  
5  *      Purpose:   RTX example program  
6  *  
7  *      This code is part of the RealView Run-Time Library.  
8  *      Copyright (c) 2013 KEIL - An ARM Company. All rights reserved.  
9  *  
10 #include "XMC4500.h"           // Device header  
11 #include "cmsis_os.h"         // ARM::CMSIS:RTOS:Keil RTX  
12 #include "LED.h"  
13 #include "DTE_Components.h"   // Component selection
```

- Errors and warnings should disappear
- Press **CTRL+S** to save the file

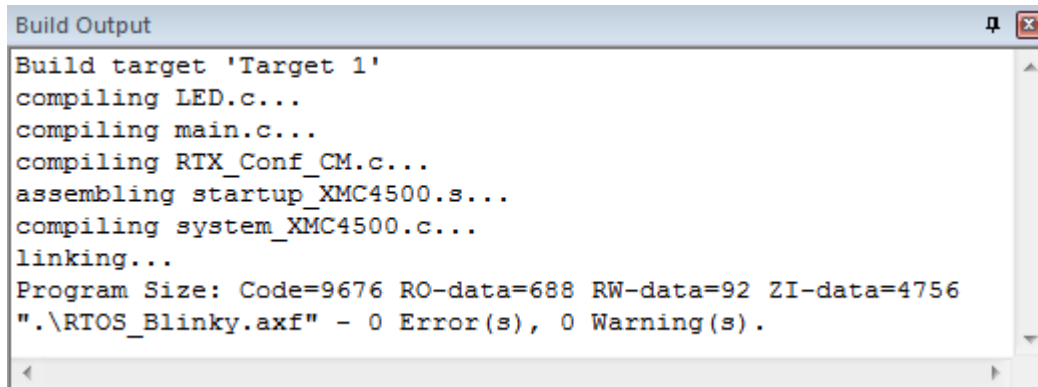
Required RTOS Settings

- Double-click **RTX_Conf_CM.c**. The file opens in the editor.
- Click **Configuration Wizard** tab (at the bottom of the file)
- Graphical representation of the configuration files appears. Click **Expand All**
- All available settings for Keil RTX are displayed
- We need to change the RTOS Kernel Timer input clock frequency [Hz] to 120 MHz as this is the core frequency
- Click on **12000000** and add a **0**
- Press **CTRL+S** to save the file



Build the Project

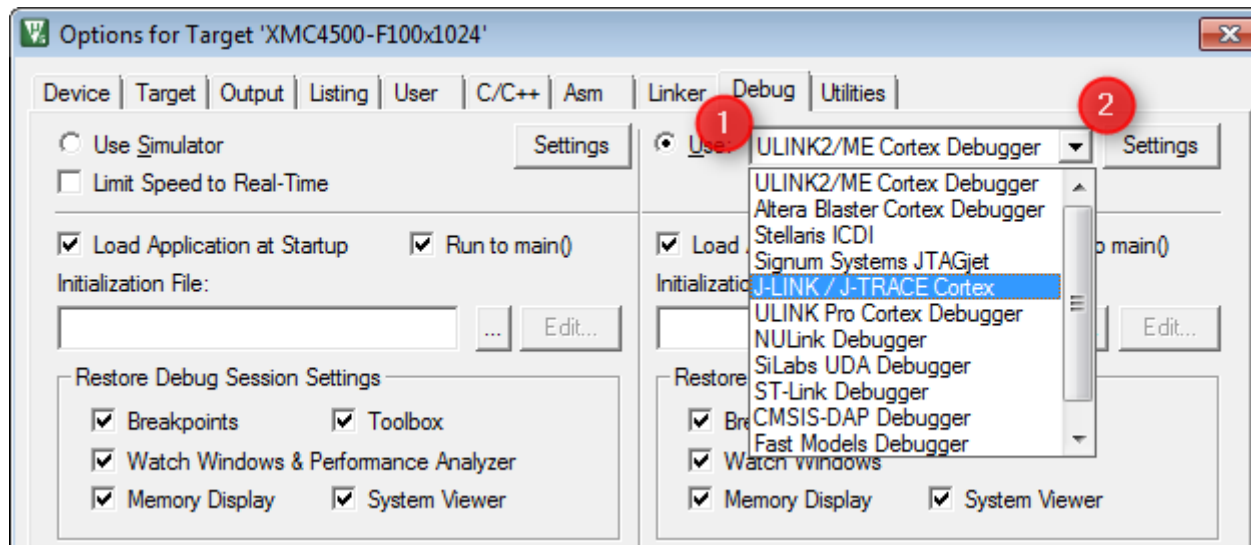
- Press **F7** or click **Project** → **Build target** to start the build process
- The project should build without an error or a warning



```
Build Output
Build target 'Target 1'
compiling LED.c...
compiling main.c...
compiling RTX_Conf_CM.c...
assembling startup_XMC4500.s...
compiling system_XMC4500.c...
linking...
Program Size: Code=9676 RO-data=688 RW-data=92 ZI-data=4756
".\RTOS_Blinky.axf" - 0 Error(s), 0 Warning(s).
```

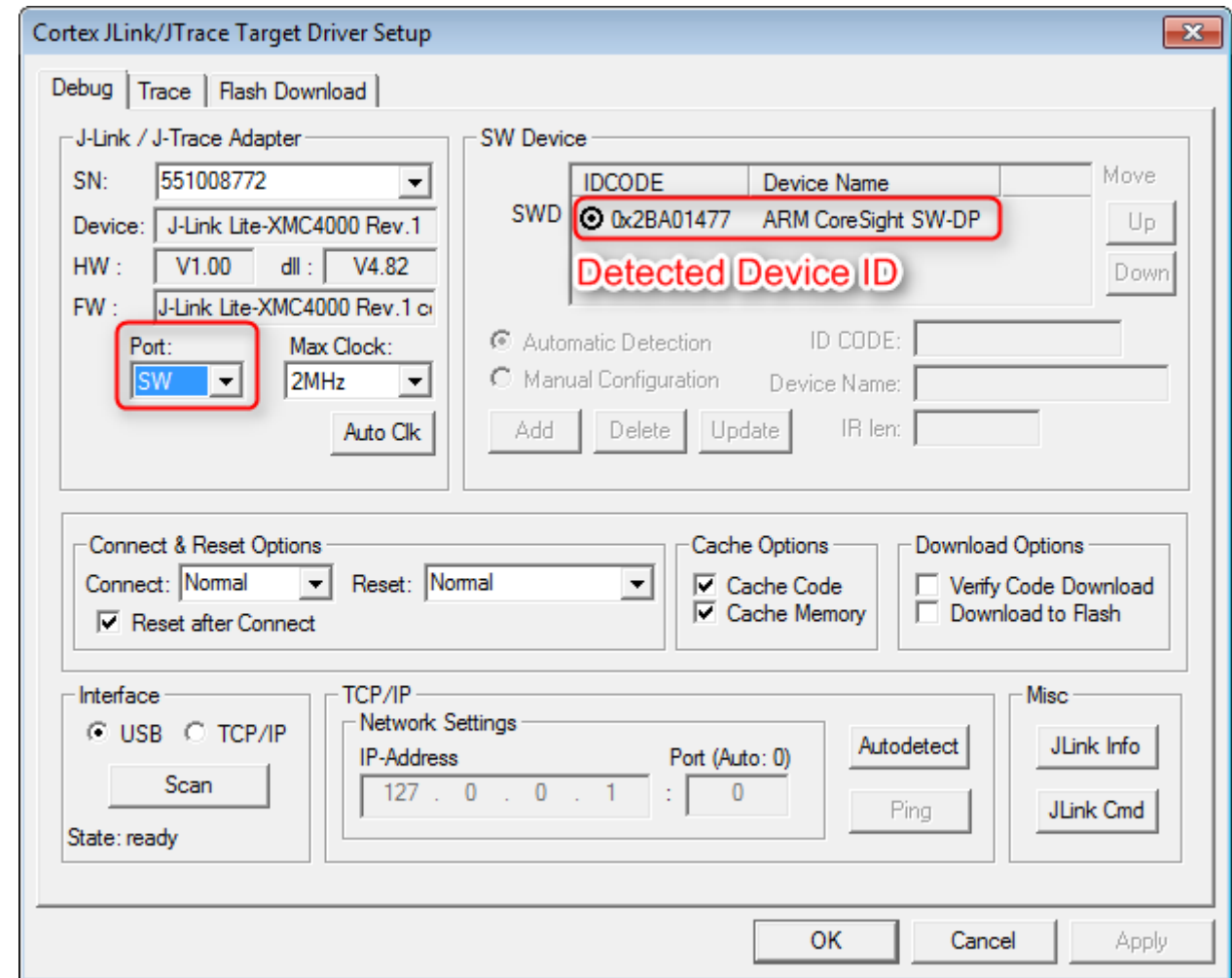
Select the J-LINK Debugger

- The XMC4500 Relax Kit has an integrated J-LINK debug adapter
 1. To change default adapter, click on **ULINK2/ME Cortex Debugger** and scroll down until **J-LINK/J-TRACE Cortex**
 2. Click on **Settings**, to check the connectivity between the target and your PC



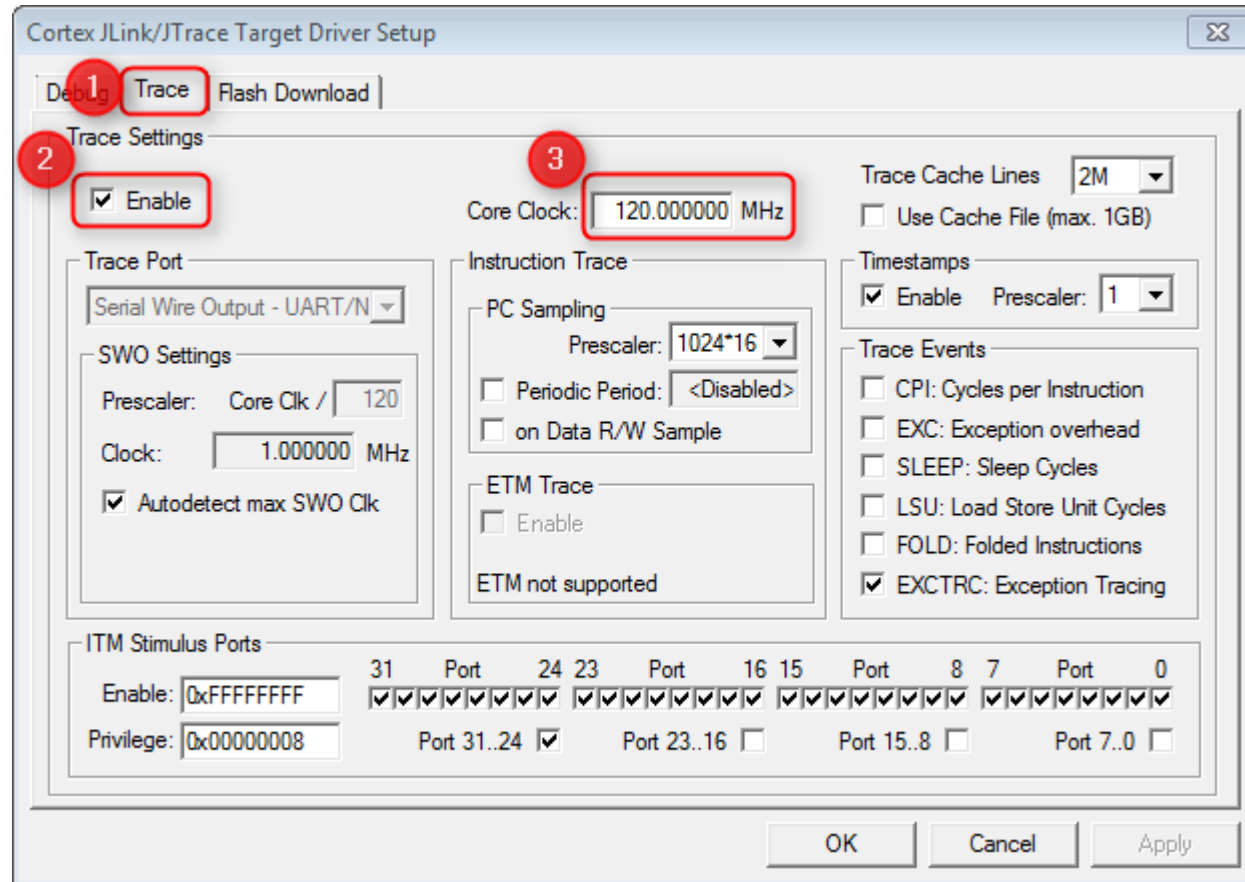
Enable the SW Port

- μ Vision will try to connect to the J-LINK using a JTAG port. This is not available on the Relax Kit
- Click on **Port: JTAG** and set to **SW**
- The connected device will be identified automatically




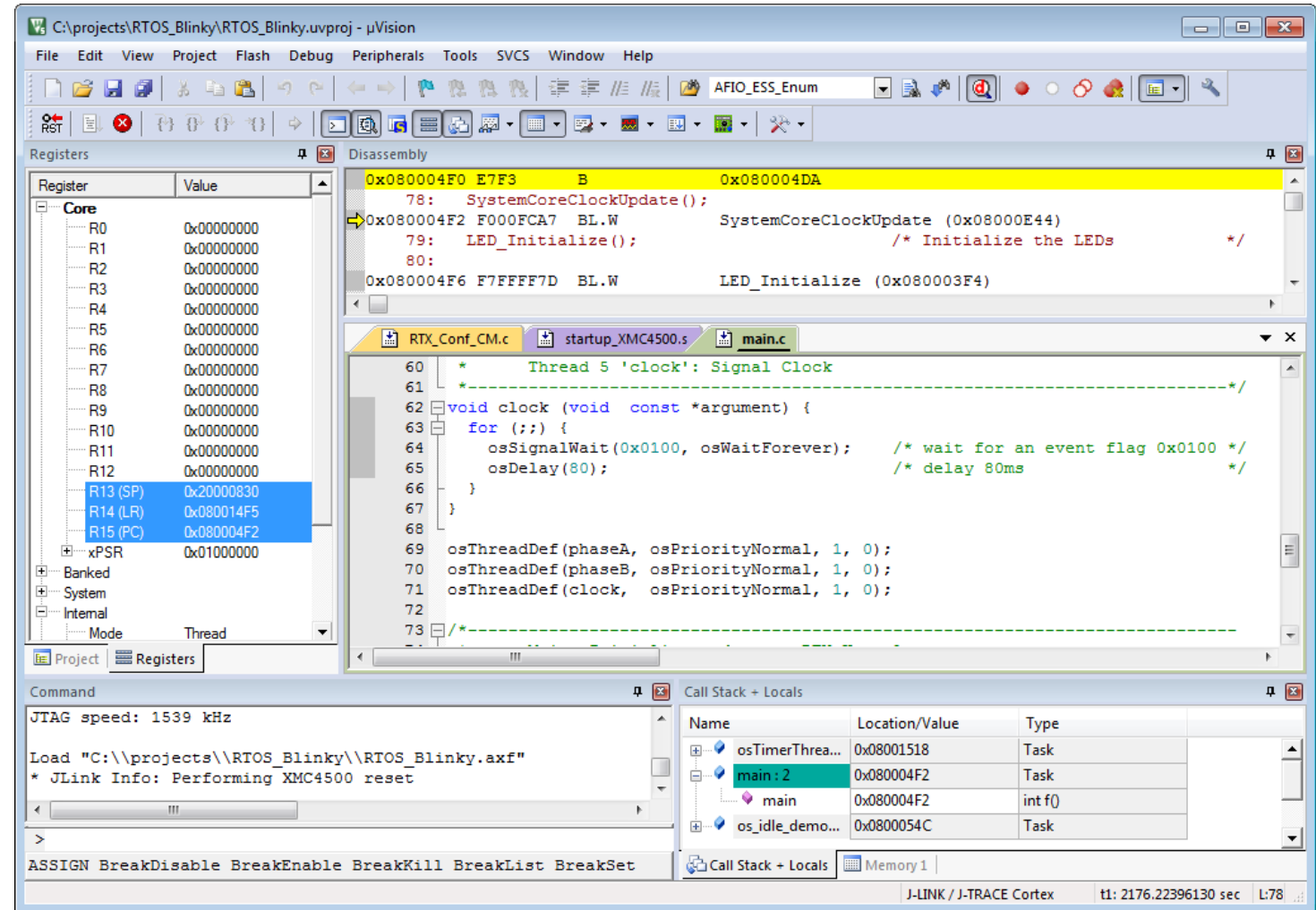
Enabling Trace in J-LINK

1. Click on the **Trace** tab
2. Check **Enable**
3. Enter the correct Core Clock: **120 MHz**
4. Click **OK** twice




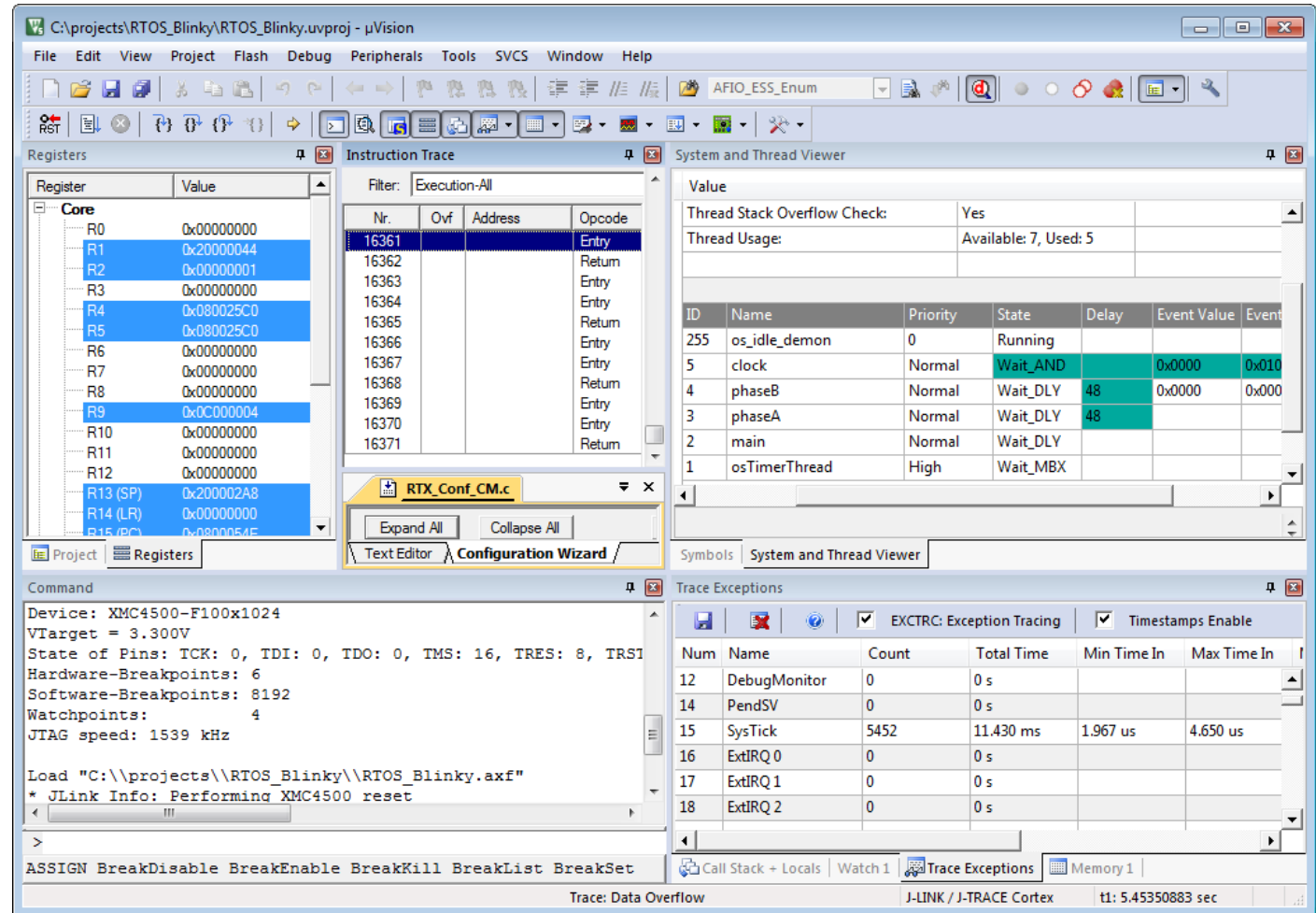
Start a Debug Session

- Go to **Debug** → **Start/Stop Debug Session** (or press CTRL+F5) to switch to the µVision debugger.
- During the start of the debug session, µVision loads the application, executes startup code, and stops at the main C function
- Click **Run**  on the toolbar. LED1 and LED2 will start flashing alternately



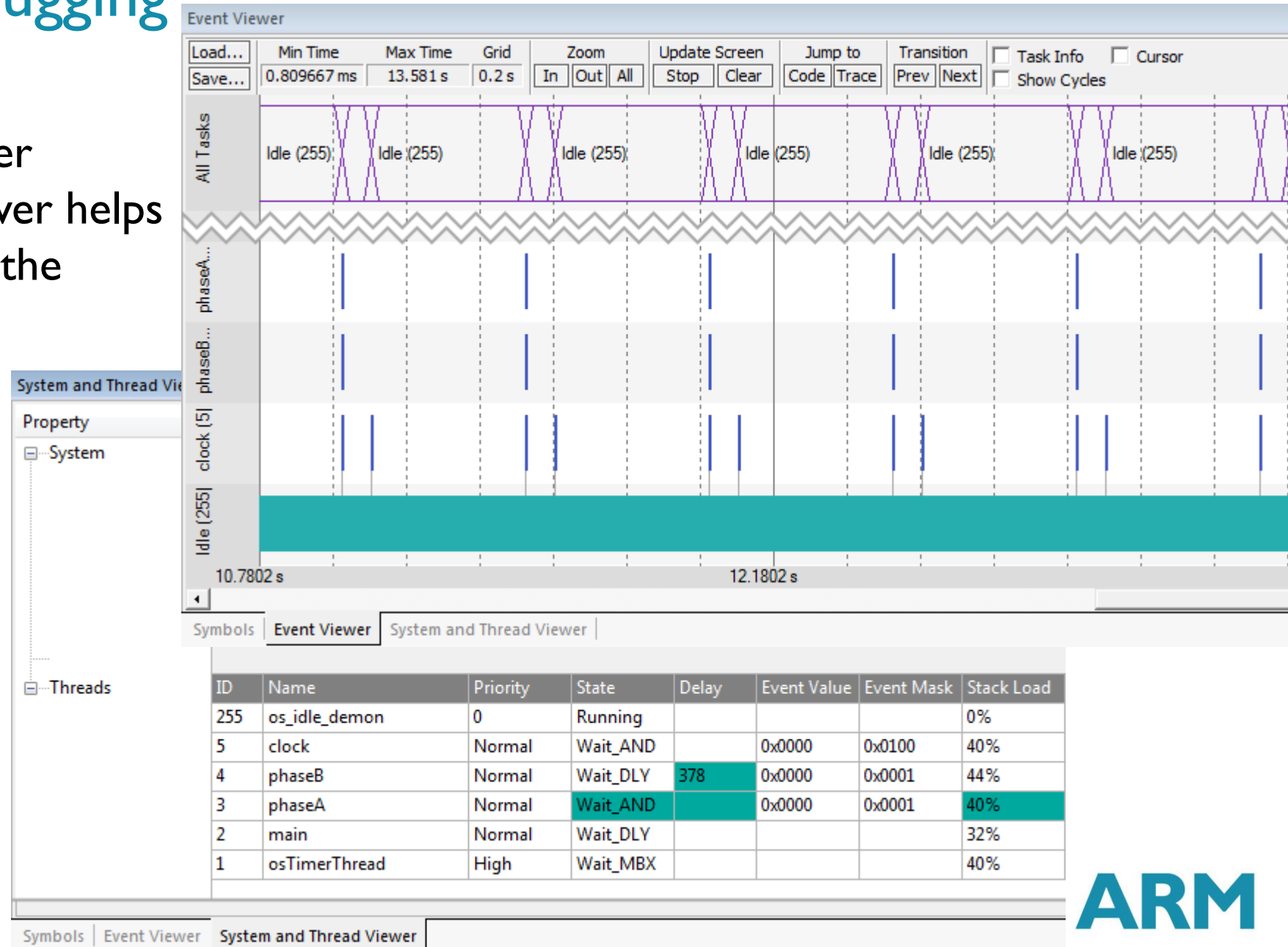
Add more Debug Information

- End the session by clicking **Stop** 
- Go to **Debug** → **OS Support** → **System and Thread Viewer**
- Go to **Debug** → **OS Support** → **Event Viewer**
- Go to **View** → **Trace** → **Trace Exceptions**
- Go to **Peripherals** → **System Viewer** → **PORTS** → **PORT I**



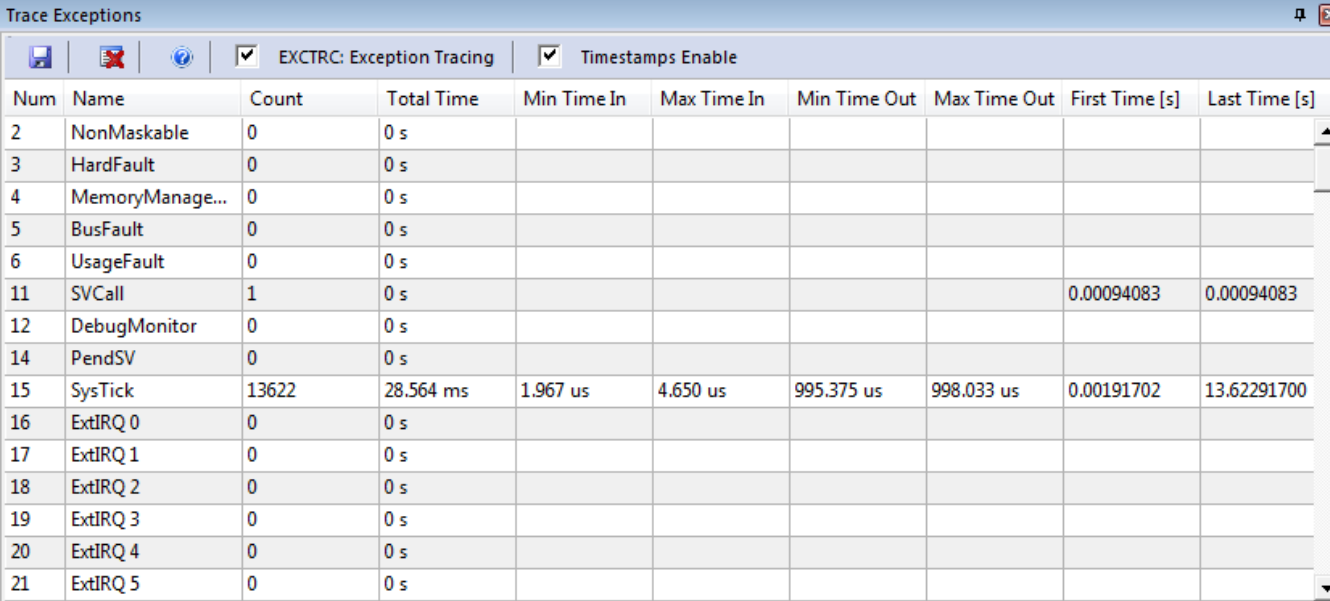
Kernel-Aware Debugging

- System and Thread Viewer together with Event Viewer helps to track the progress of the program



Trace Exceptions

- Displays statistical information about program exceptions
- Exception name and number, number of times entered
- Max and min time spent in and out of exceptions
- First and last time entered

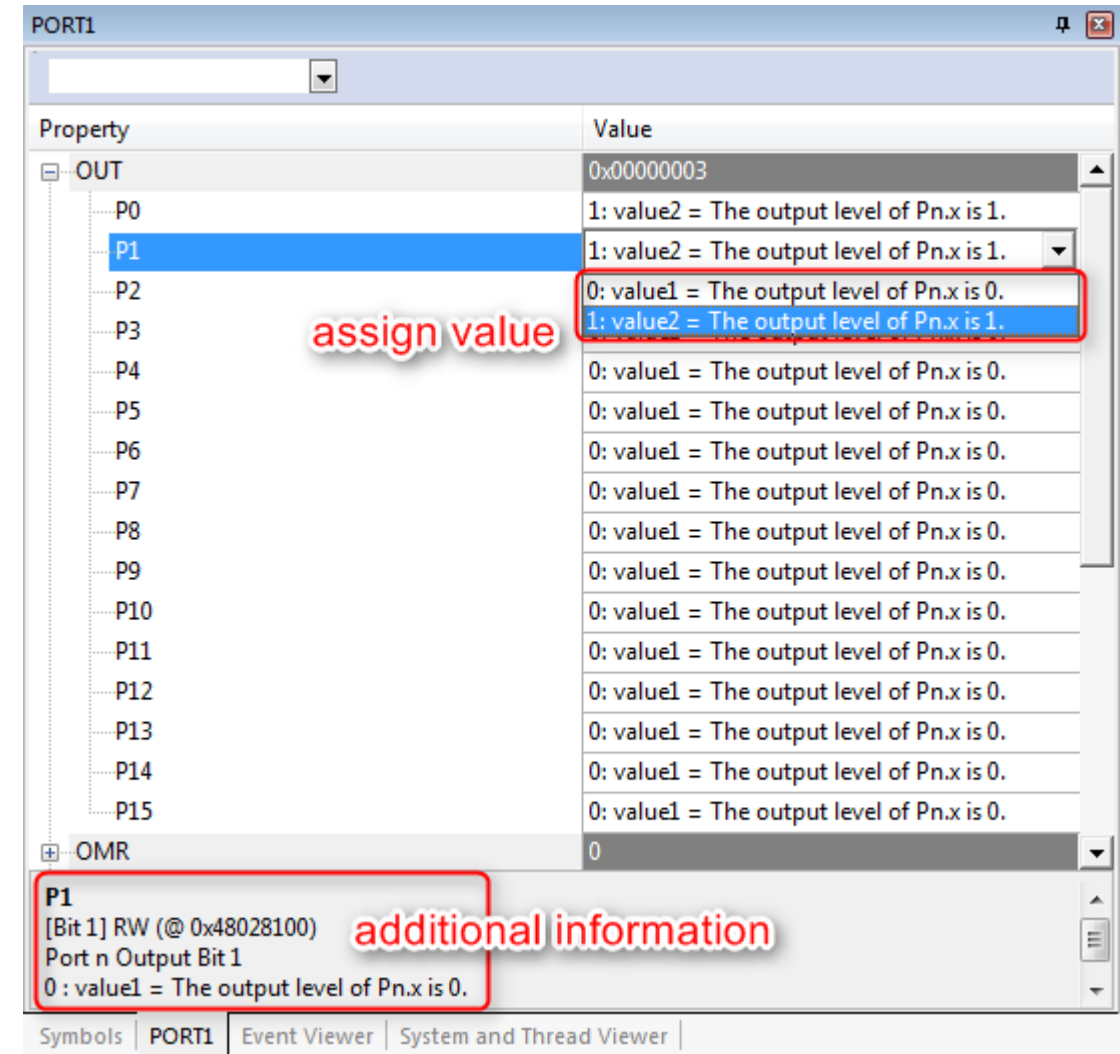


Num	Name	Count	Total Time	Min Time In	Max Time In	Min Time Out	Max Time Out	First Time [s]	Last Time [s]
2	NonMaskable	0	0 s						
3	HardFault	0	0 s						
4	MemoryManage...	0	0 s						
5	BusFault	0	0 s						
6	UsageFault	0	0 s						
11	SVCall	1	0 s					0.00094083	0.00094083
12	DebugMonitor	0	0 s						
14	PendSV	0	0 s						
15	SysTick	13622	28.564 ms	1.967 us	4.650 us	995.375 us	998.033 us	0.00191702	13.62291700
16	ExtIRQ 0	0	0 s						
17	ExtIRQ 1	0	0 s						
18	ExtIRQ 2	0	0 s						
19	ExtIRQ 3	0	0 s						
20	ExtIRQ 4	0	0 s						
21	ExtIRQ 5	0	0 s						

Call Stack + Locals | Watch 1 | Trace Exceptions | Event Counters | Memory 1

System Viewer

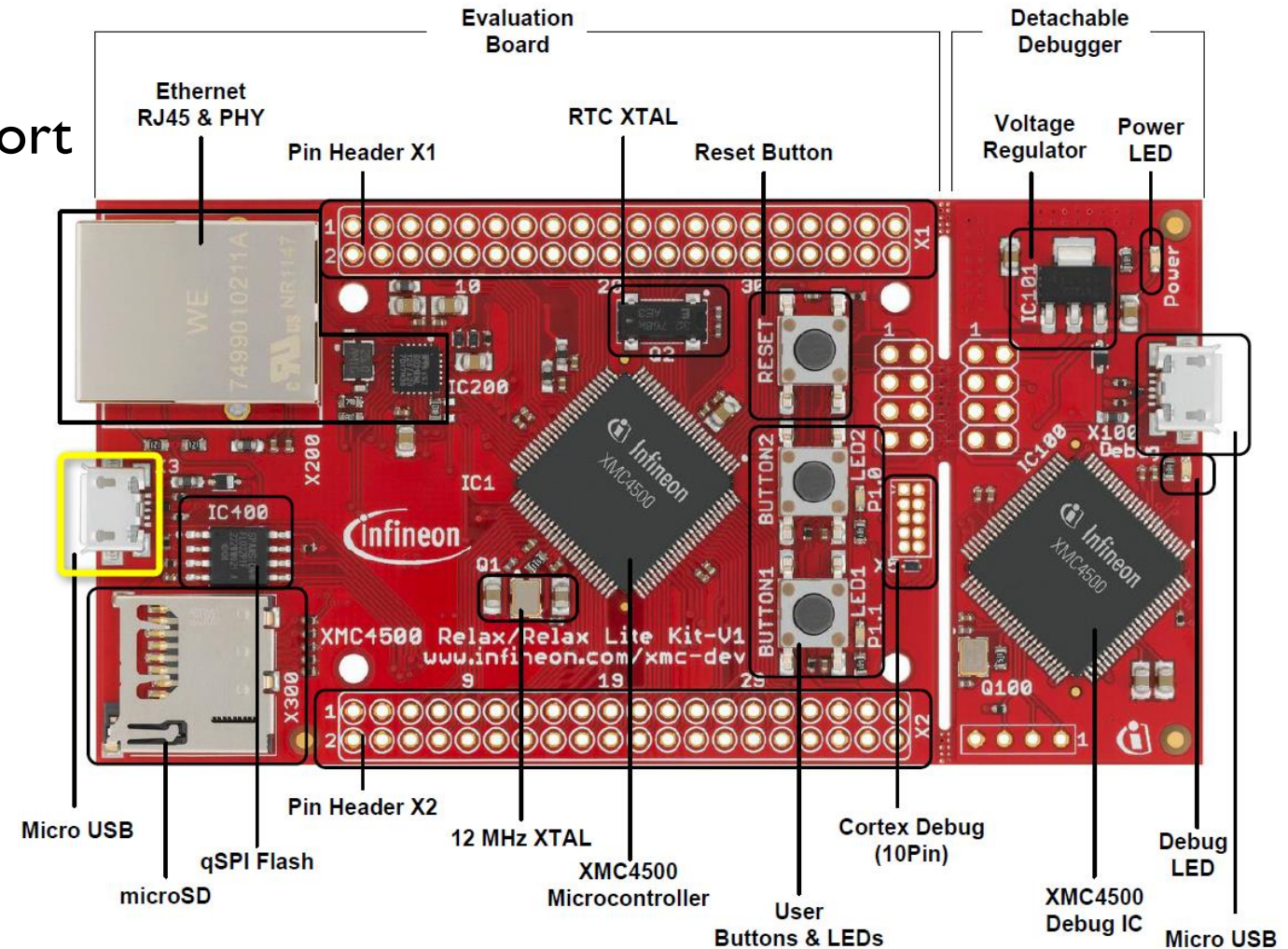
- Displays information about peripheral registers
- Allows on-the-fly debugging of Special Function Registers (SFRs)
- Use it to:
 - View peripheral register property values
 - View additional information about a property
 - Change properties at runtime (click into the value field and enter a new value)



Extending the Project with USB

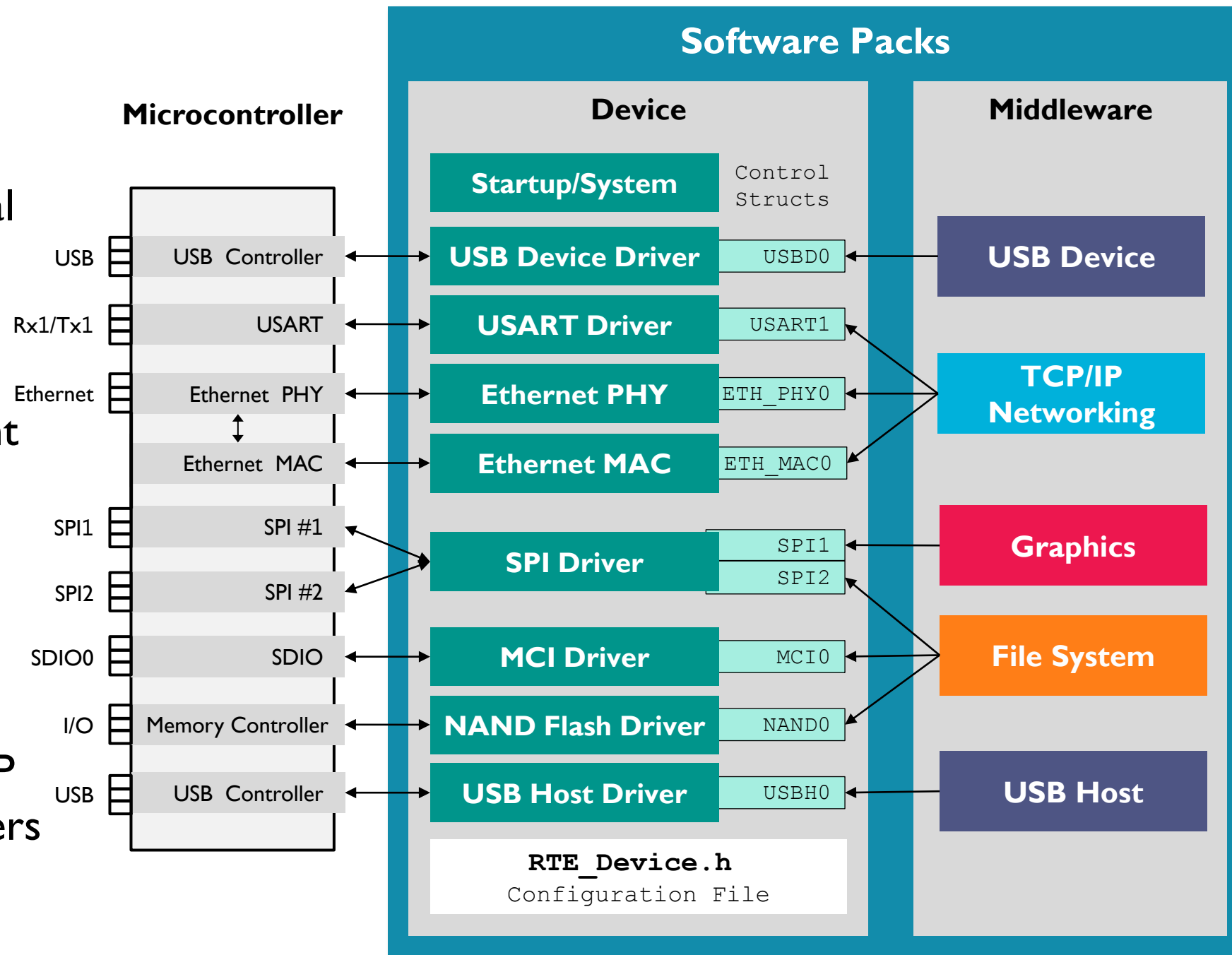
Adding USB Functionality to the Project

- To control the LEDs via a PC based program, we will use the Micro USB port next to the RJ45 connector




CMSIS-Driver 2.0

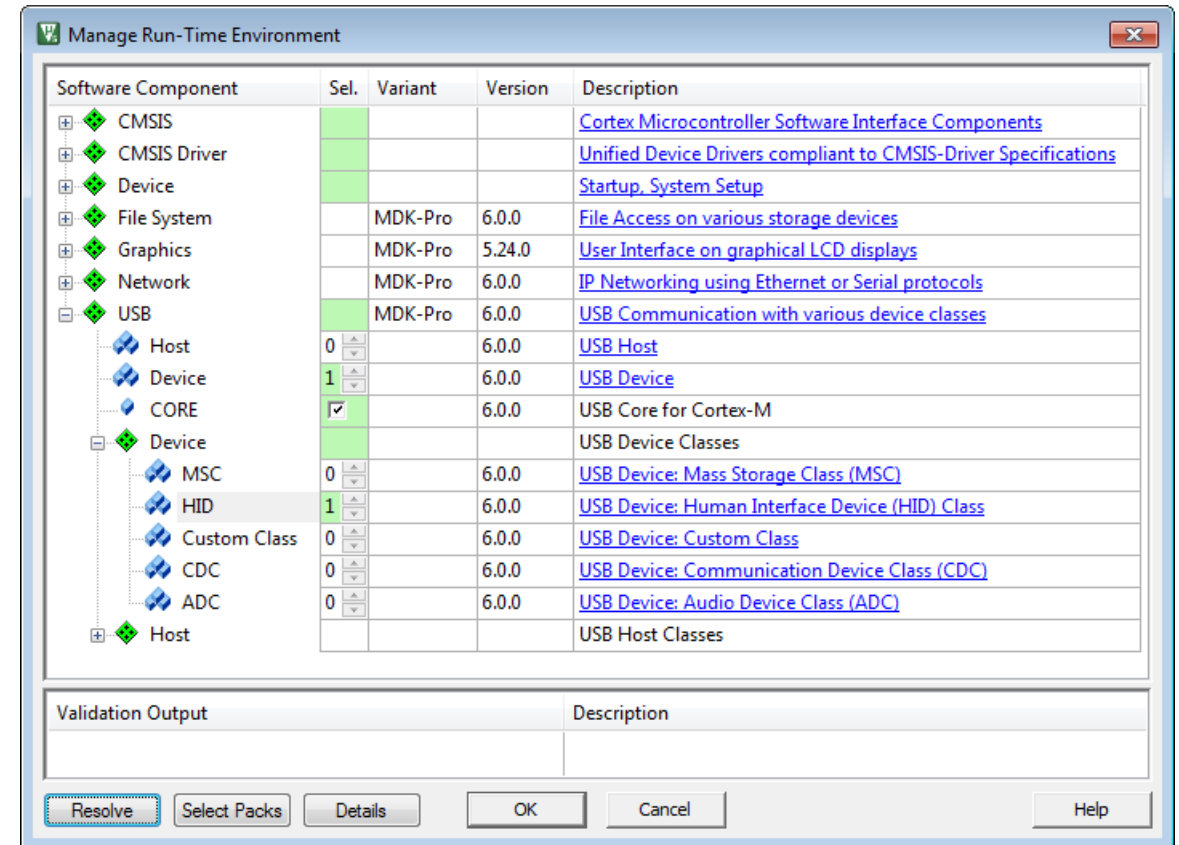
- API describing peripheral driver interfaces for middleware stacks and user applications
- Generic and independent of a specific RTOS
- Covers a wide range of use cases for the supported peripheral types
- Infineon::XMC4000_DFP provides compliant drivers for USB



Manage Run-Time Environment

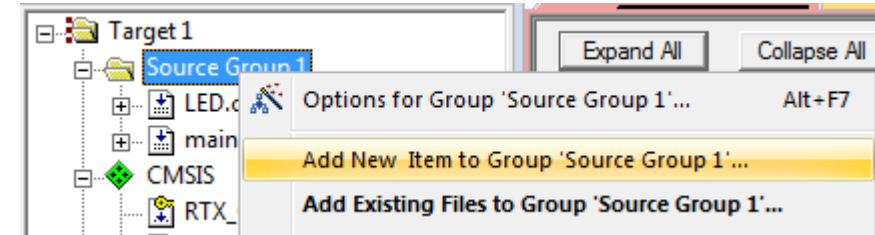
Select USB Middleware Components

- Stop the Debug session
- Click on 
- Expand **USB**
 - Select **CORE**
- Expand **USB:Device**
 - Set **HID** to 1
- Click **Resolve**
- Click **OK**

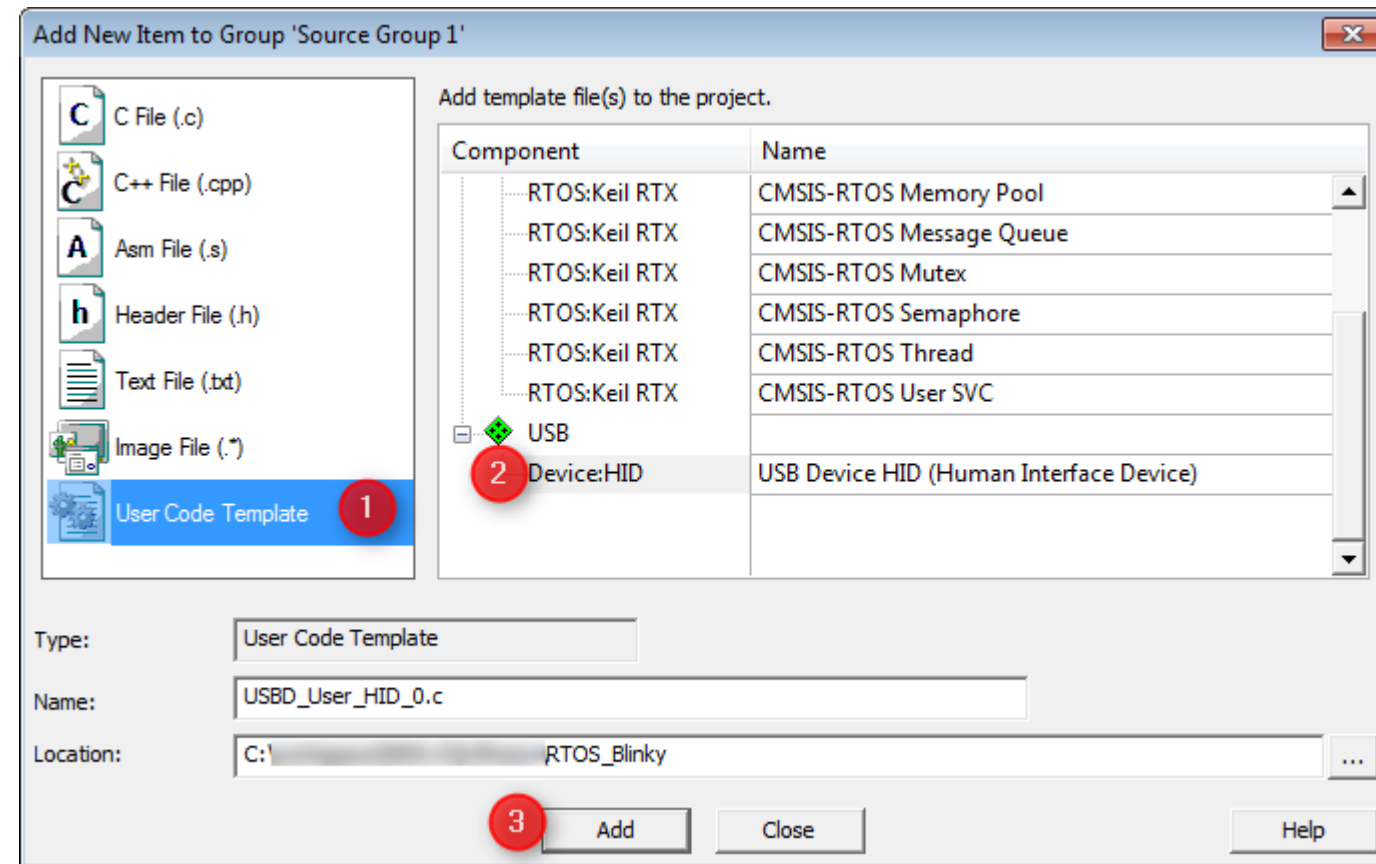


Add USB HID Template File

- Right click on **Source Group 1** and select **Add New Item to Group 'Source Group 1'...**



- Click on **User Code Template**
- Scroll down until you see **Device:HID**. Click on it
- Click **Add**



USBDUser_HID_0.c Template File (I)

- It implements the application specific functionality of the HID class and is used to receive and send data reports to the USB Host
- The implementation must match the configuration file USBD_Config_HID_0.h
- Add the following:
 - `#include "LED.h"`

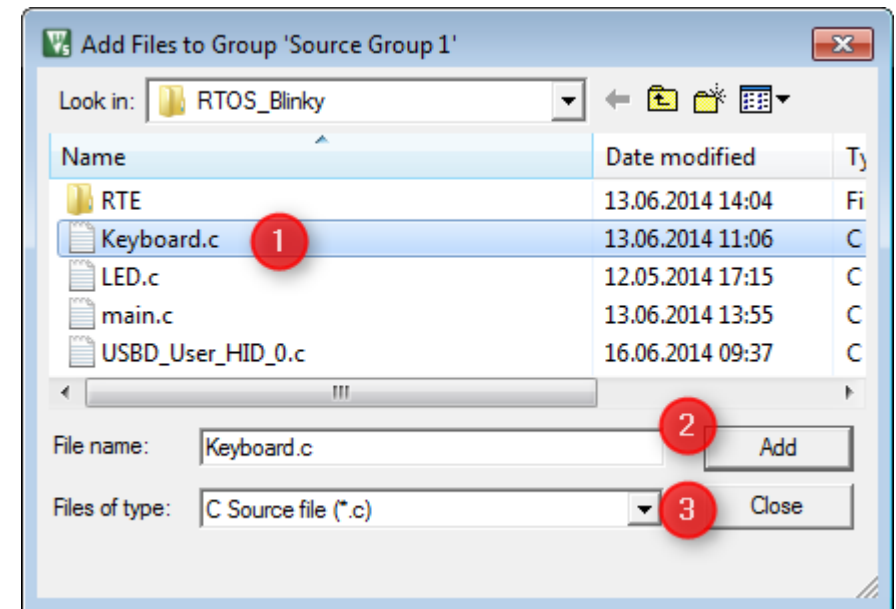
USBDUser_HID_0.c Template File (2)

- It contains function prototypes that need to be adapted to the user application:

```
■ bool USBD_HID0_SetReport (uint8_t rtype, uint8_t req, uint8_t rid, const
uint8_t *buf, int32_t len) {
    uint8_t i;
    switch (rtype) {
        case HID_REPORT_OUTPUT:
            for (i = 0; i < 4; i++) {
                if (*buf & (1 << i))
                    LED_On (i);
                else
                    LED_Off (i);
            }
            break;
        case HID_REPORT_FEATURE:
            break;
    }
    return true;
}
```

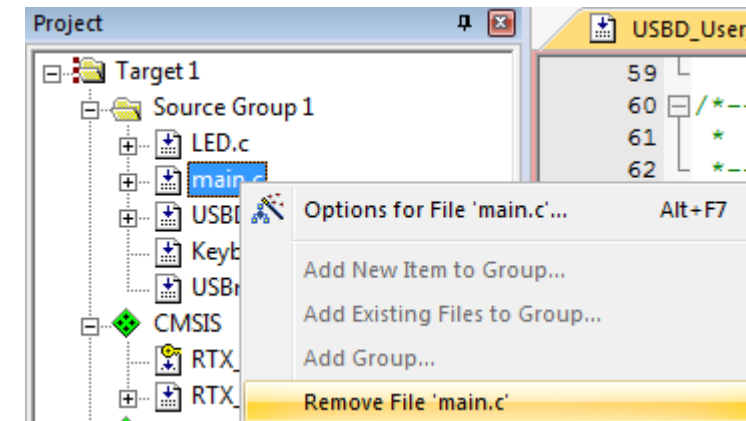
Using the Push-buttons

- BUTTON1 and BUTTON2 can be used in the HID program as well
 - Copy Keyboard.c and Keyboard.h to your project directory
 - Add Keyboard.c to the project:
 - Right click on **Source Group 1** and select **Add Existing Files to Group 'Source Group 1'...**
1. Click on **Keyboard.c**
 2. Click **Add**
 3. Click **Close**



Implement the Application

- Copy USBmain.c to your project directory
- Add USBmain.c to the project:
- Right click on **Source Group 1** and select **Add Existing Files to Group 'Source Group 1'...**
 1. Click on **USBmain.c**
 2. Click **Add**
 3. Click **Close**
- Right-click **main.c** and select **Remove File 'main.c'**



RTOS Configuration

- USB demands additional resources for the RTOS
- Change the following in `RTX_Conf_CM.c`:
 - Default Thread stack size (`OS_STKSIZE`): 512 bytes (value: 128)
 - Main Thread stack size (`OS_MAINSTKSIZE`): 512 bytes (value: 128)
 - Number of threads with user-provided stack size (`OS_PRIVCNT`): 4
 - Total stack size [bytes] for threads with user-provided stack size (`OS_PRIVSTKSIZE`): 2048 bytes (value: 512)
 - RTOS Kernel Timer input clock frequency [Hz] (`OS_CLOCK`): 120000000

Run the Application

- Build the application (F7)
- Download it into Flash ()
- Unplug the USB cable
- Plug it into the Micro-USB port **X3** next to the RJ45
- You might see a notification from Windows that drivers are being installed

HID Client Application

C:\Keil\ARM\Utilities\HID_Client\Release\HIDClient.exe

- A Windows program is available to test the functionality of the code
- The HIDClient.exe utility is located in "C:\Keil\ARM\Utilities\HID_Client\Release" and can run stand-alone without the need to install the software
- To check the client utility with your board, do the following:
 - Download the application to your board.
 - Verify all jumper settings on the board.
 - Connect the board to a Windows PC. The PC should recognize the HID device and install the correct driver automatically.
 - Run "C:\Keil\ARM\Utilities\HID_Client\Release\HIDClient.exe".
 - Select the Device to establish the communication channel.
 - Test the application by enabling the correct LEDs (here 0 and 1) and by pressing the push-buttons on the development board



MDK Professional Evaluation License

- The license you have received today will expire 2nd July, 2014
- If you want to test the Middleware longer, please visit
<https://www.keil.com/trial>
- You will receive a 30-Day Free Trial License for MDK Professional

μVision IDE Features

Context Sensitive F1 Help

- Pressing F1 with cursor on any keyword will bring up the related help page:

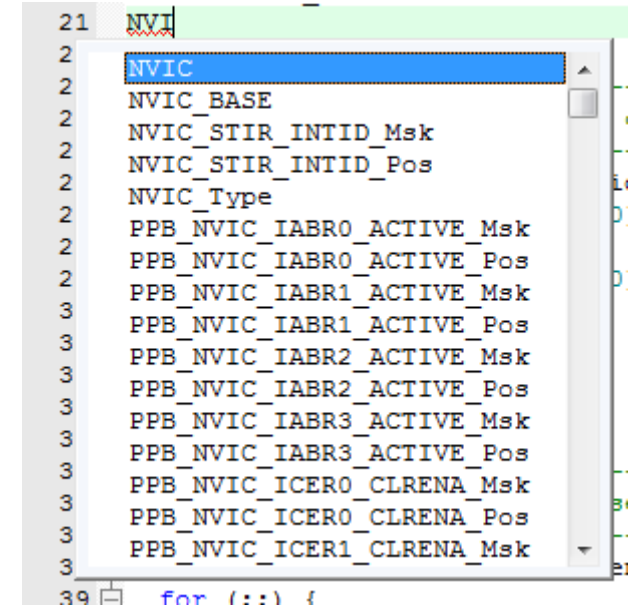
```
73 /*-----  
74 *      Main: Initialize and start RTX Kernel  
75 *-----  
76 int main (void) {  
77     SystemCoreClockUpdate();  
78     LED_Init();  
79     /* Initialize the  
80  
81     tid_phaseA = osThreadCreate(osThread(phaseA), NULL);  
82     tid_phaseB = osThreadCreate(osThread(clock), NULL);  
83     tid_clock = osThreadCreate(osThread(clock), NULL);  
84  
85     osSignalSet(tid_phaseA, 0x0001);  
86     /* set signal to p  
87     osDelay(osWaitForever);
```

Cursor on SystemCoreClockUpdate

The screenshot shows the CMSIS-CORE help page for the `SystemCoreClockUpdate` function. The page is titled "CMSIS-CORE Version 3.30" and "CMSIS-CORE support for Cortex-M processor-based devices". The left sidebar contains a navigation menu with categories like "CMSIS-CORE", "Reference", and "System and Clock Configuration". The "System and Clock Configuration" category is expanded, showing "SystemCoreClockUpdate" as the selected item. The main content area displays the function signature `void SystemCoreClockUpdate (void)` and its description: "Updates the variable `SystemCoreClock` and must be called whenever the core clock is changed during program execution. The function evaluates the clock register settings and calculates the current core clock." Below this, the `SystemInit` function is also shown. At the bottom, the "Variable Documentation" section describes the `uint32_t SystemCoreClock` variable, stating it holds the system core clock frequency and can be used by debuggers to query the frequency of the clock.

Code Completion

- List showing all program symbols that contain the currently typed characters
- List appears after typing:
 - 3 characters (default)
 - A **trigger character**:
 - . For structures or classes
 - > For pointer structures
 - :: For symbols within a specific scope
- **CTRL+<space>**
- Insert the highlighted list-item into the code by pressing:
 - Tab, space, or enter
 - Typing a bracket
 - Any trigger character



Parameter Information

- For a function or method Parameter Information shows:
 - Parameter names
 - Amount of parameters
 - Parameter types

```
osSignalSet(via_clock, 0x0100), /* set signal to clock thread */  
osSignalSet(  
osDelay(500); int32_t osSignalSet(osThreadId thread_id, int32_t signals) */  
osSignalSet(tid_clock, 0x0100); /* set signal to clock thread */
```

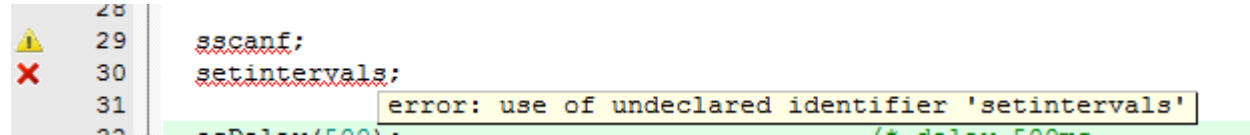
- While typing, the bolded text indicates the next required parameter

Dynamic Syntax Checking while Typing

- Validates program syntax
 - Alerts to potential code violations before compilation

- Errors/warnings shown by:

- Squiggly red lines in the editor
- Icons next to the line number



- Hover the mouse on an icon for details about the syntax violation

- Also in:

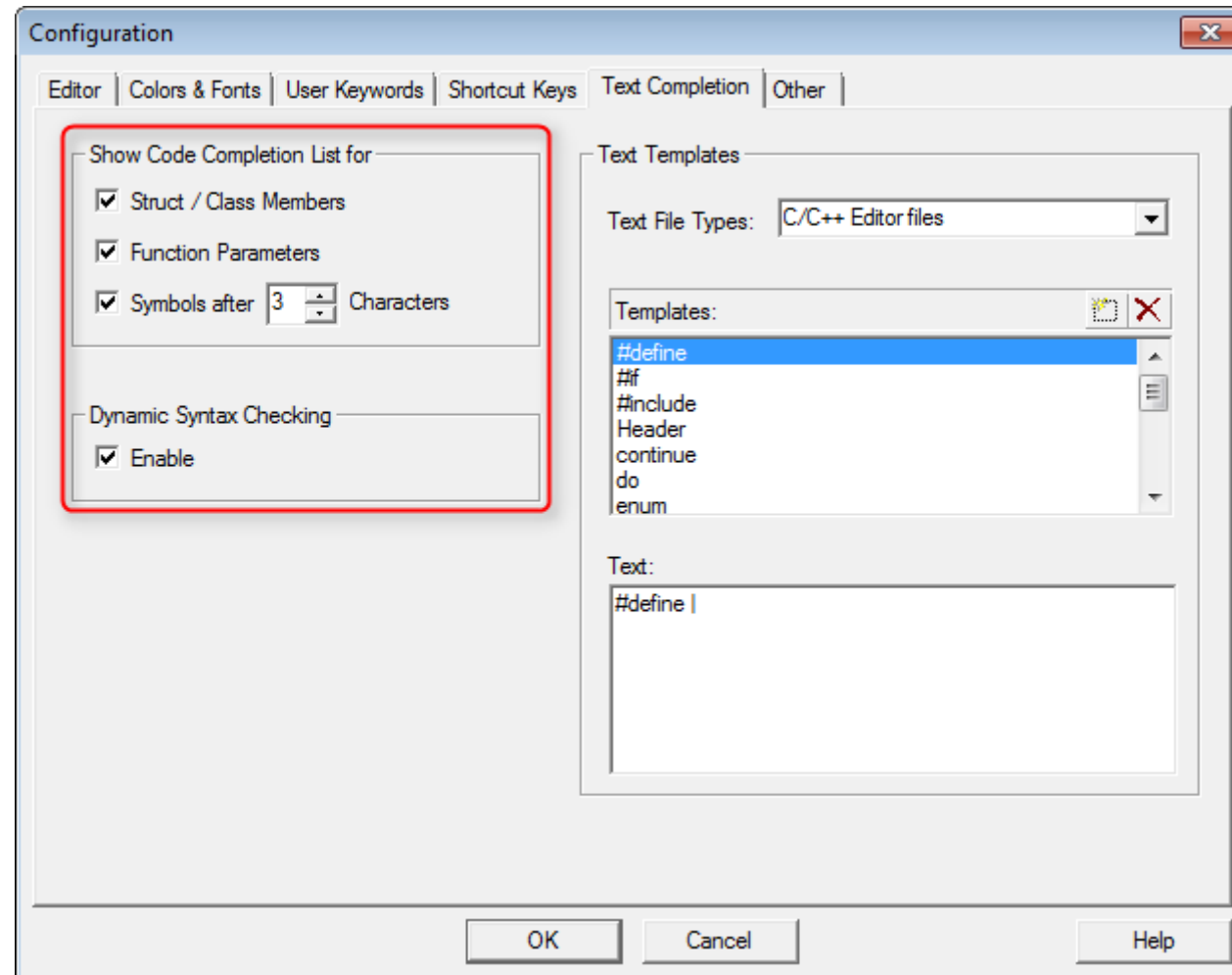
- View → Error List Window

Error List			
main.c: Errors: 1 - Warnings: 2		File	Line
E	error: use of undeclared identifier 'setintervals'	main.c	30
W	warning: implicitly declaring library function 'scanf' with type 'int (const char *restrict, cons...	main.c	29
W	warning: expression result unused	main.c	29

Build Output Error List

Feature Configuration

- Go to **Edit** → **Configuration**, click on the **Text Completion** tab



Collaterals on USB Stick

- AN258 - Using DAVE3 with MDK Version 5
- AN260 - Infineon XMCI100 2Go Lab for uVision V5
- AN263 - Infineon XMCI200 Boot Kit Lab for uVision V5
- Getting Started – Create Applications with MDK Version 5
- μ Vision Keyboard Shortcuts

Links on USB Stick

- MDK Version 5 Overview: <http://www2.keil.com/mdk5>
- MDK Version 5 Download: <https://www.keil.com/demo/eval/arm.htm>
- Infineon on keil.com: <http://www2.keil.com/infineon>
- Keil MDK for Infineon XMC1000: <http://www2.keil.com/infineon/mdk>
- Support: www.keil.com/support
- Forums: <http://www.keil.com/forum>
- Application Notes: <http://www.keil.com/appnotes>
- Keil Tools on ARM Connected Community: <http://community.arm.com/groups/tools>