Keil MDK Version 5
Component-based Software Development

Christopher Seidl
Technical Marketing Manager, DSG

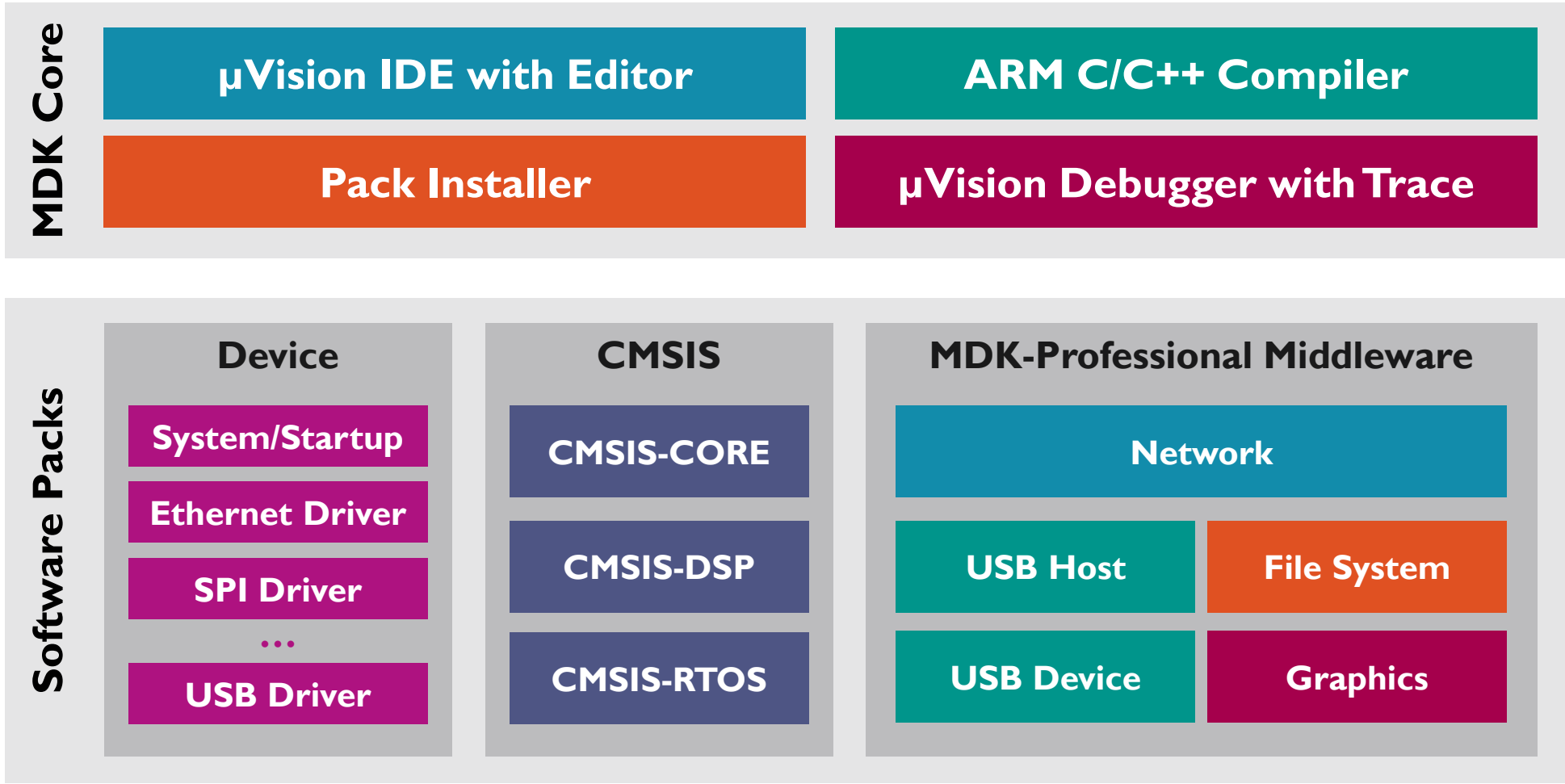
Installing the Workshop Material

- 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)

ARM

Software Packs in Keil MDK Version 5

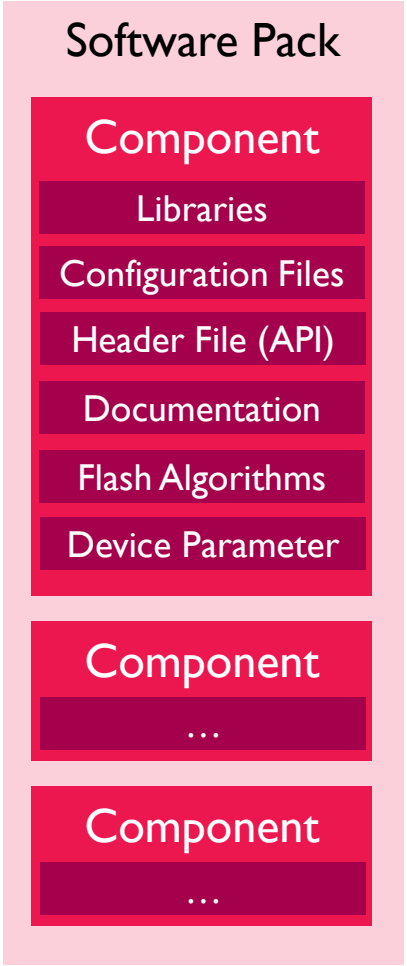


ARM

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

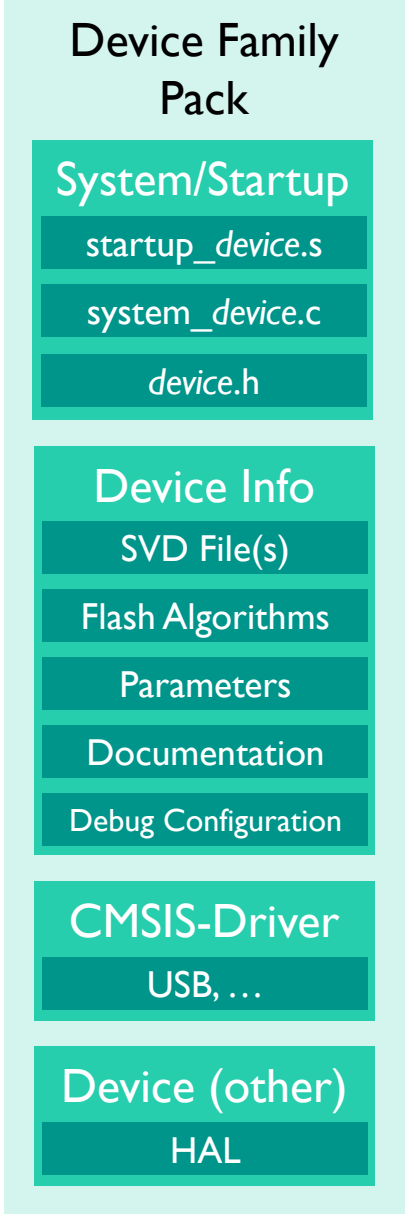


ARM

Device Family Pack

Example: Infineon.XMC4500_DFP

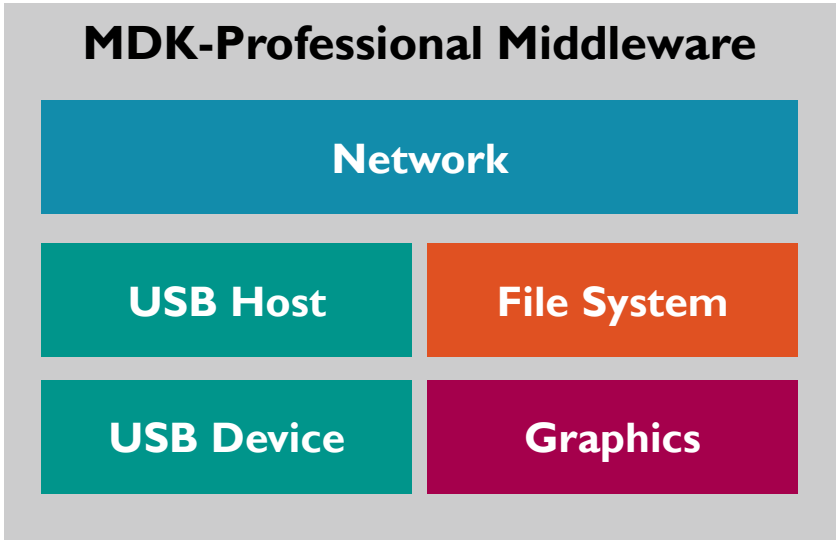
- Device family and devices
  - System and startup code, device header files
  - SVD files
  - Flash algorithms
- Functional grouping of files
- Development board information (if available)
- Usage of source code and library files for:
  - Specific processors
  - Tool chains
- Example Projects
- User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



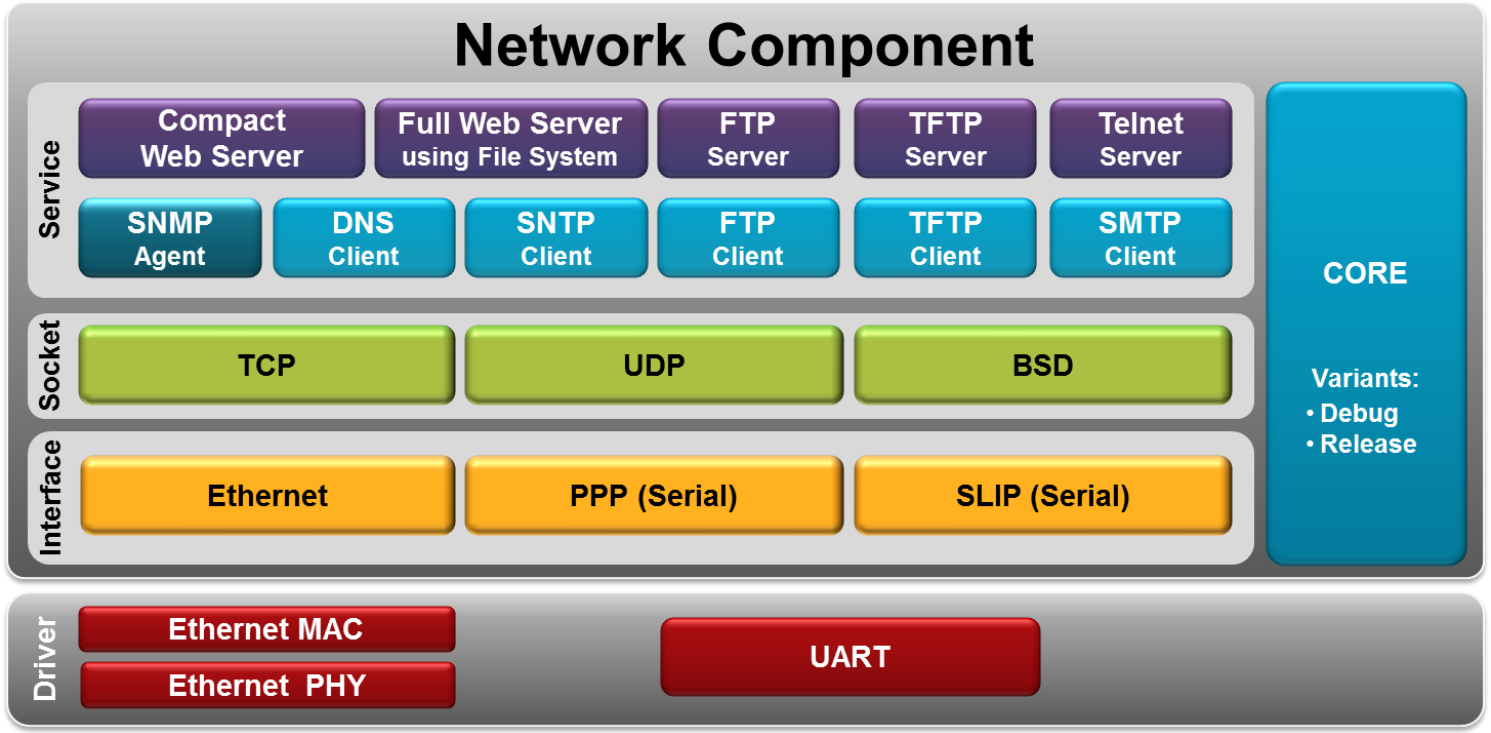
ARM

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

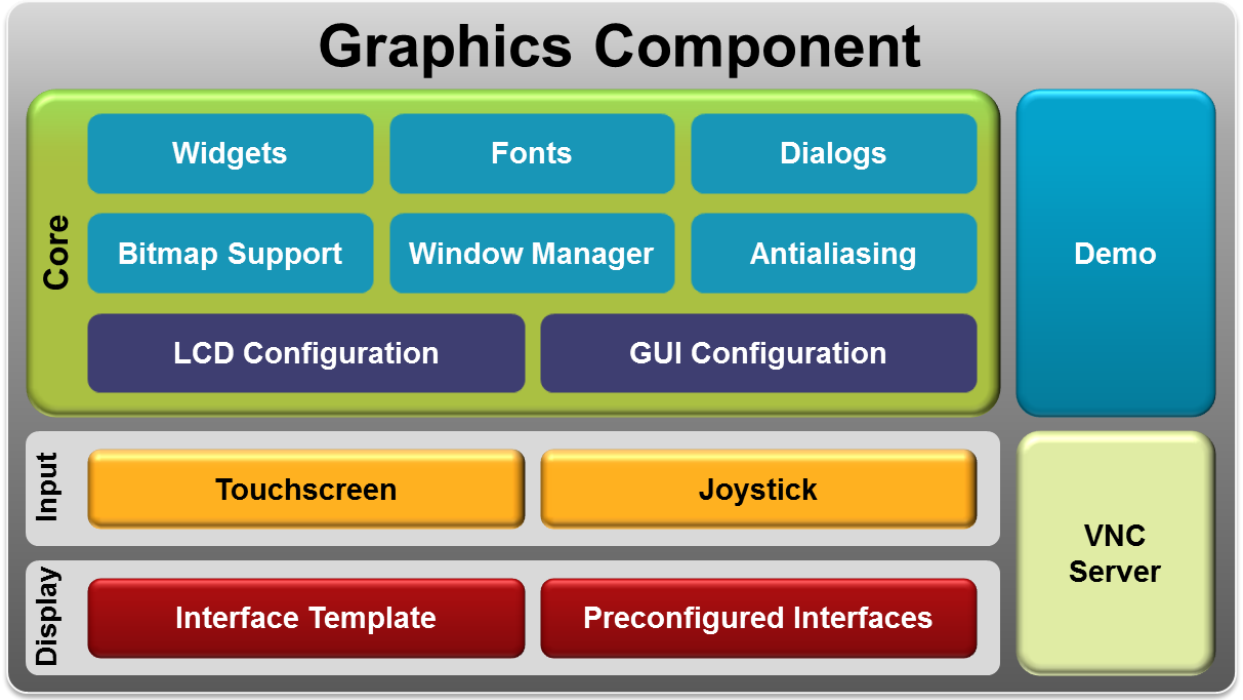


Middleware: Network



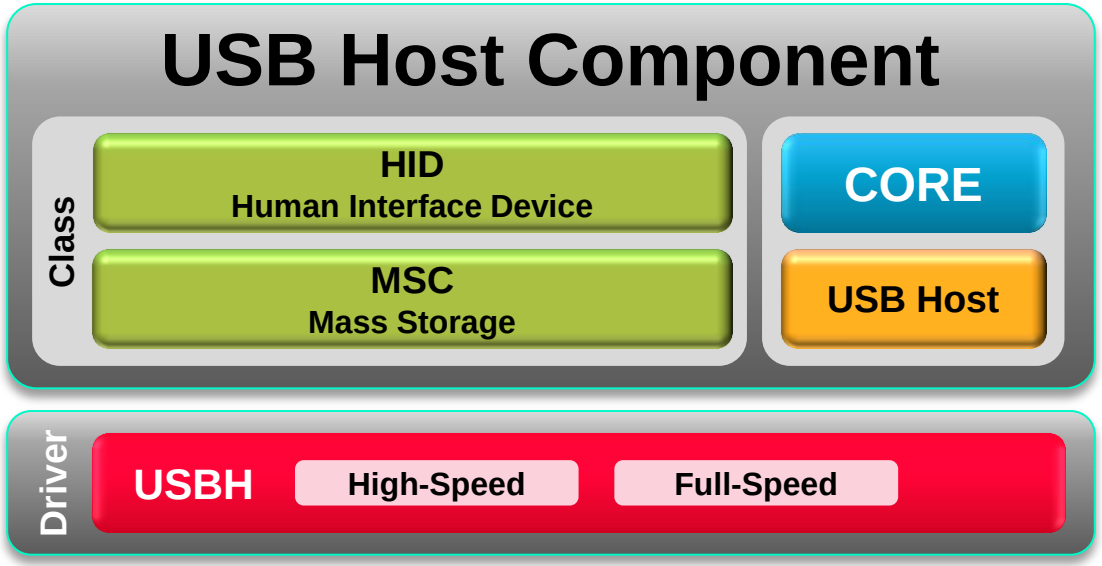
- Add TCP/IP network connectivity via Ethernet or Serial
- Extensive range of service applications included
- Designed for Cortex-M devices with small memory footprint

Middleware: Graphic



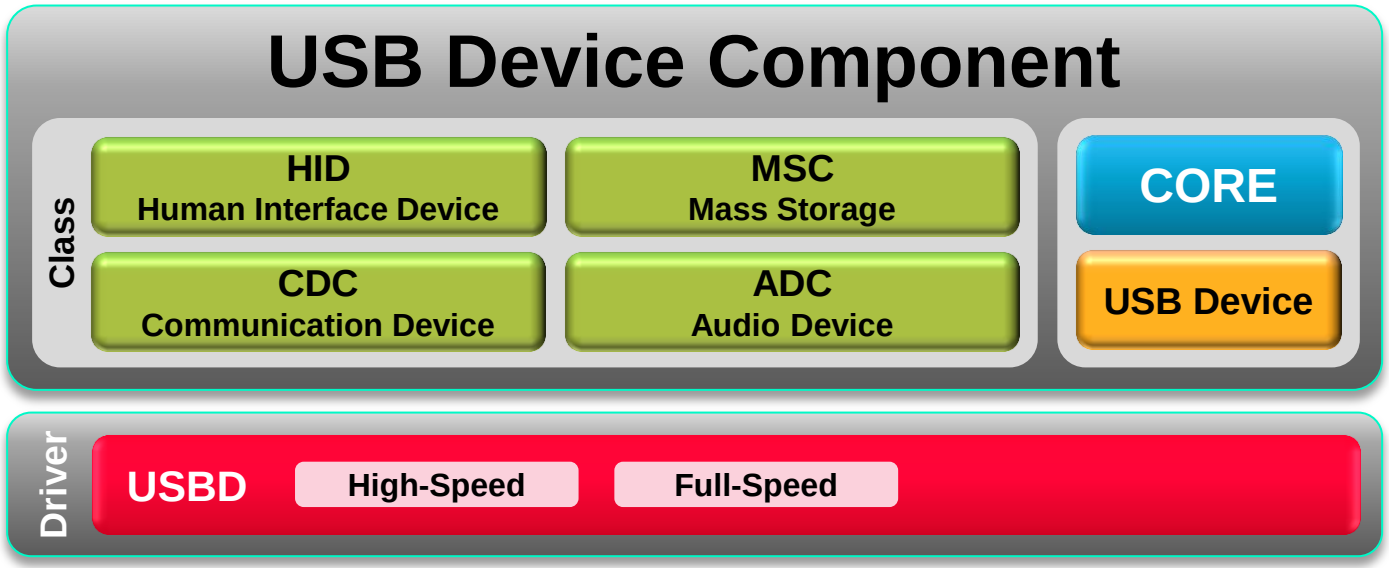
- Add a graphical user interface (GUI) to an application quickly
- Compatible with hundreds of display controllers
- Touch-screen support for many TFT LCDs

Middleware: USB Host



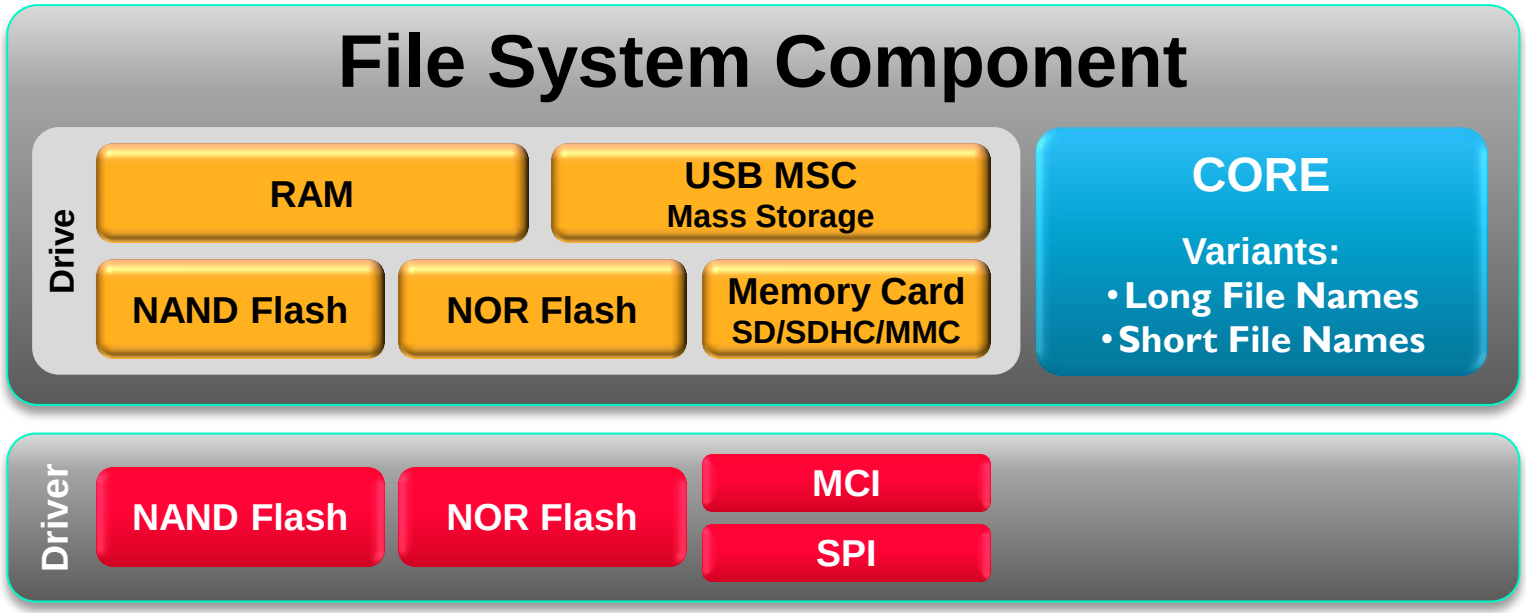
- Connect common USB devices easily
- Support for various popular microcontroller host controllers
- High performance and small footprint
- USB 1.1 Low Speed (1.5 Mbit/s) and Full Speed (12 Mbit/s)
- Supports USB Flash drives and card storage devices

Middleware: USB Device



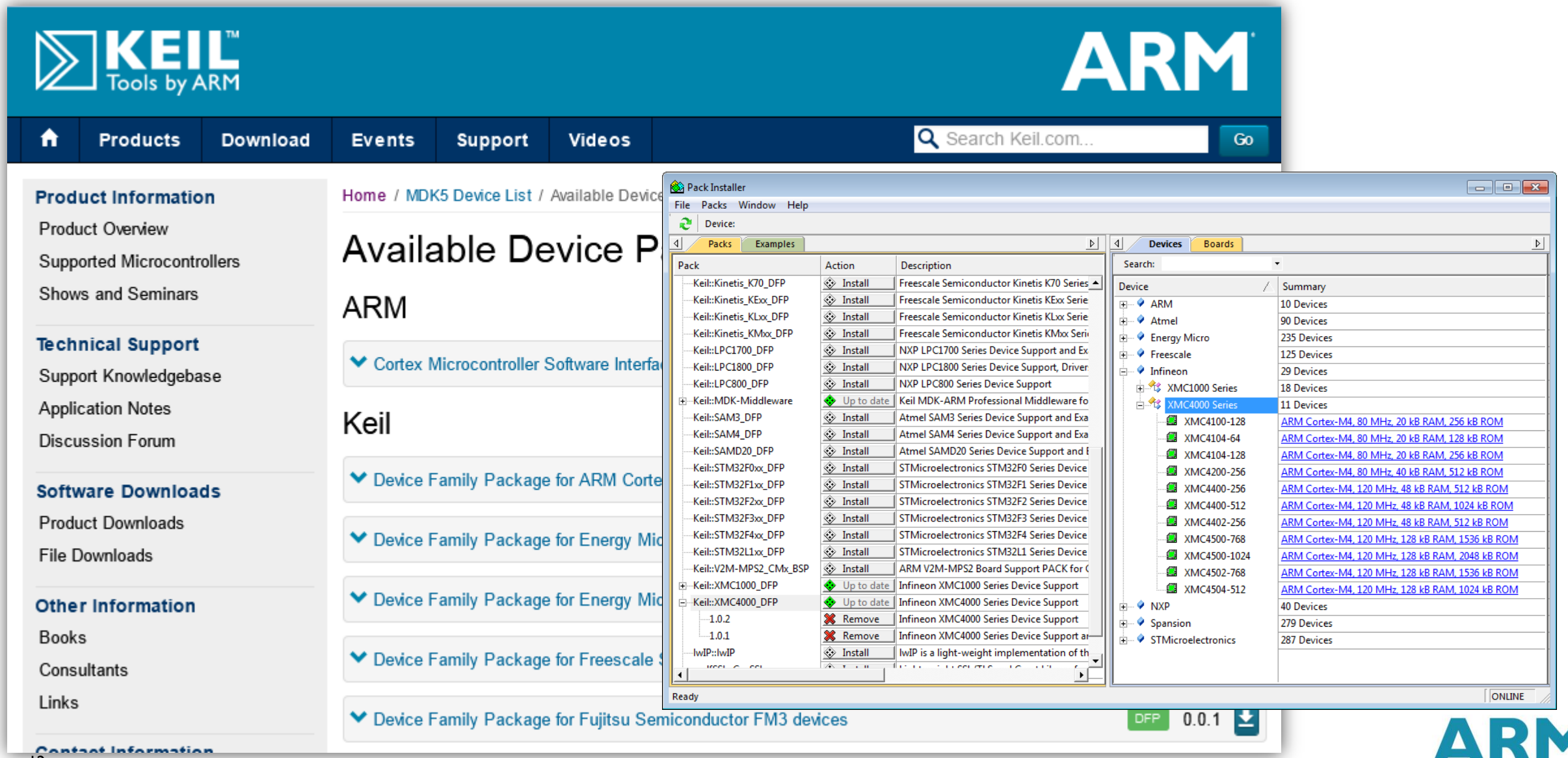
- Standard USB Device classes supported
- Plug'N'Play: No need to develop a driver for Windows or Linux hosts
- USB 1.1 Low Speed (1.5 Mbit/s) and Full Speed (12 Mbit/s)

Middleware: File System

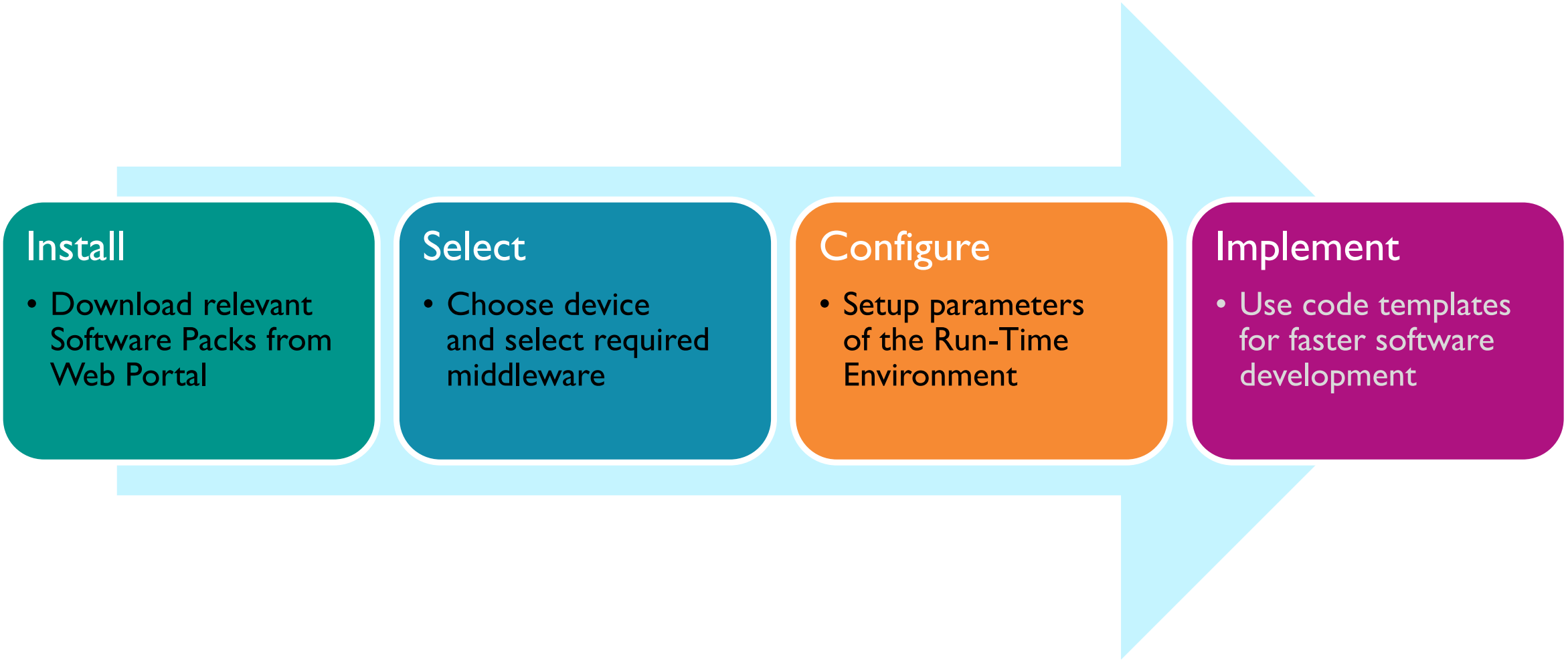


- Enable applications to work with locally stored data
- Storage devices: ROM, RAM, Flash, and SD/MMC/SDHC
- FAT32 file system supported
- Simultaneous access to multiple storage devices

Web Portal for Software Packs

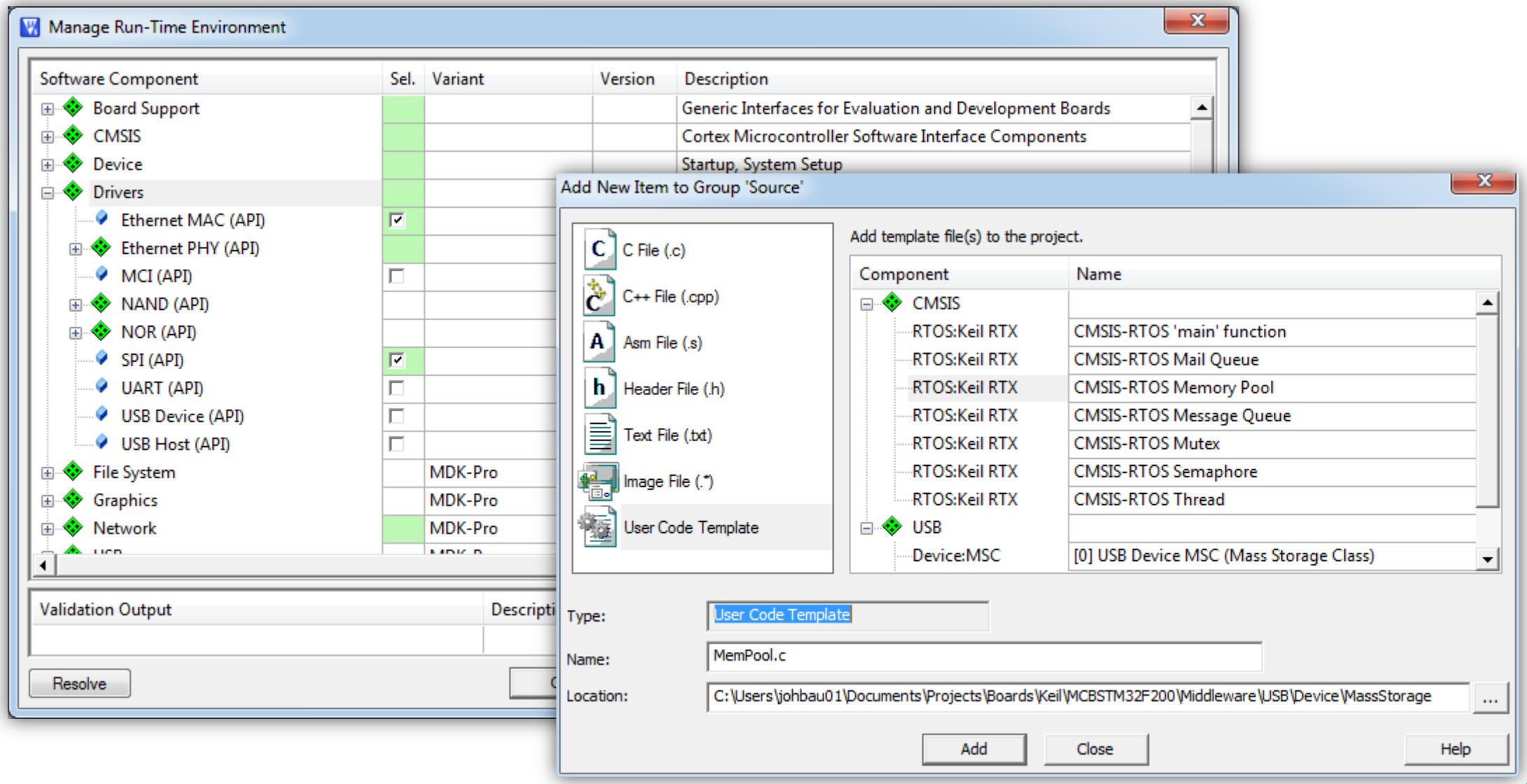


Keil MDK Workflow using Software Packs



ARM

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

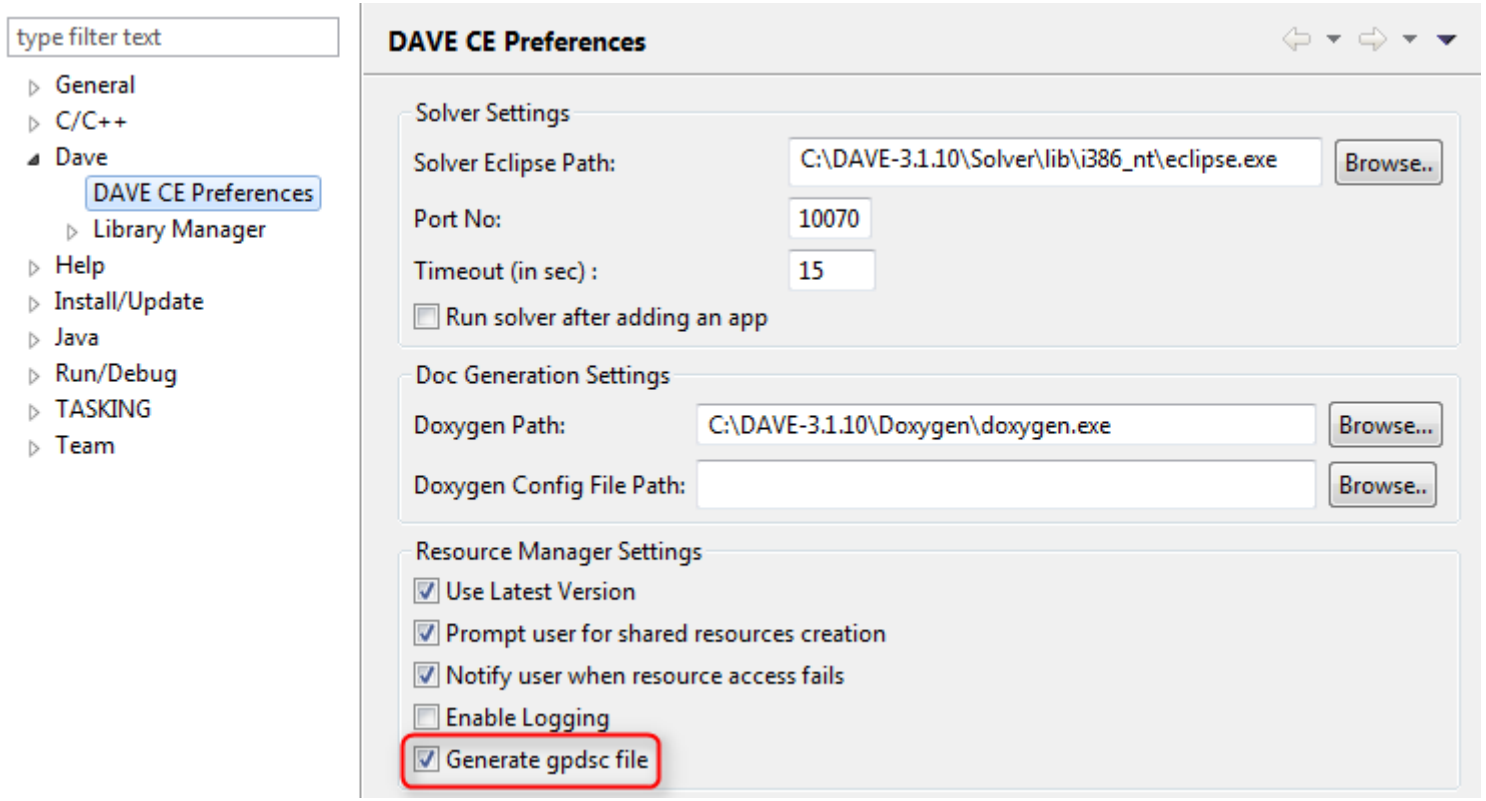
- Support for new devices is provided by Software Packs
- Open standard allows adding third party components

Importing DAVE™ projects in MDK Version 5 Hands-On

ARM

Required Setting for GPDSC Generation

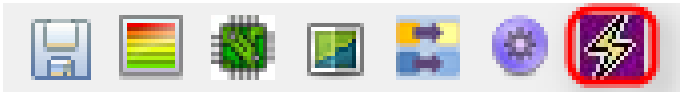
- Go to **Window → Preferences** and **Dave → DAVE CE Preferences**
- Make sure that **Generate gpdsc file** is enabled:



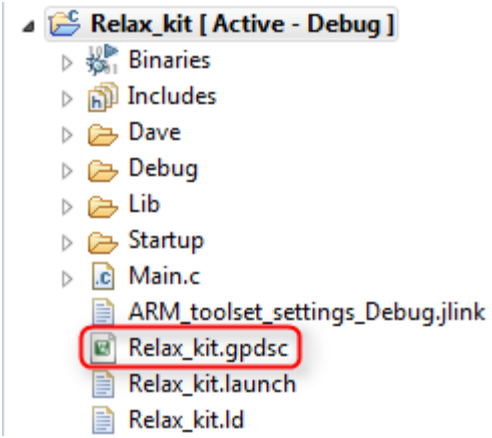
ARM

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: 

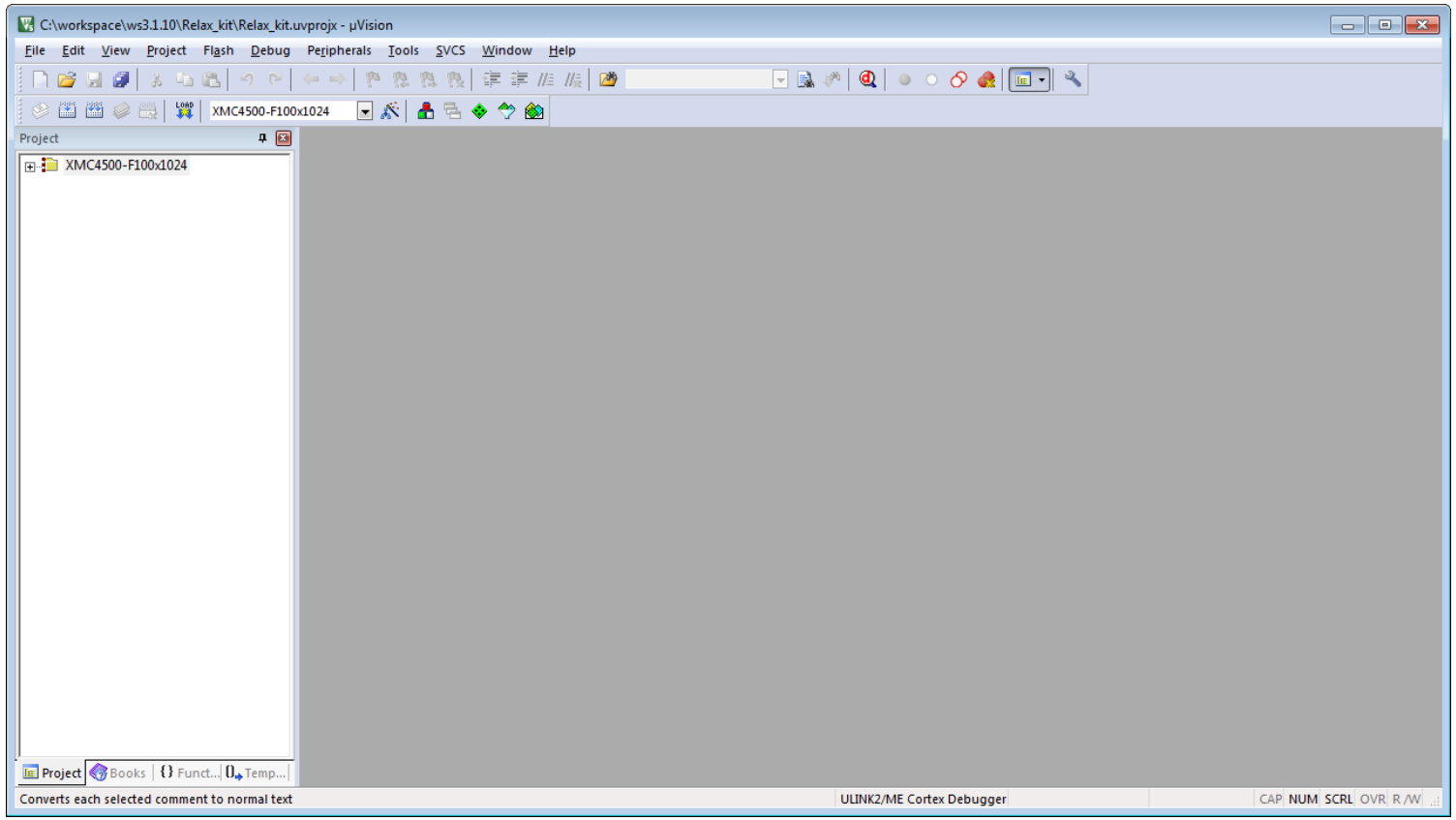
- Check if the GPDSC file has been generated:



ARM

Invoking µVision® from DAVE™

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

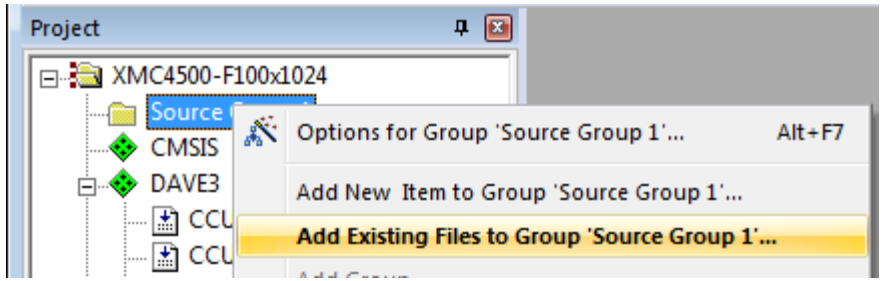


- **Note**: µVision will create a project in the same folder as the DAVE™ project

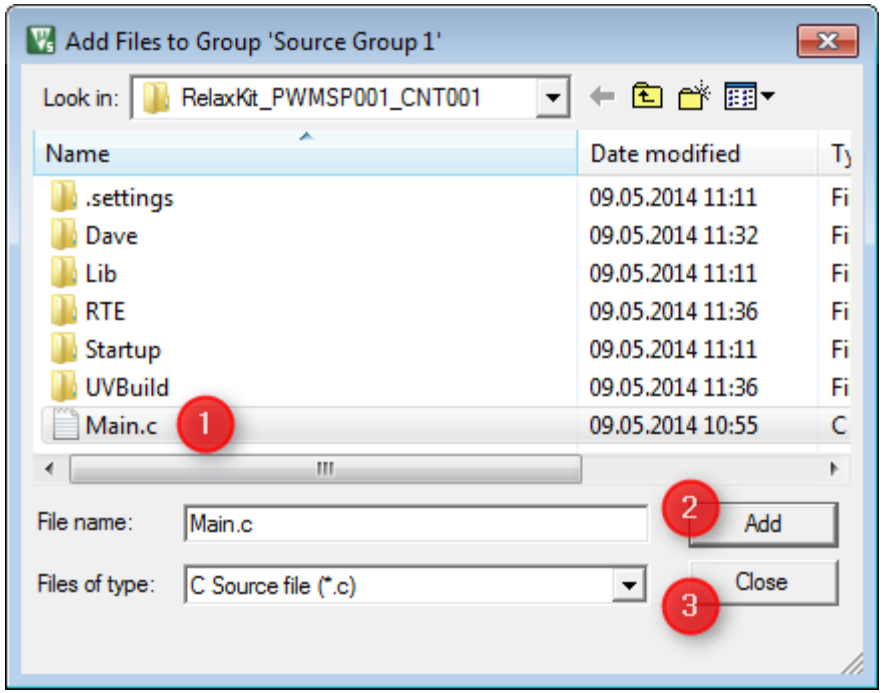
ARM

Add a Main.c File to the µVision® 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'…**

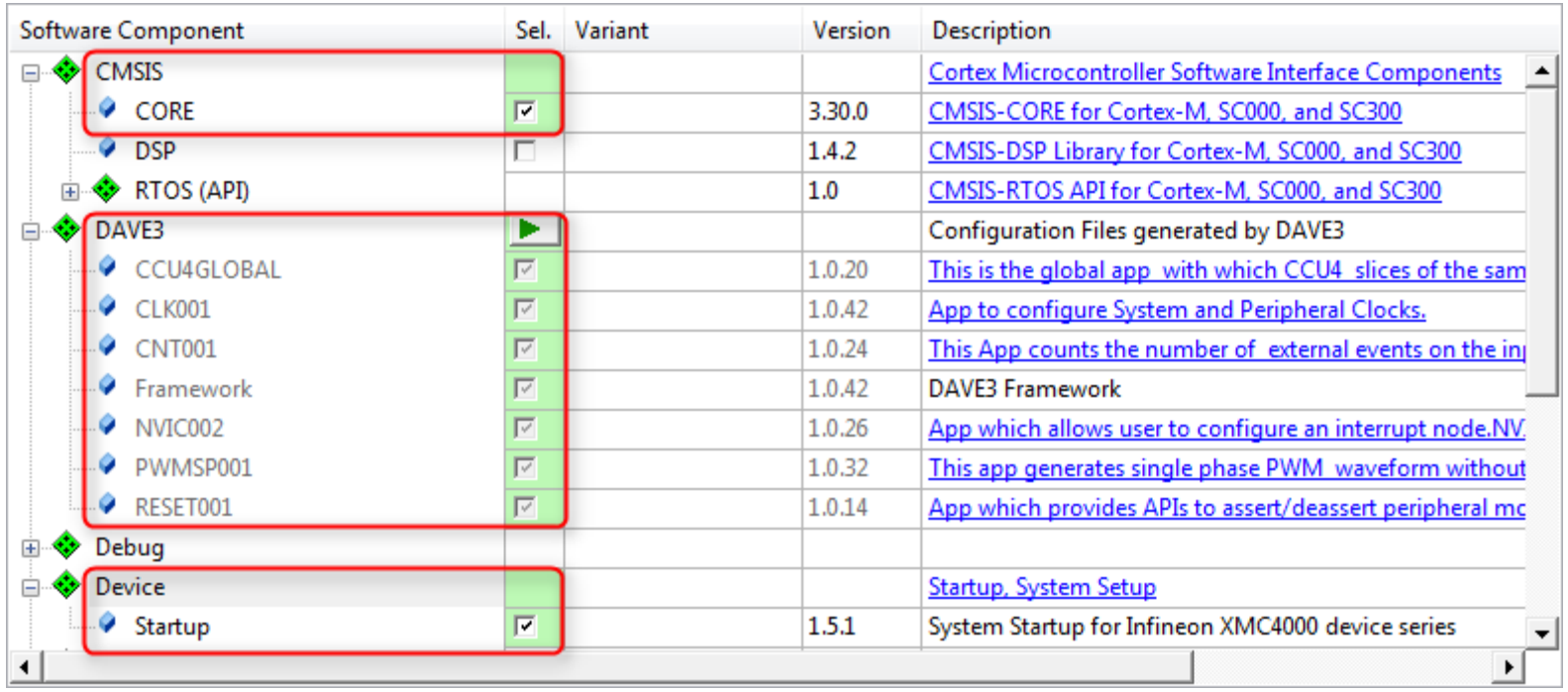


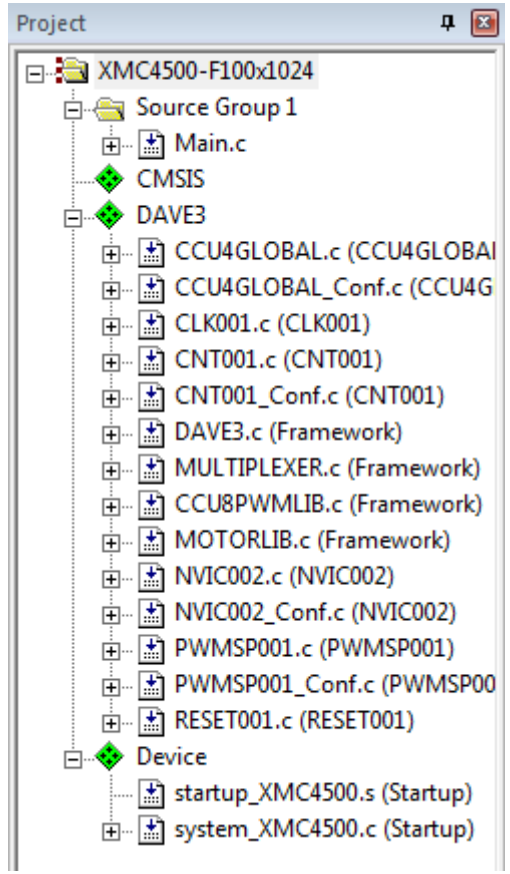
1. Click on **Main.c**
2. Click **Add**
3. Click **Close**



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:

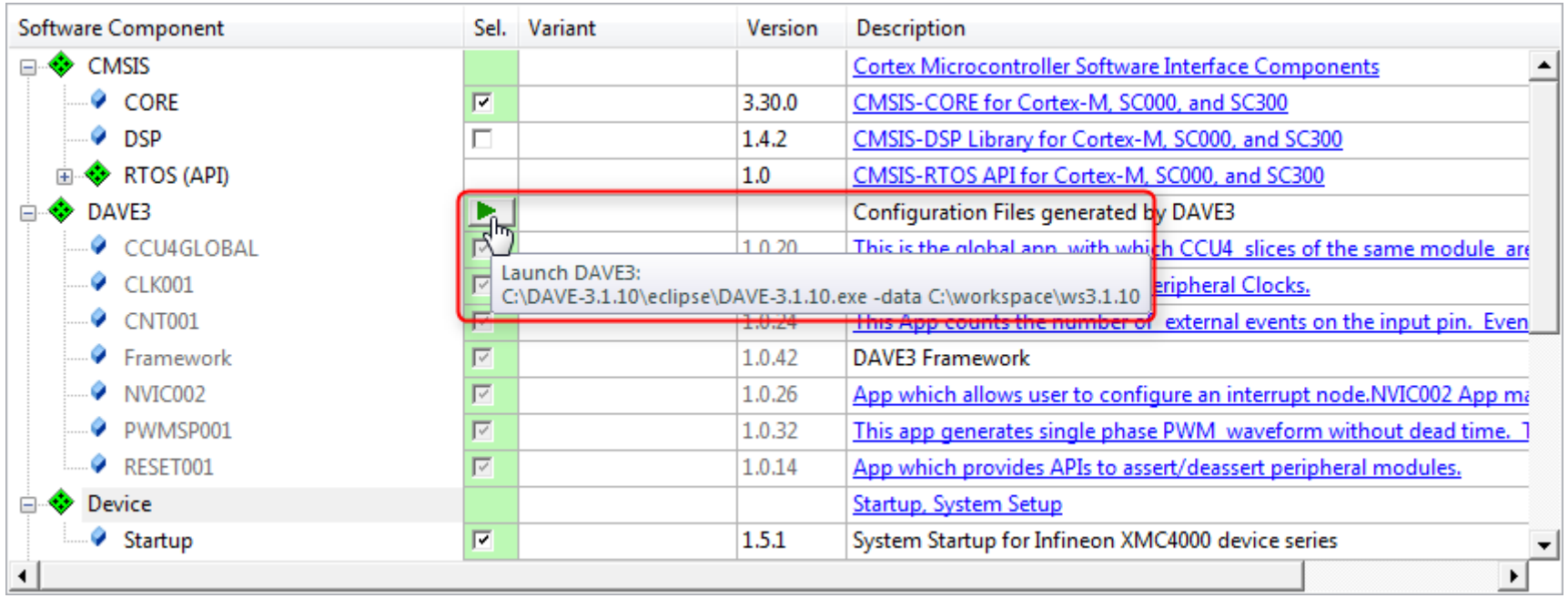




ARM

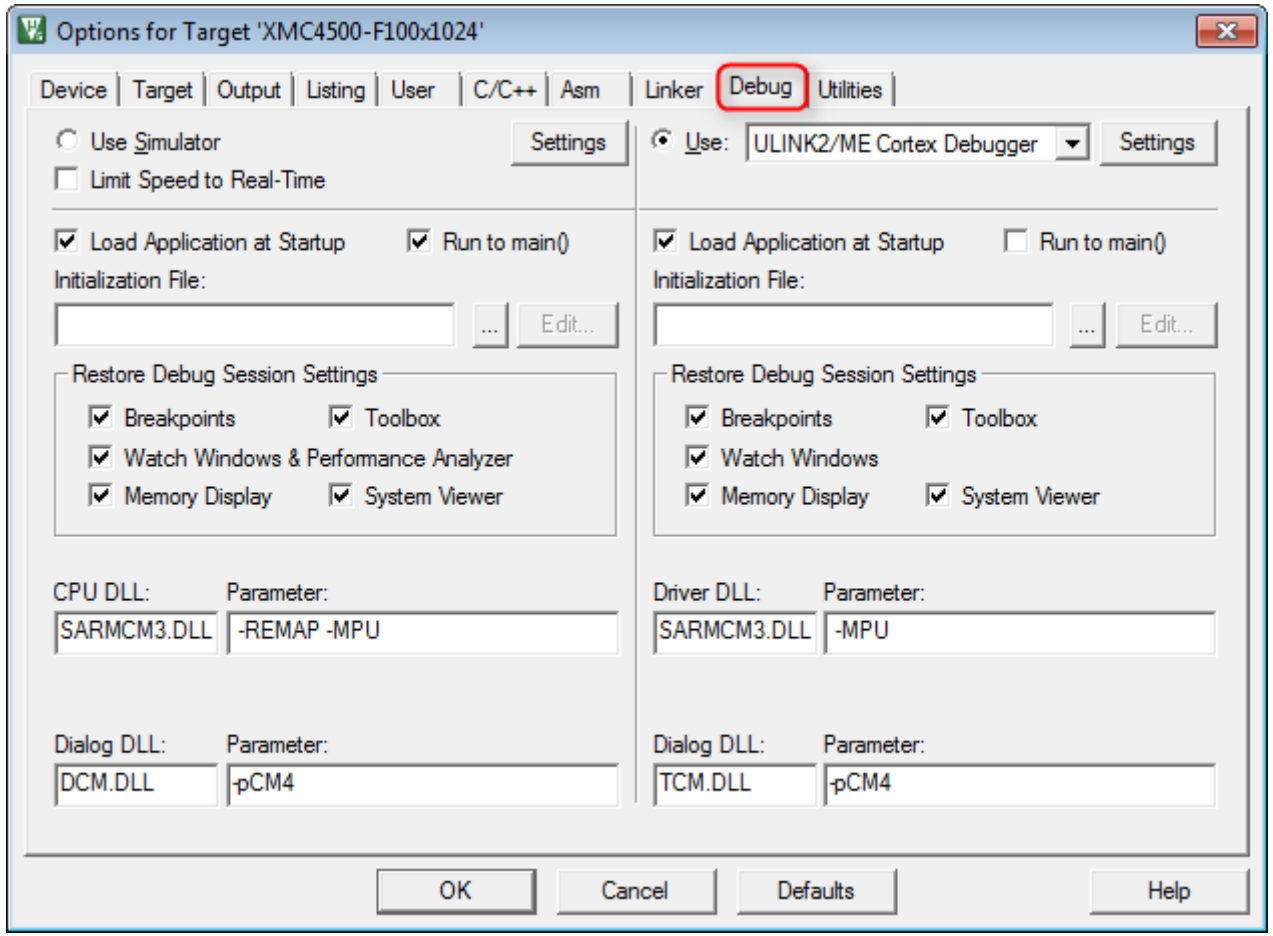
Going Back to DAVE™

- 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:

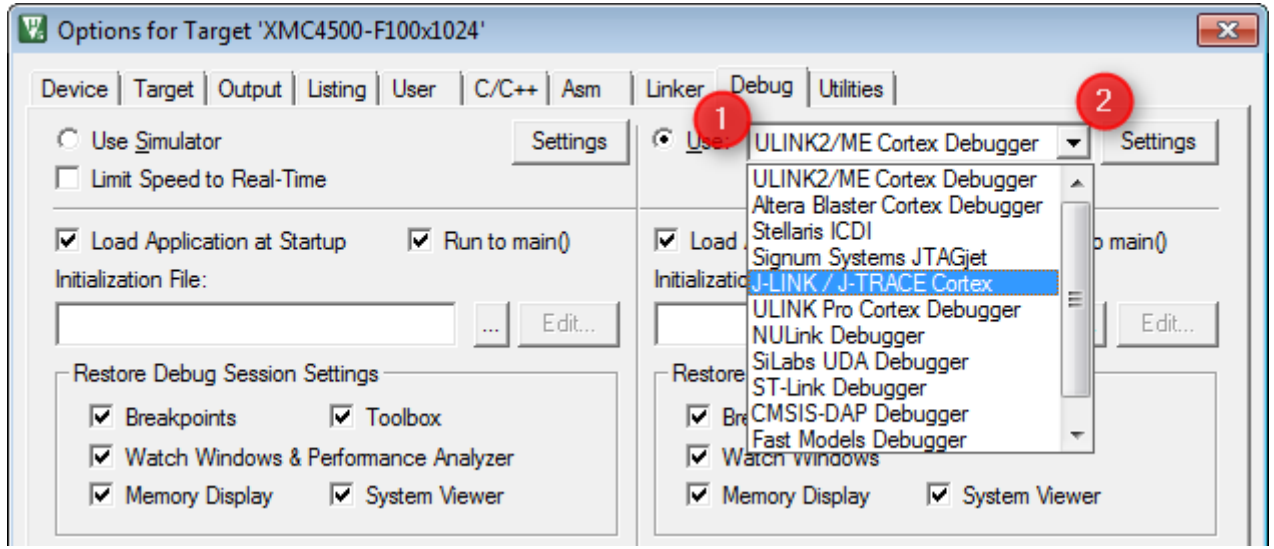


ARM

Select the J-LINK Debugger

- The XMC4500 Relax Kit has an integrated J-LINK debug adapter

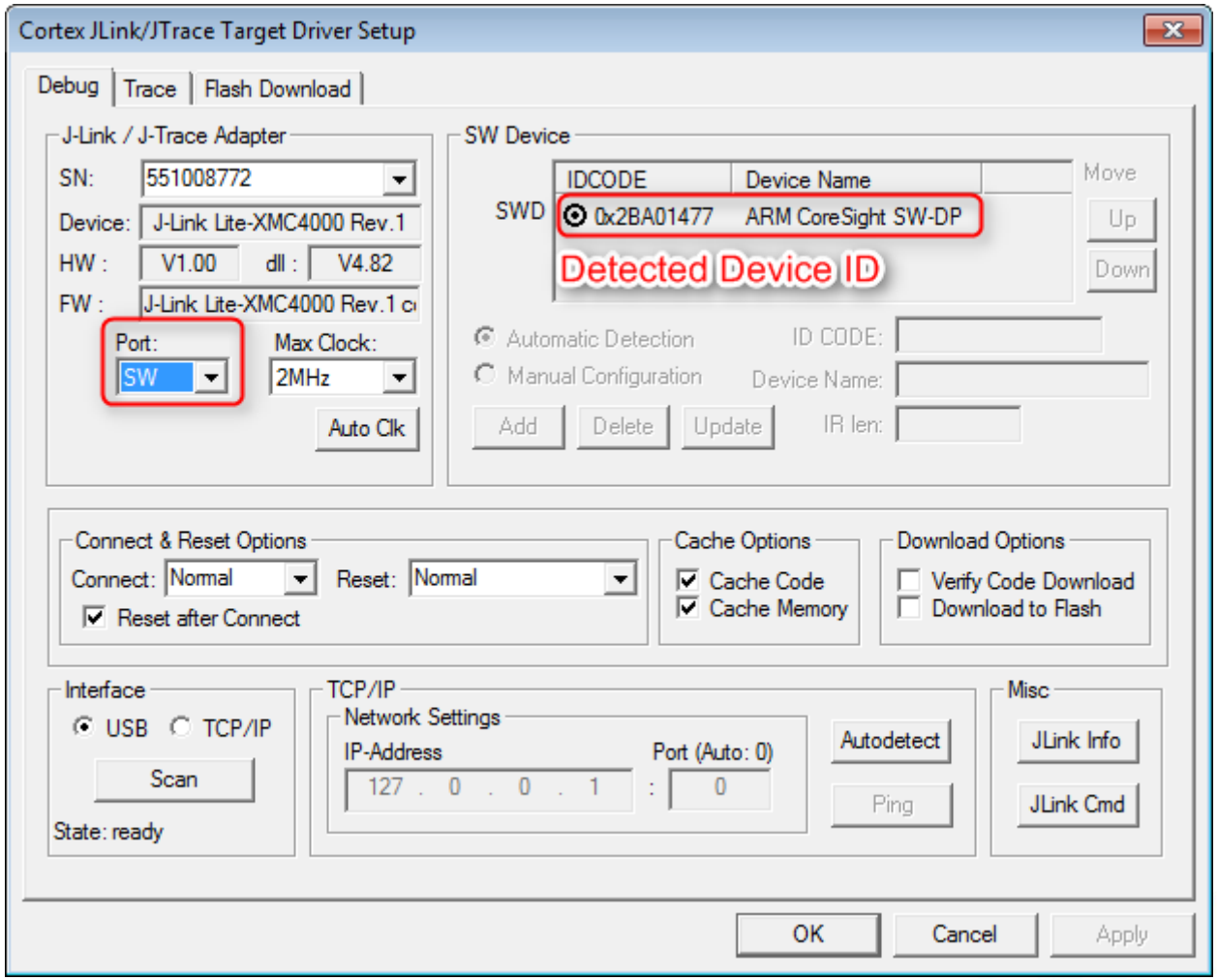
1. To change default adapter, click on **ULINK2/ME Cortex Debugger** and scroll down until **J-LINK/J-TRACE Cortex**
2. Click on **Settings**, to check the connectivity between the target and your PC



ARM

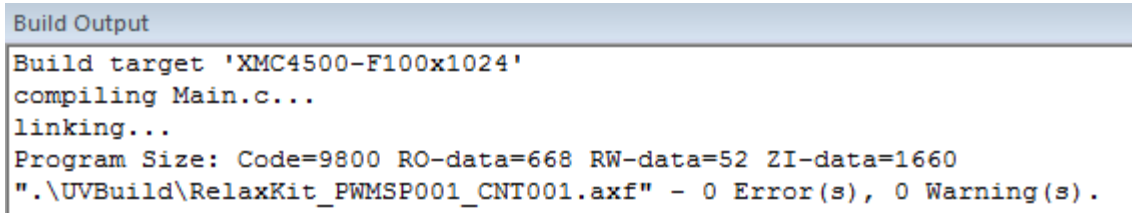
Enable the SW Port

- 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

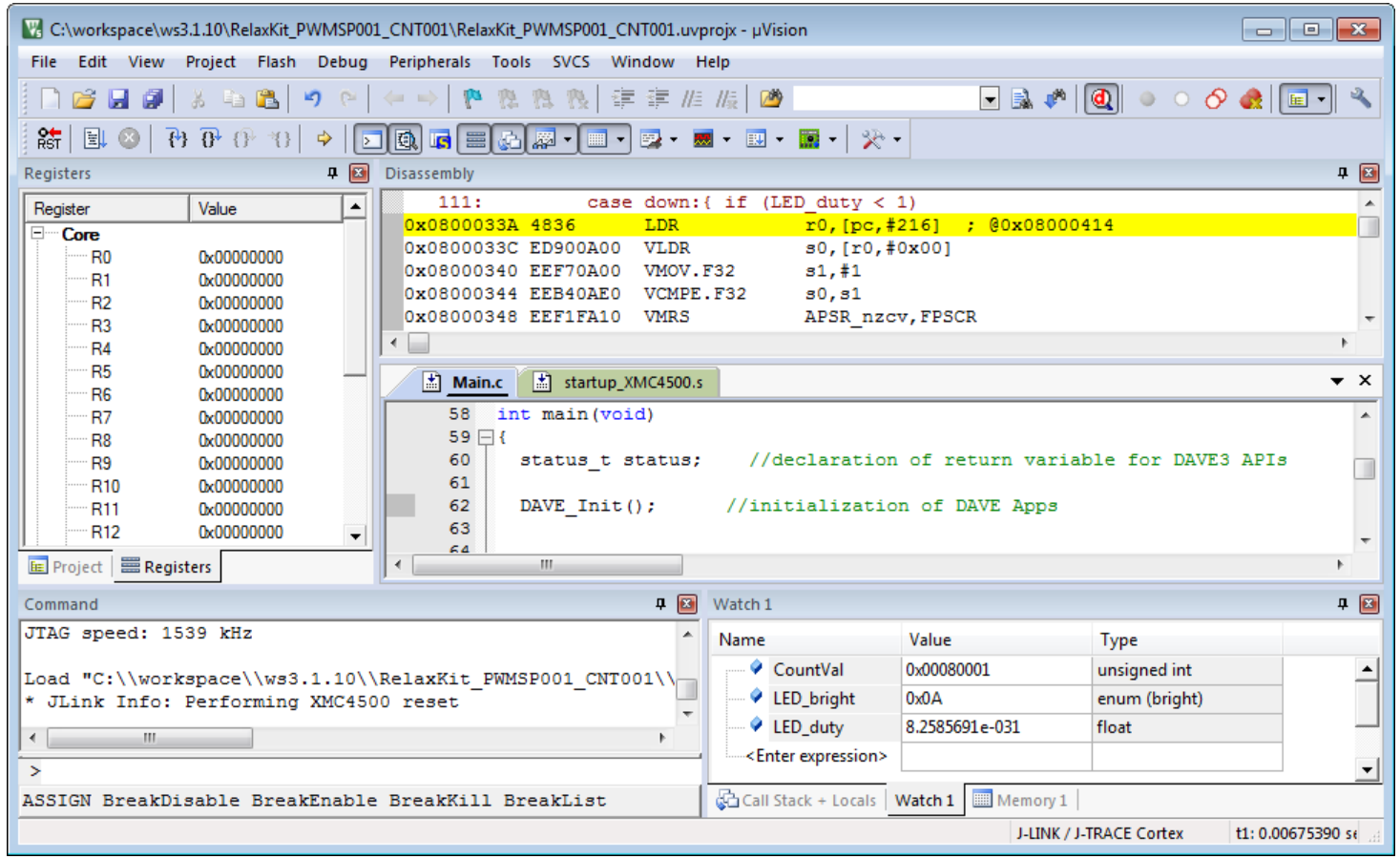


- Go to **Flash → Download** (or click on ) to download the program into the target's Flash memory

ARM

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. The LED connected to P1.1 will start flashing



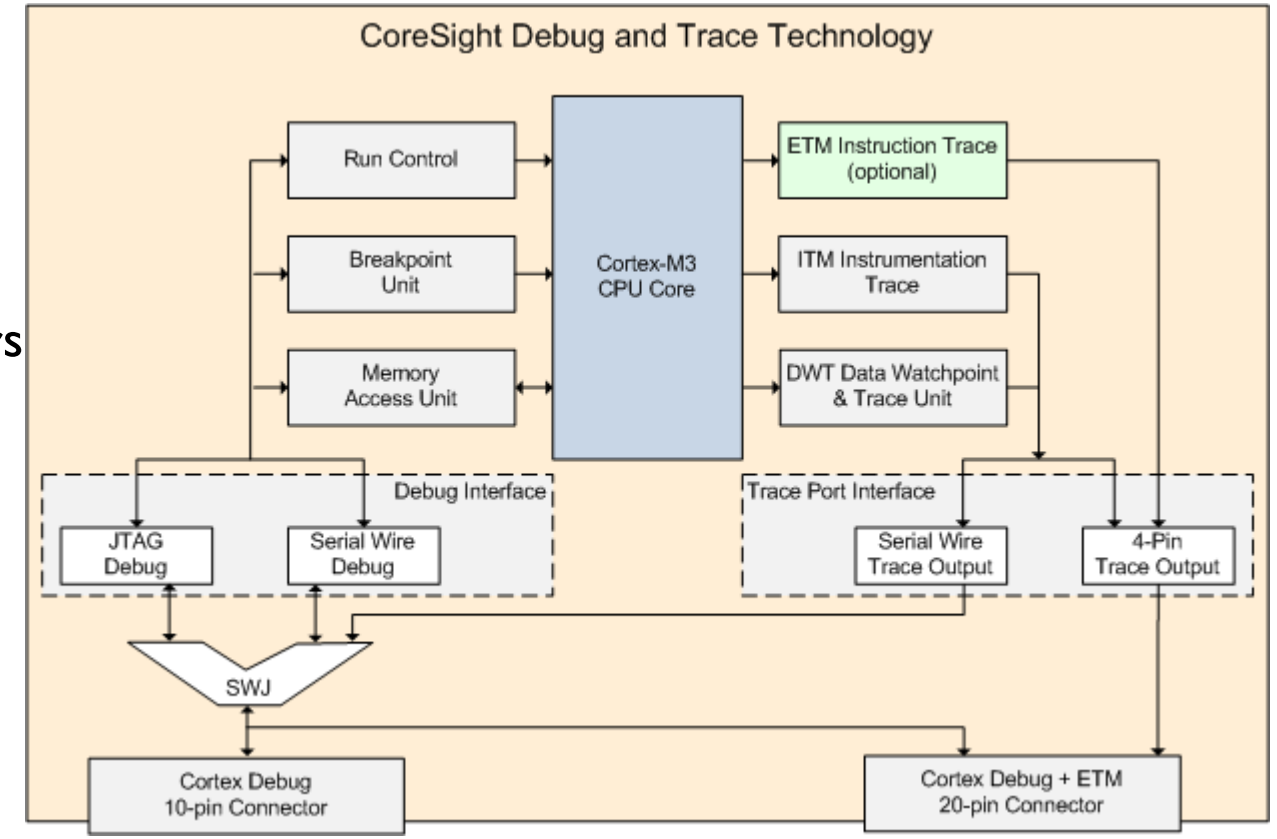
printf Debugging using the Instrumentation Trace Macrocell (ITM)

ARM

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



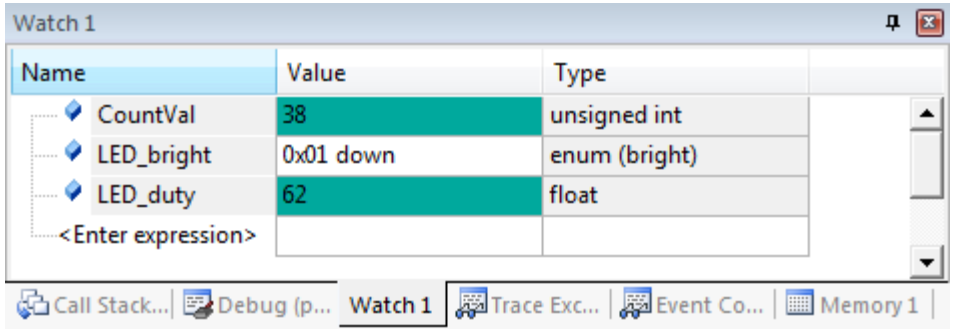
ARM

Serial Wire Viewer (SWV) – Real-Time Trace

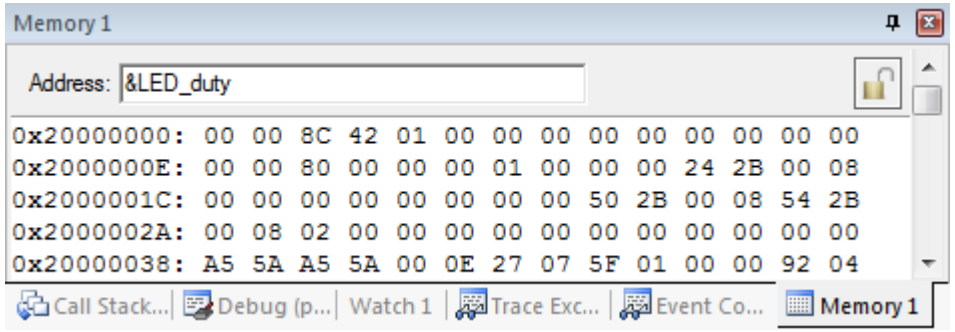
- The Serial Wire Viewer (SWV) provides real-time data trace information from various sources within the Cortex-M3/M4 device. This is output via the single SWO pin while your system processor continues running at full speed.
- Diagnostic system information is available from the ITM and DWT:
  - Data 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

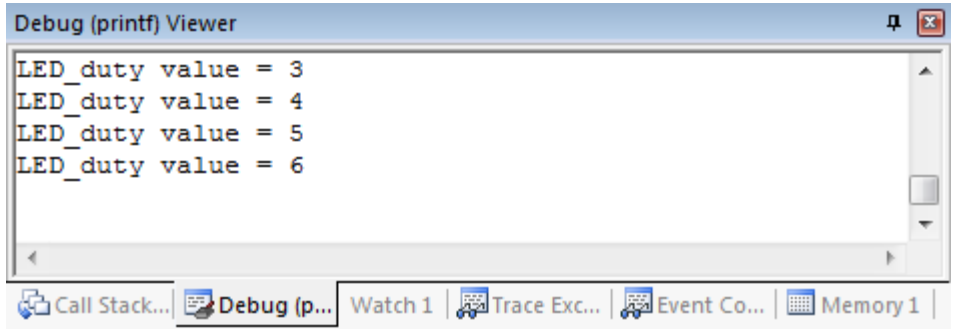
1. In the **Watch** window:



2. In the **Memory** window:



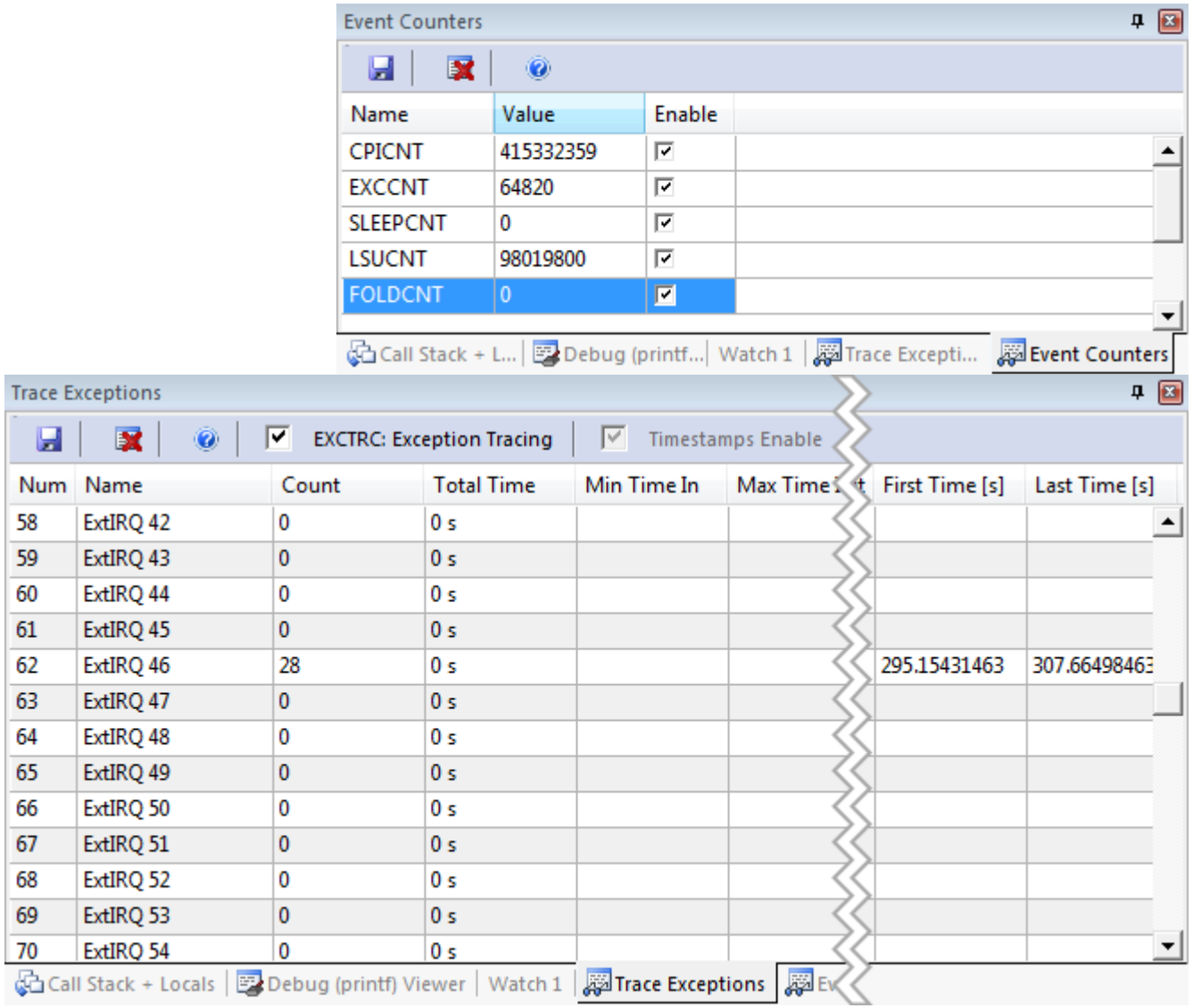
3. In the **Debug (printf) Viewer** window:



ARM

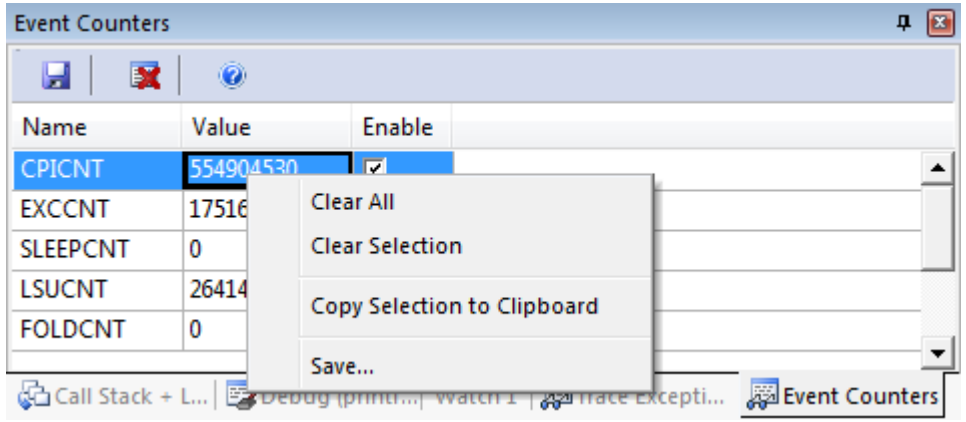
SWV Trace Windows

- 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



Event Counters

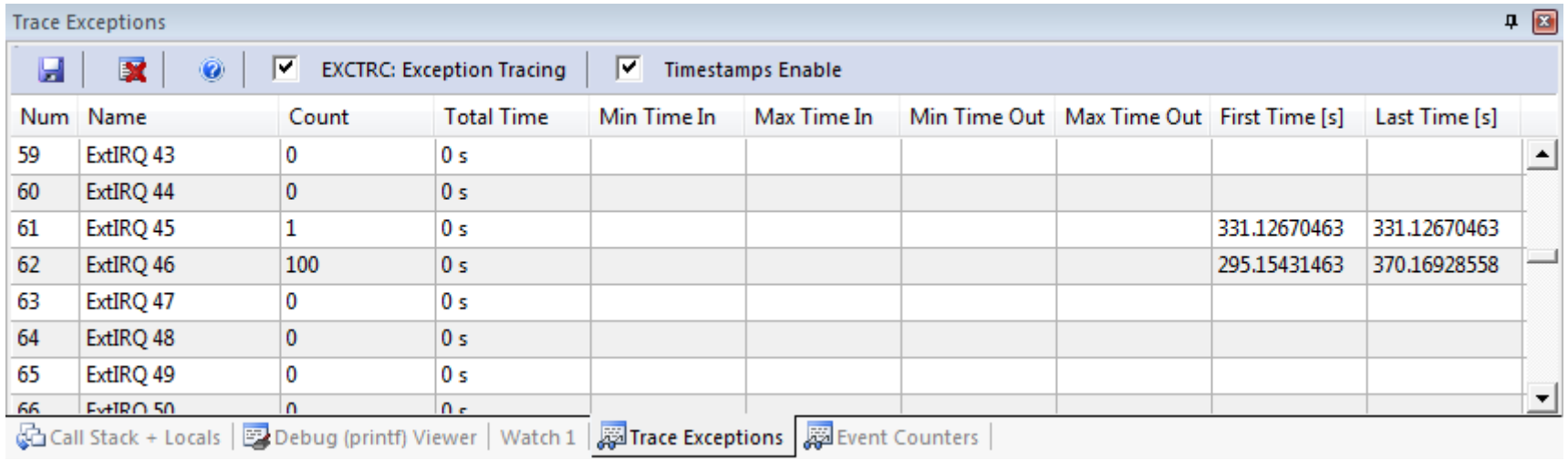
- Displays real-time values of the specific event counters
- Shows the number of times counter ‘rolls over’.
- Updated in real time
- Ability to Save Counter Data
- Right Click and select “Clear All” or “Clear Selection”



ARM

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

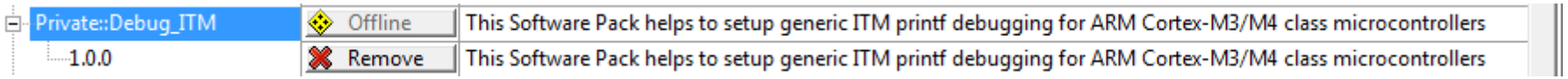


ARM

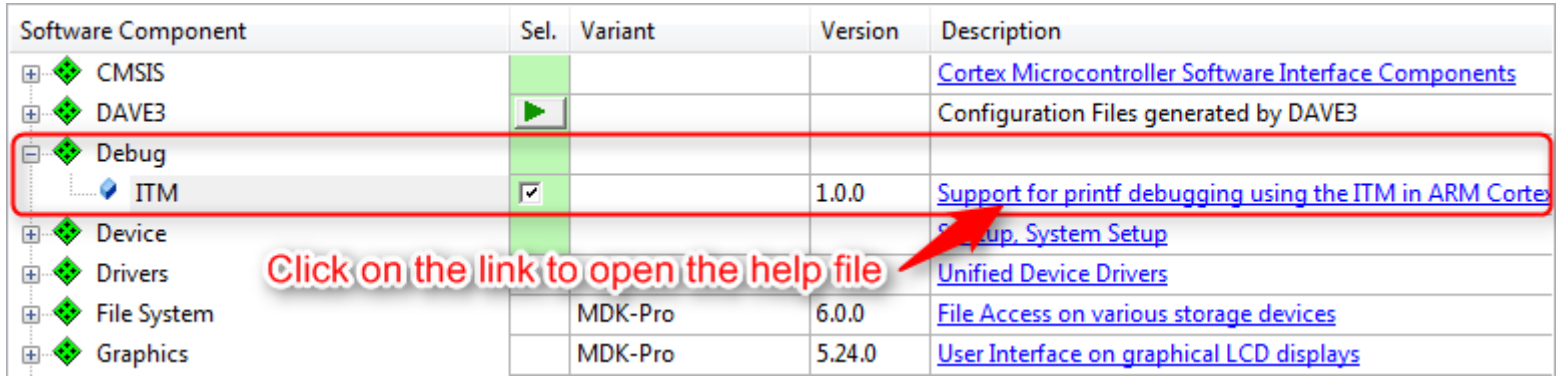
ITM printf Debugging

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:

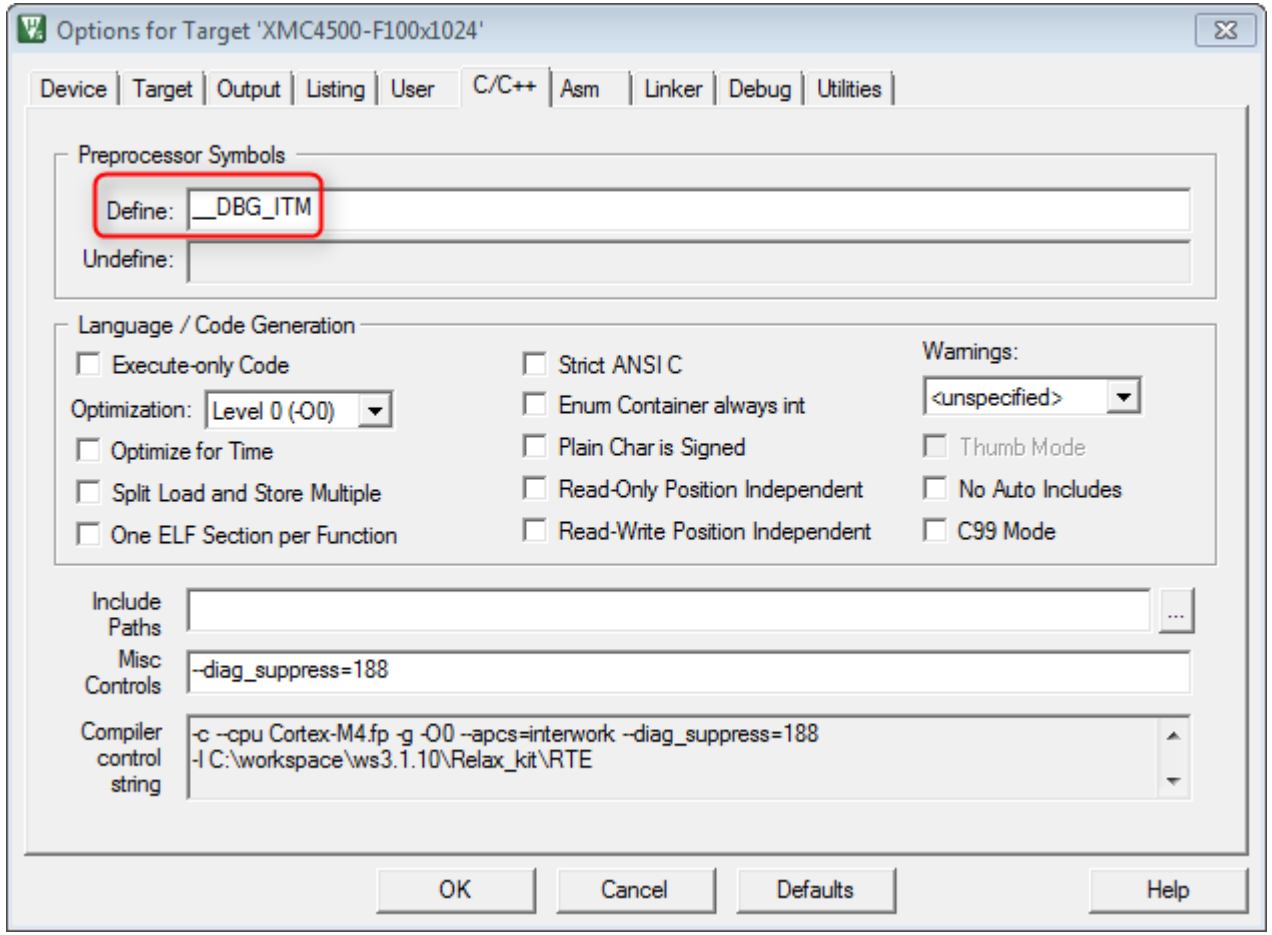


- Open the Manage Run-Time Environment window in µVision and browse to the Debug software component:



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



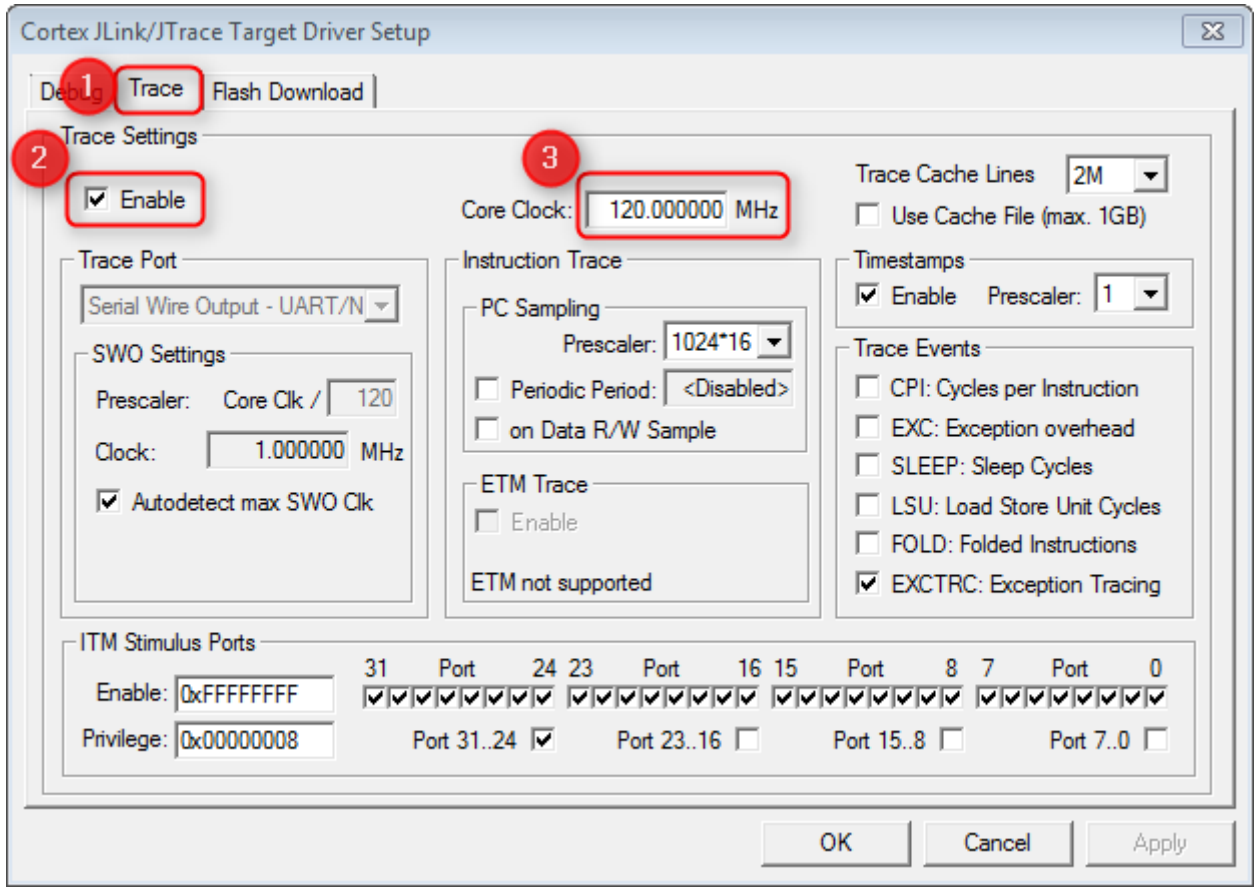
ARM

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



ARM

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);
```

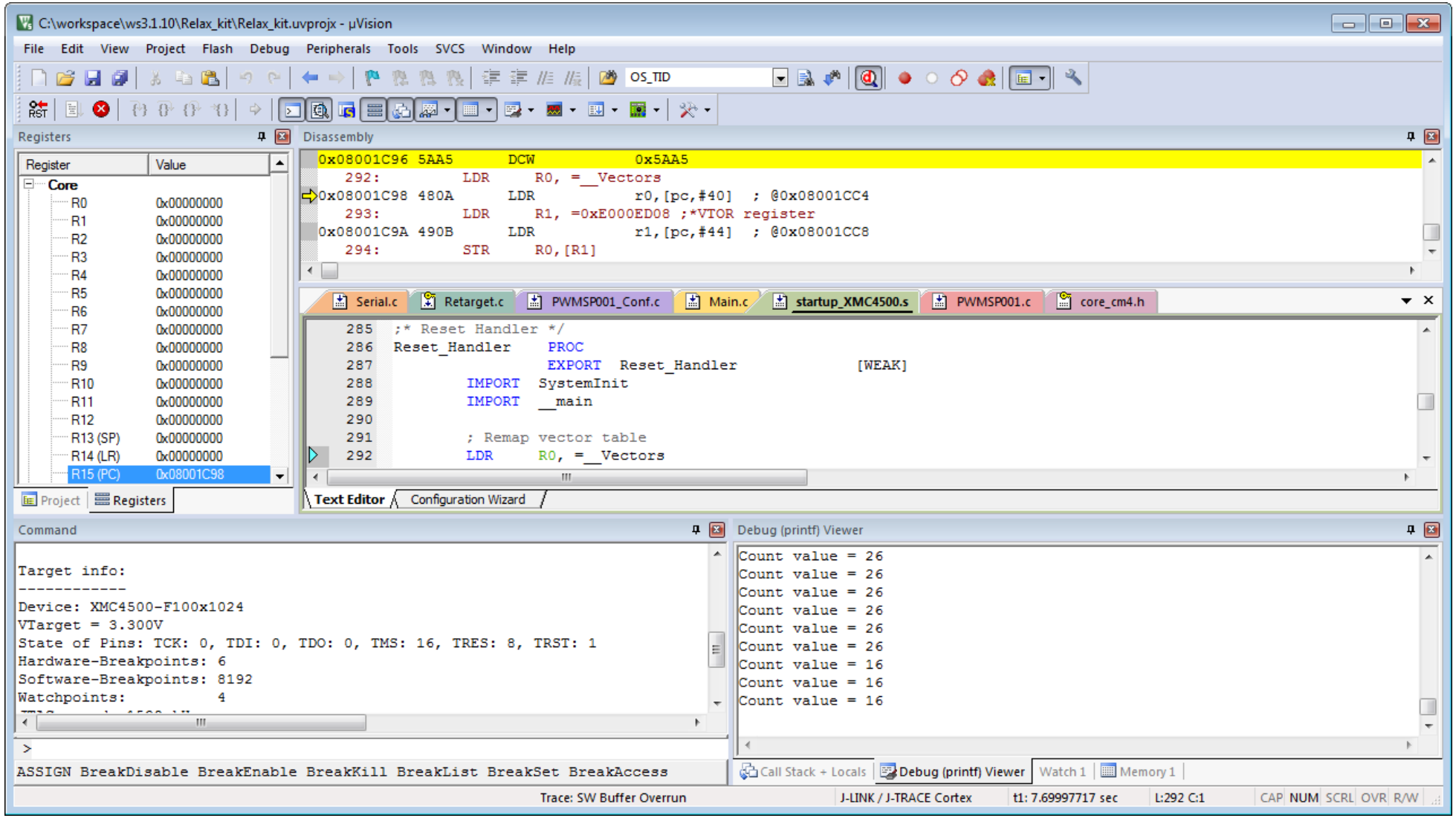
Rebuild the Project and Download to Target

- 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 µVision debugger.
- Go to **View → Serial Windows → Debug (printf)** Viewer to open that window that shows the printf output
- Run the program (press F5)

ARM

Debug (printf) Viewer



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

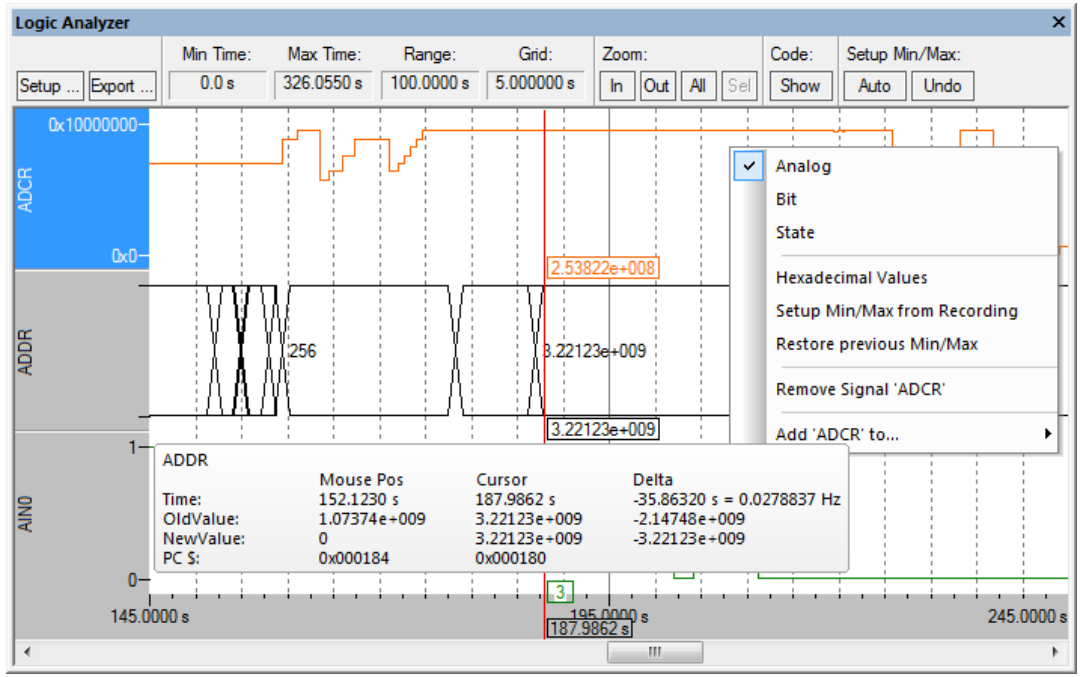
JTAG	SWD	SWO 100 Mpbs	ETM Streaming

ARM

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



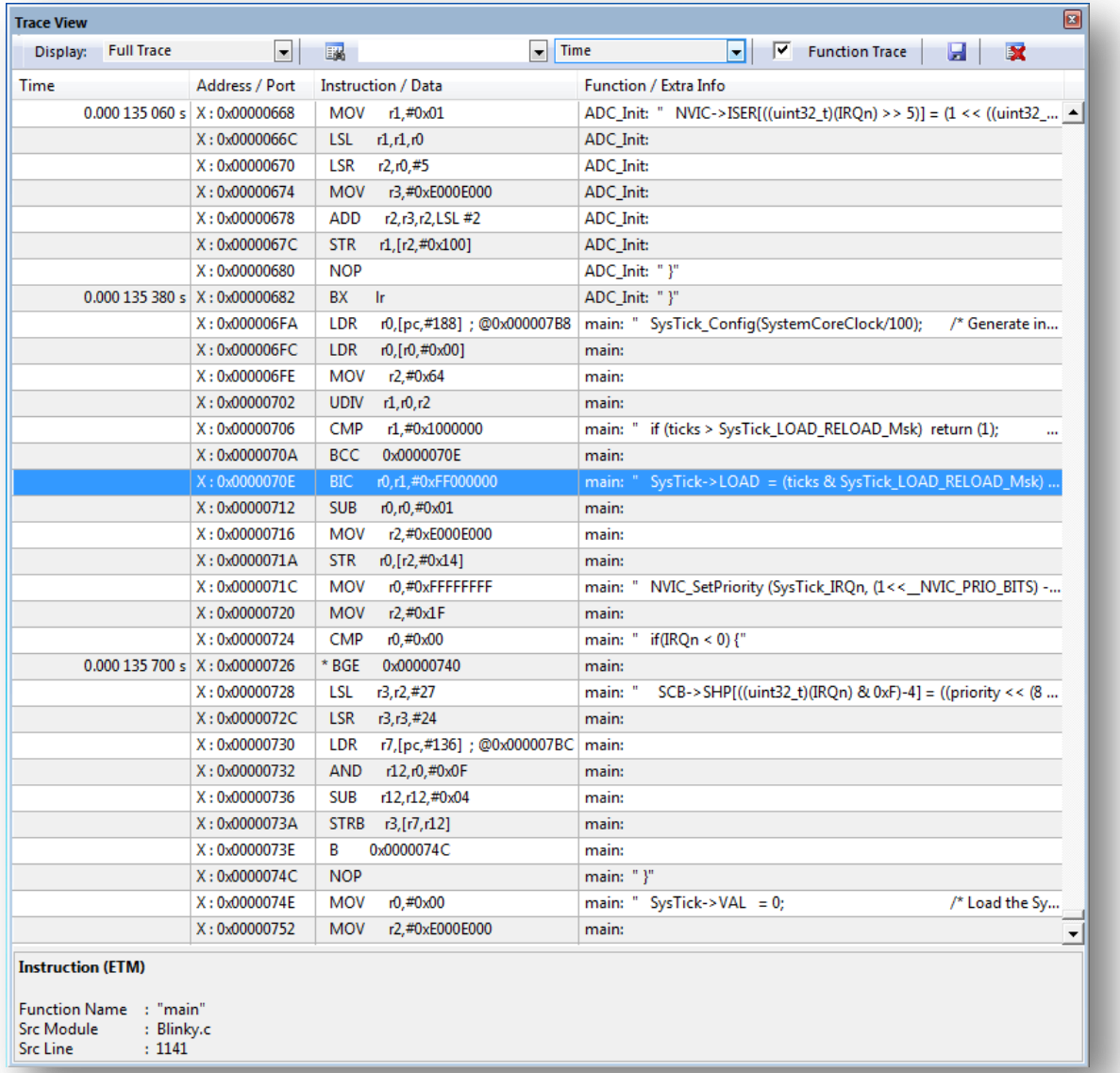
JTAG | SWD | SWO 100 Mpbs | ETM Streaming

ARM

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 Mpbs | ETM Streaming

ARM

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

ARM

Creating Applications using Software Packs

ARM

Objective: Blinky Project with RTOS

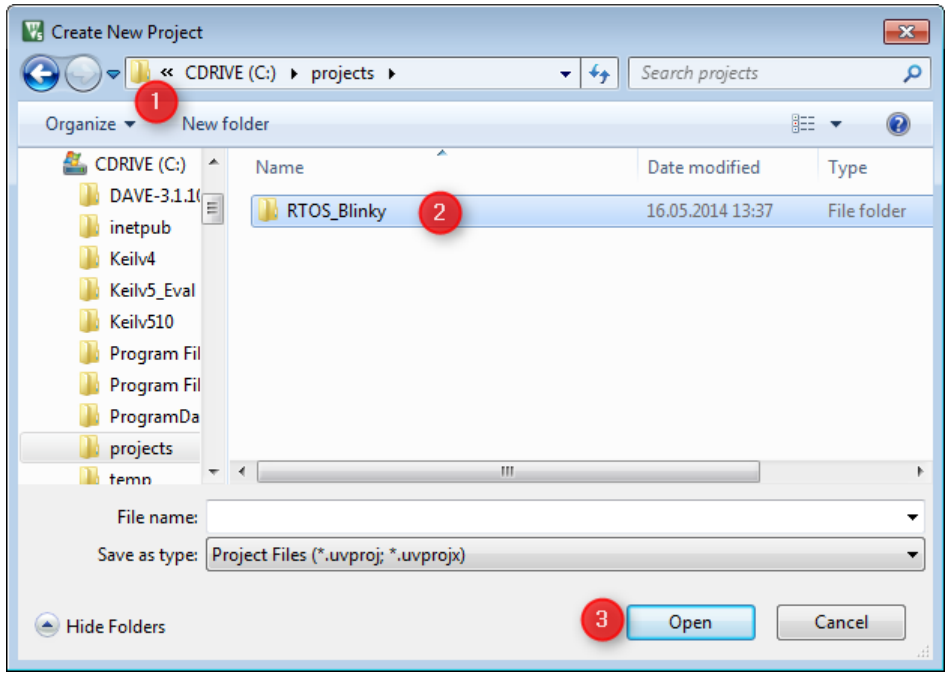
- We will create a new project from scratch, using:
  - CMSIS-CORE
  - Startup Files for CMSIS
  - CMSIS-RTOS compliant Keil RTX

- Later, we will add a USB driver and Middleware to set and reset the LEDs from a PC

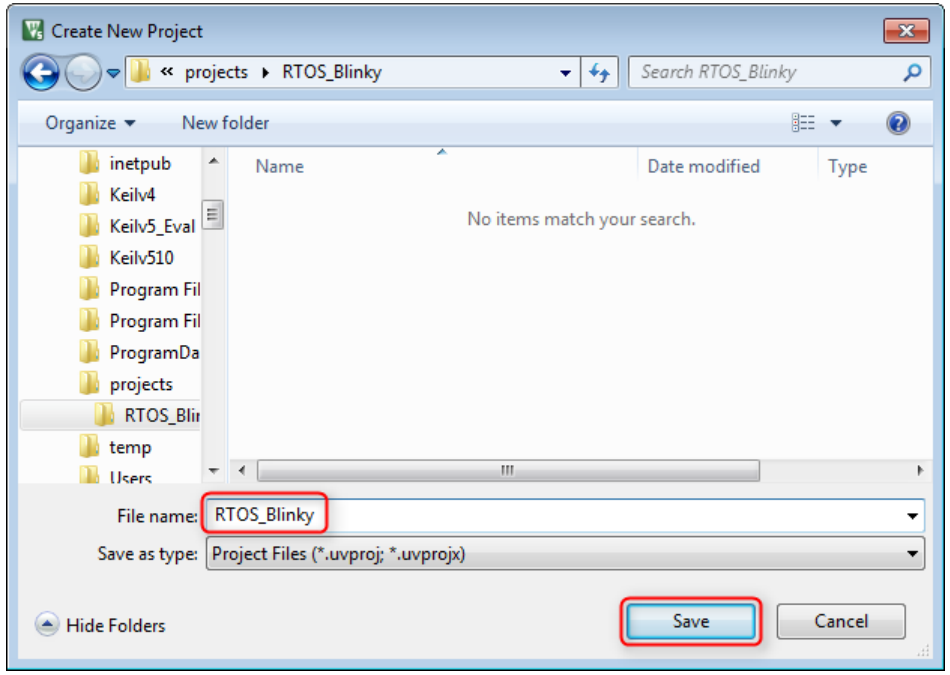
Create a New Project

- Open µVision and click **Project → New µVision Project…**

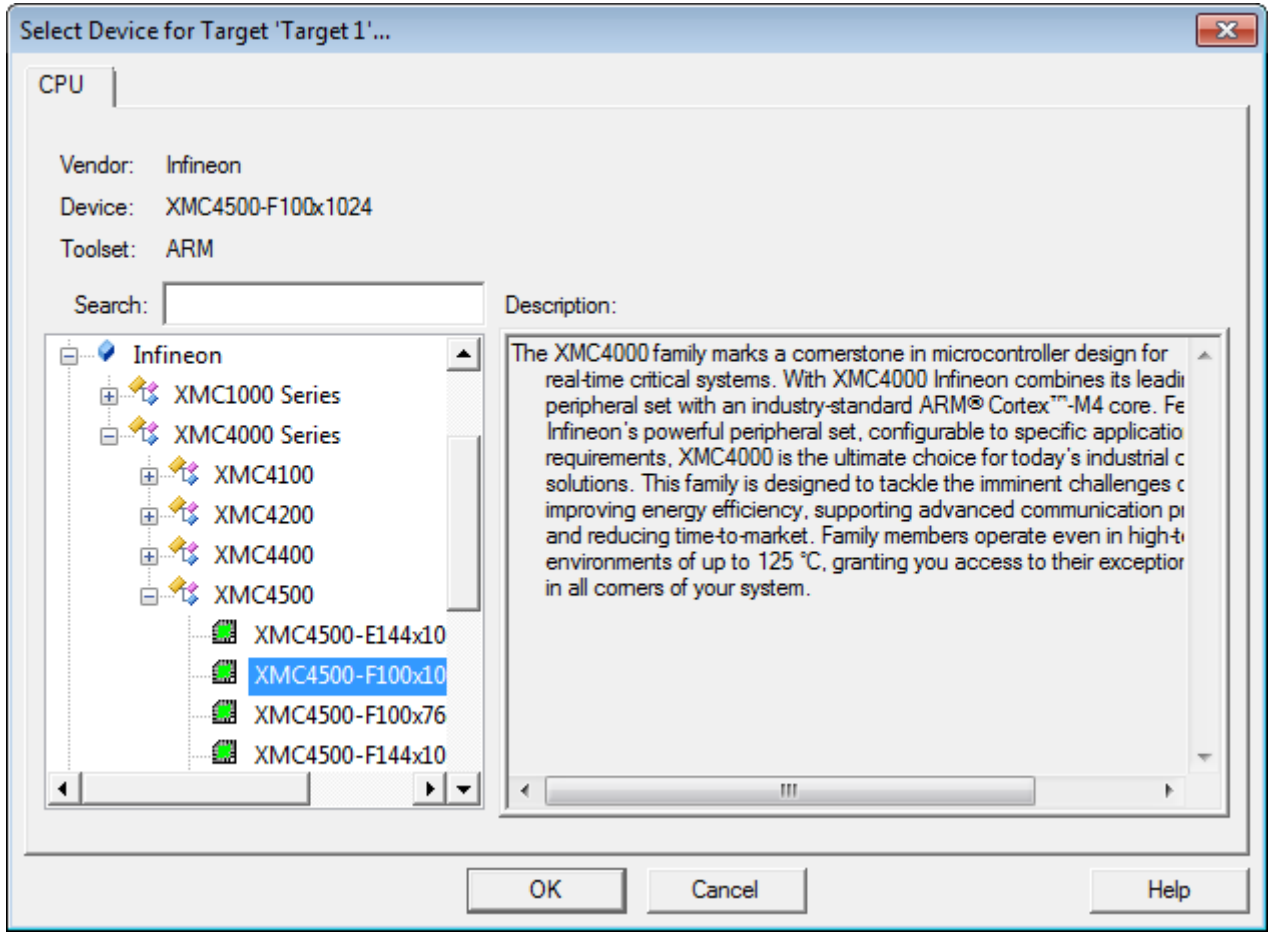
1. Click **New folder**
2. Enter Name: **RTOS_Blinky**
3. Click **Open**



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



Select Device for Target: XMC4500-F100x1024

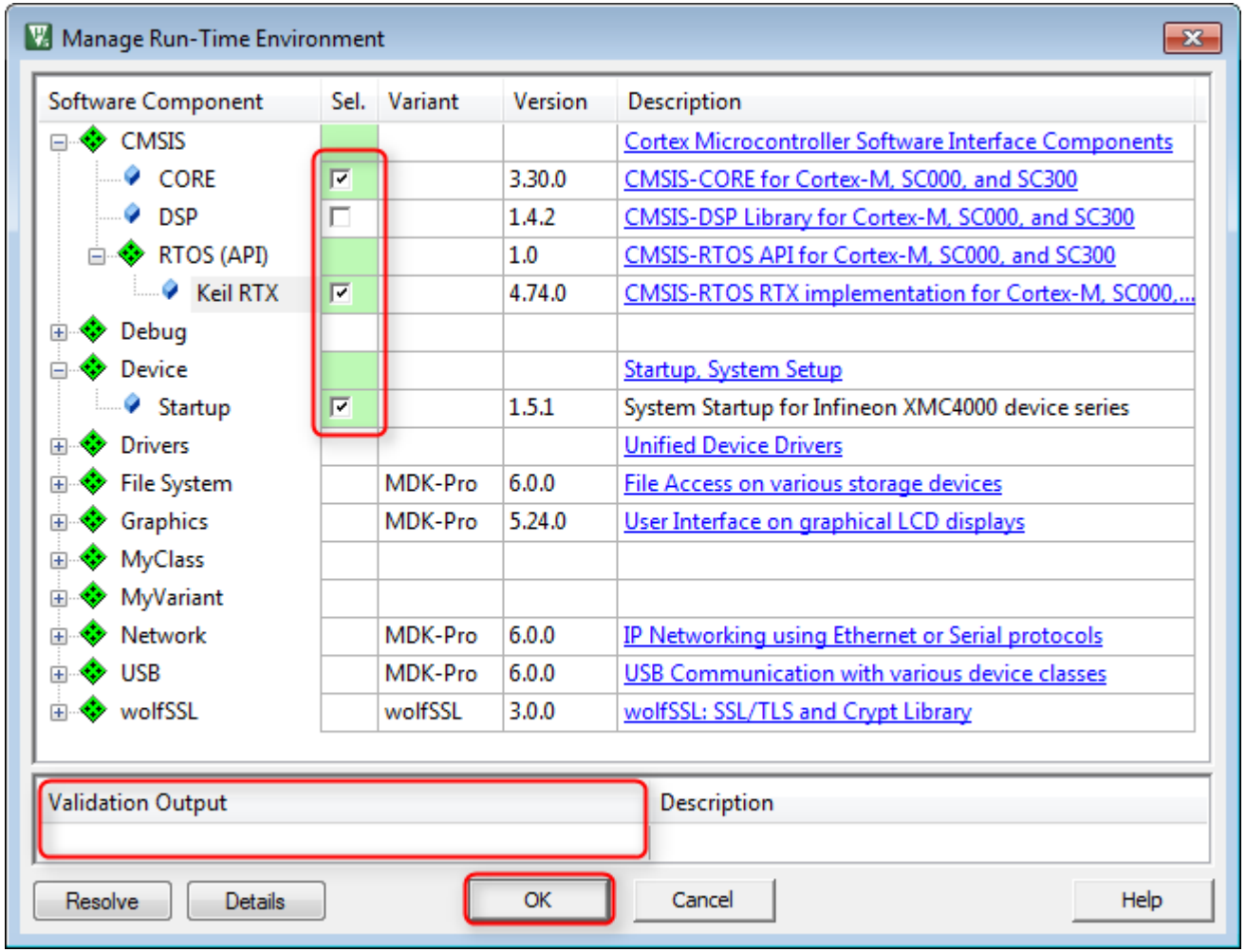


ARM

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**

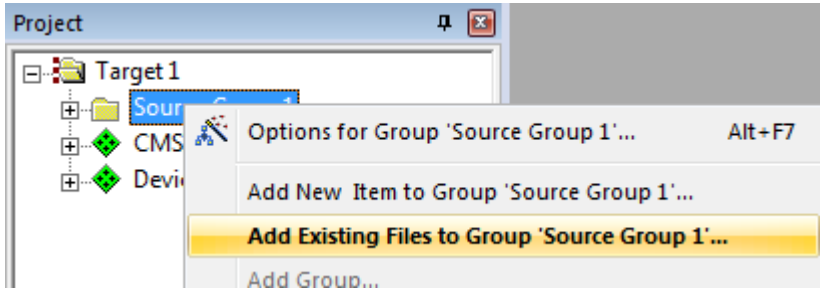


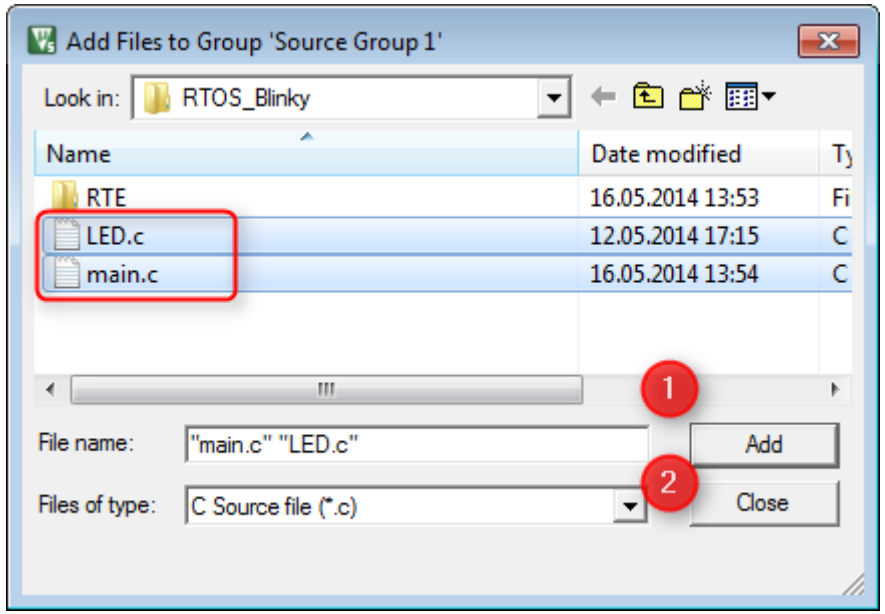
ARM

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**

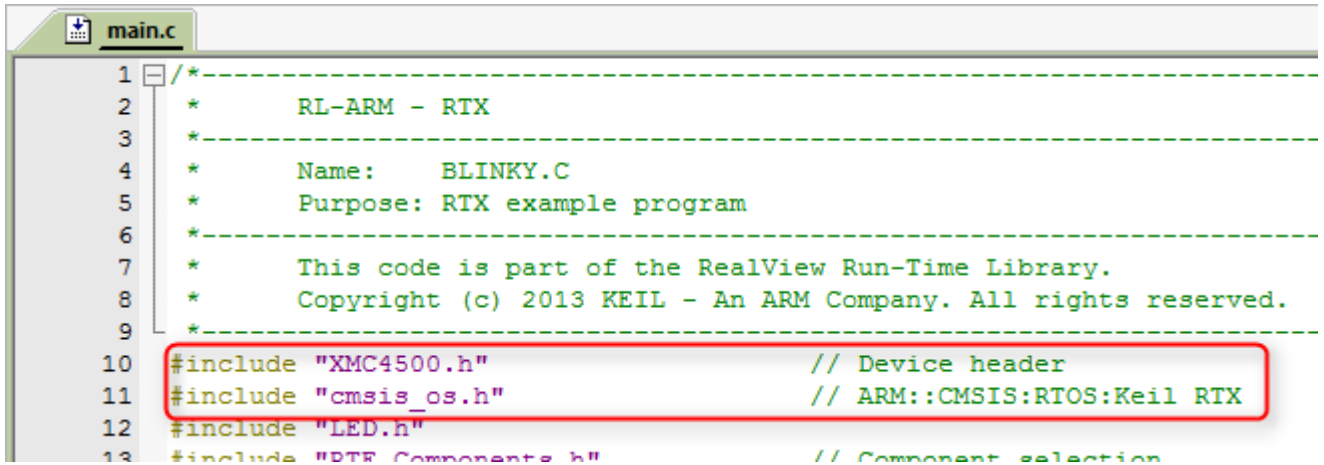




ARM

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**

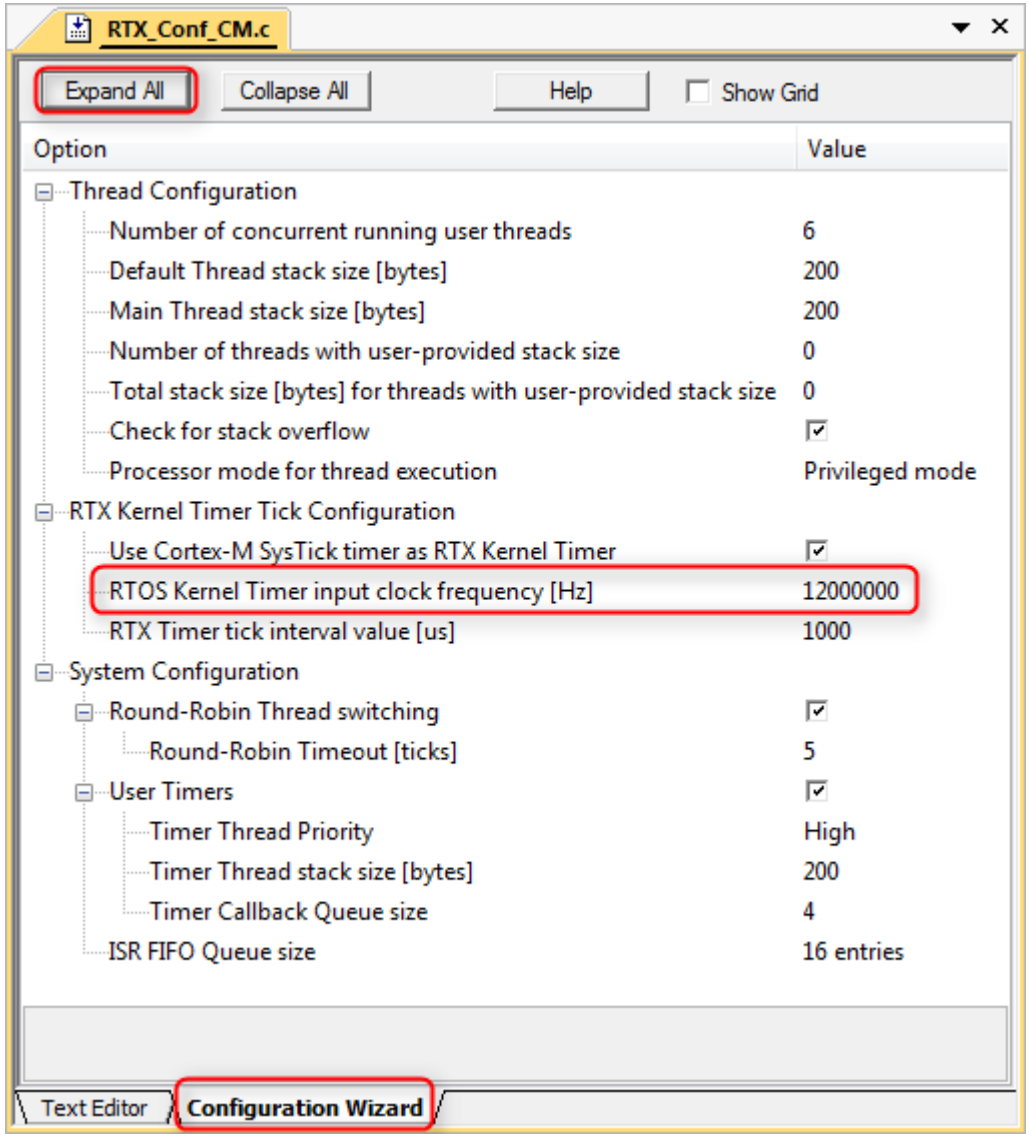


- Errors and warnings should disappear
- Press **CTRL+S** to save the file

ARM

Required RTOS Settings

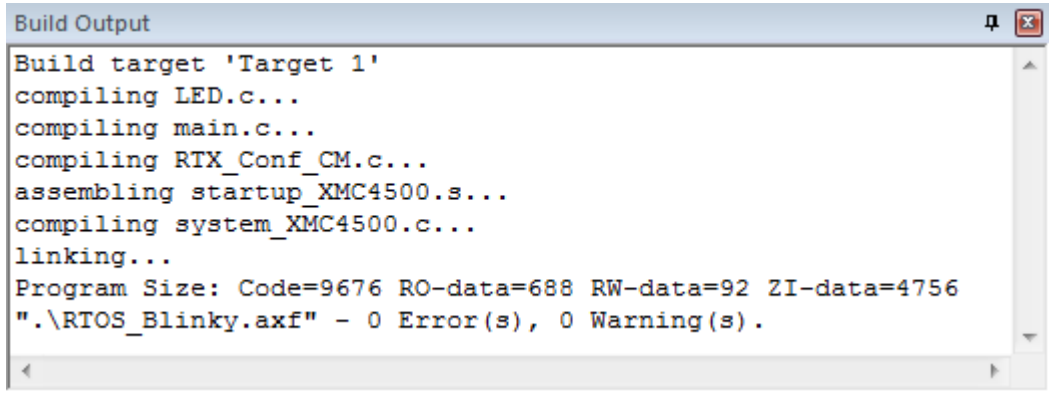
- Double-click **RTX_Conf_CM.c**. The file opens in the editor.
- Click **Configuration Wizard** tab (at the bottom of the file)
- Graphical representation of the configuration files appears. Click **Expand All**
- All available settings for Keil RTX are displayed
- We need to change the RTOS Kernel 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



ARM

Build the Project

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

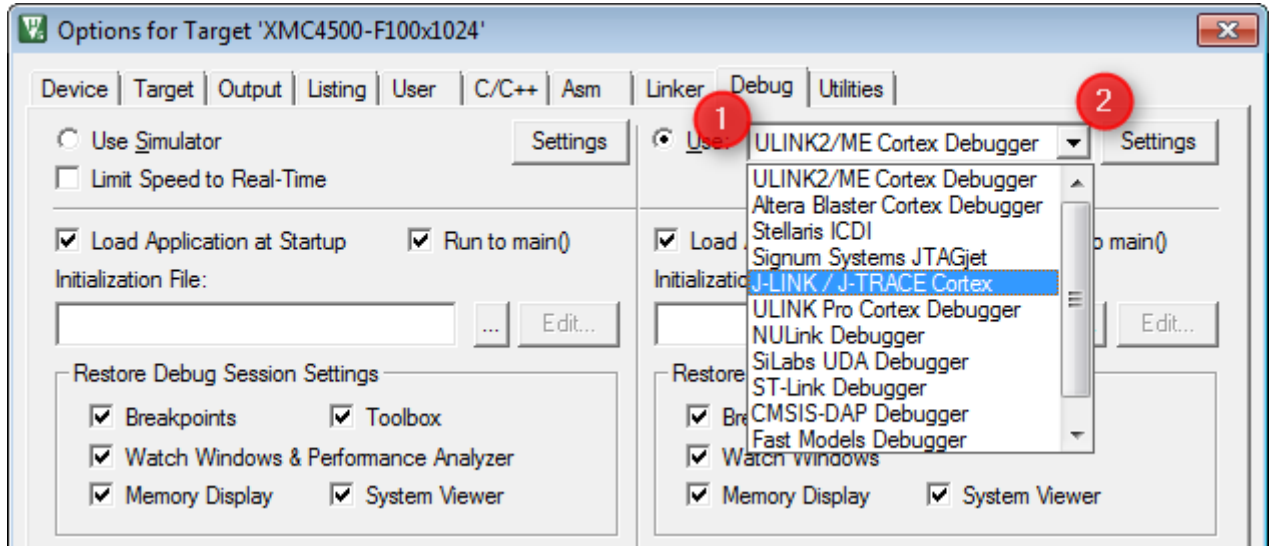


ARM

Select the J-LINK Debugger

- The XMC4500 Relax Kit has an integrated J-LINK debug adapter

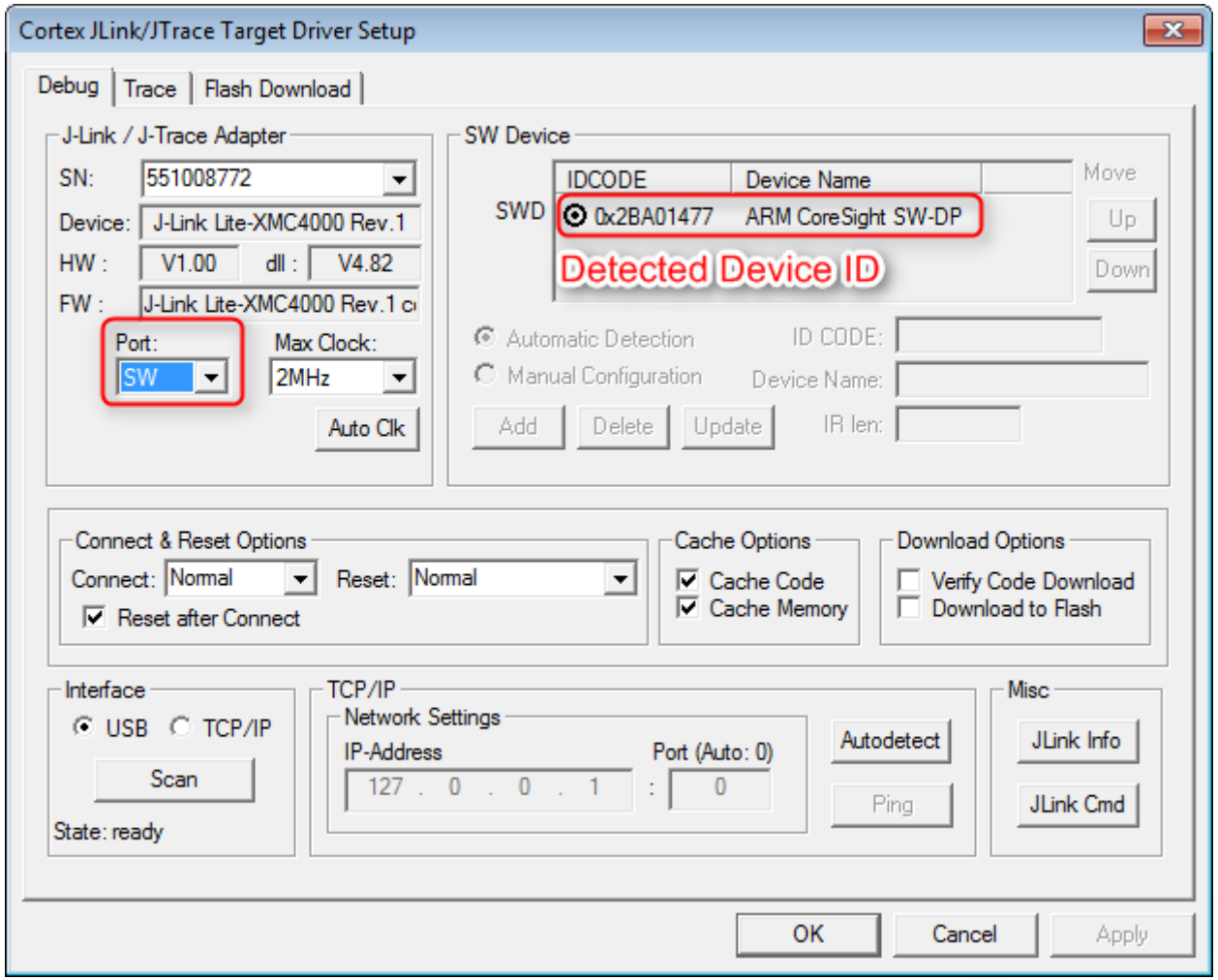
1. To change default adapter, click on **ULINK2/ME Cortex Debugger** and scroll down until **J-LINK/J-TRACE Cortex**
2. Click on **Settings**, to check the connectivity between the target and your PC



ARM

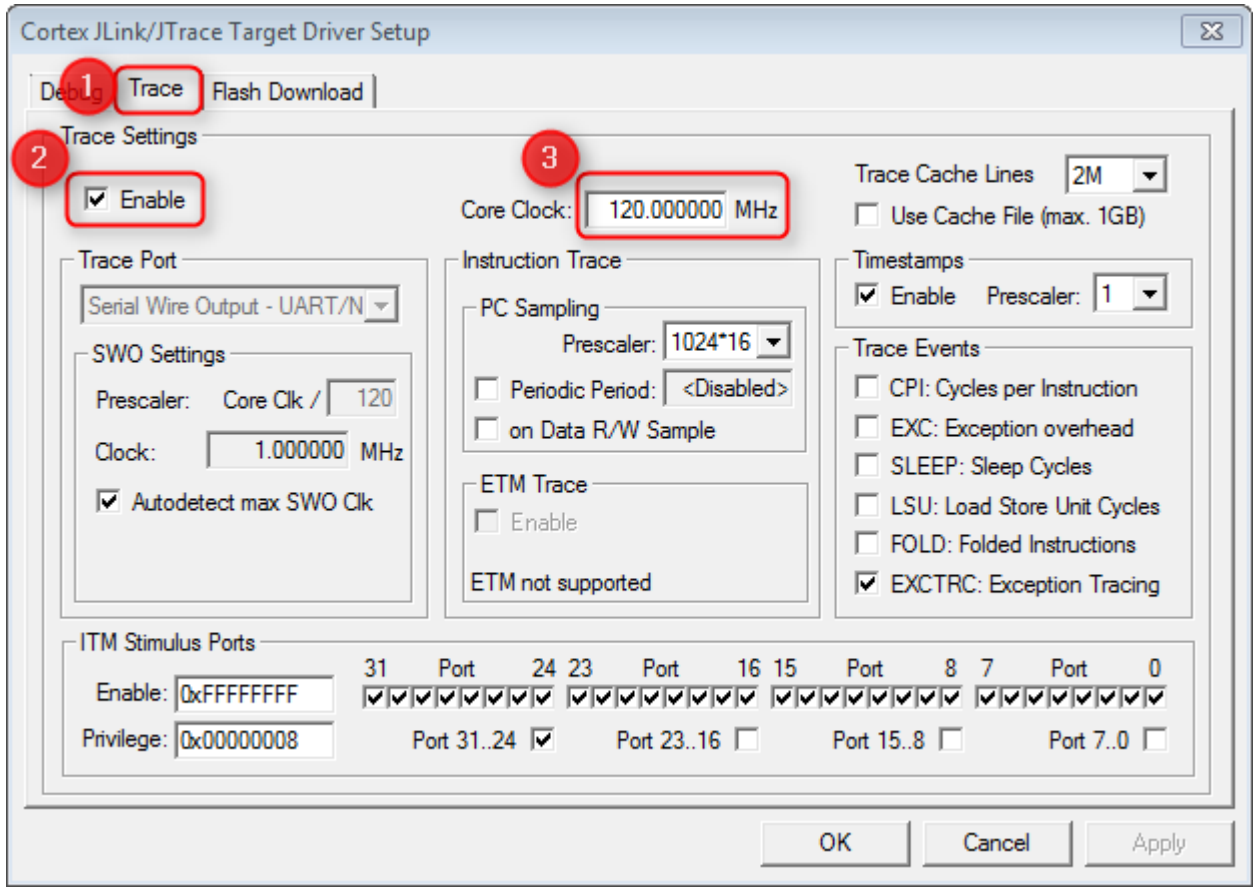
Enable the SW Port

- 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 identified automatically



Enabling Trace in J-LINK

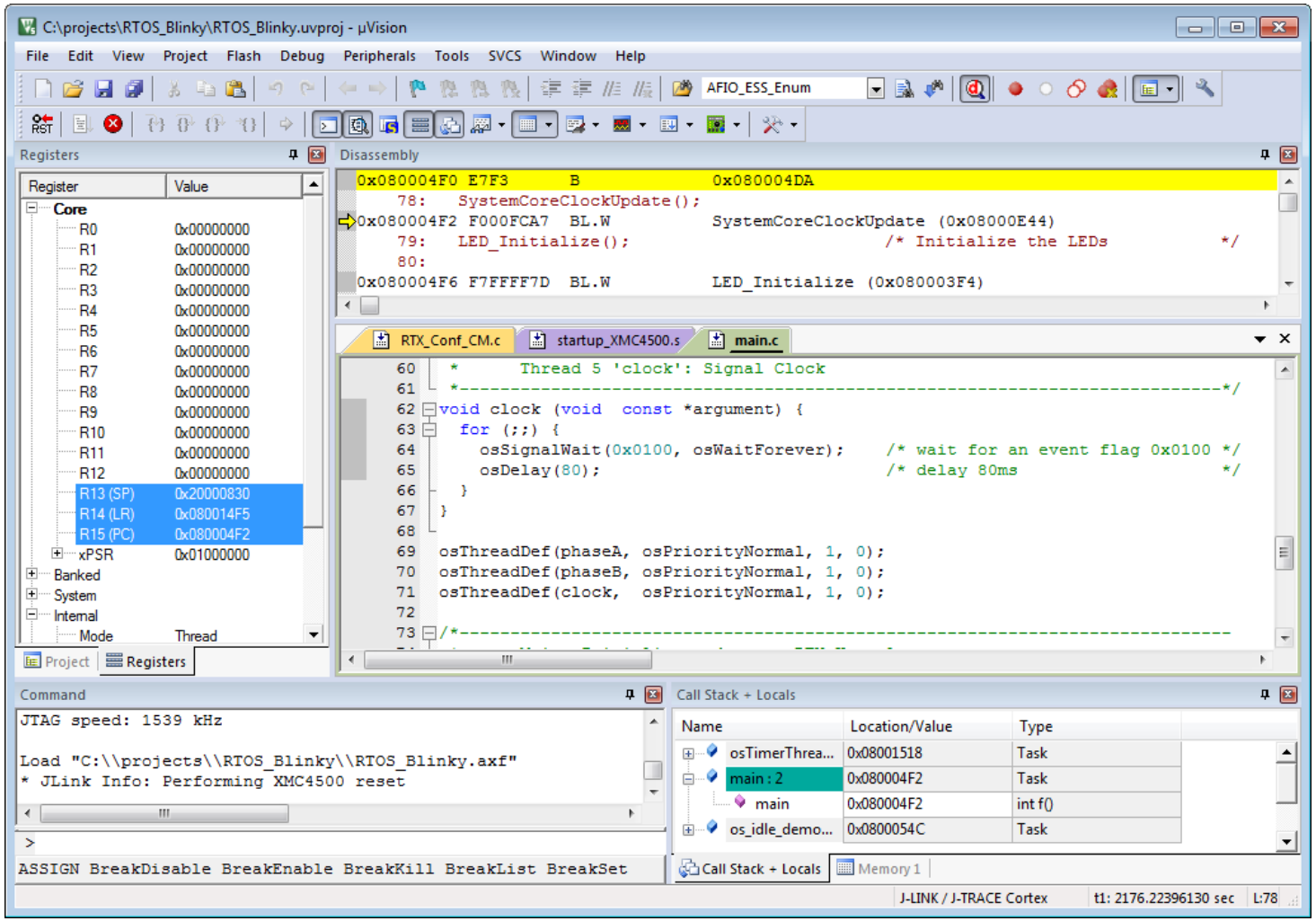
1. Click on the **Trace** tab
2. Check **Enable**
3. Enter the correct Core Clock: **120** MHz
4. Click **OK** twice



ARM

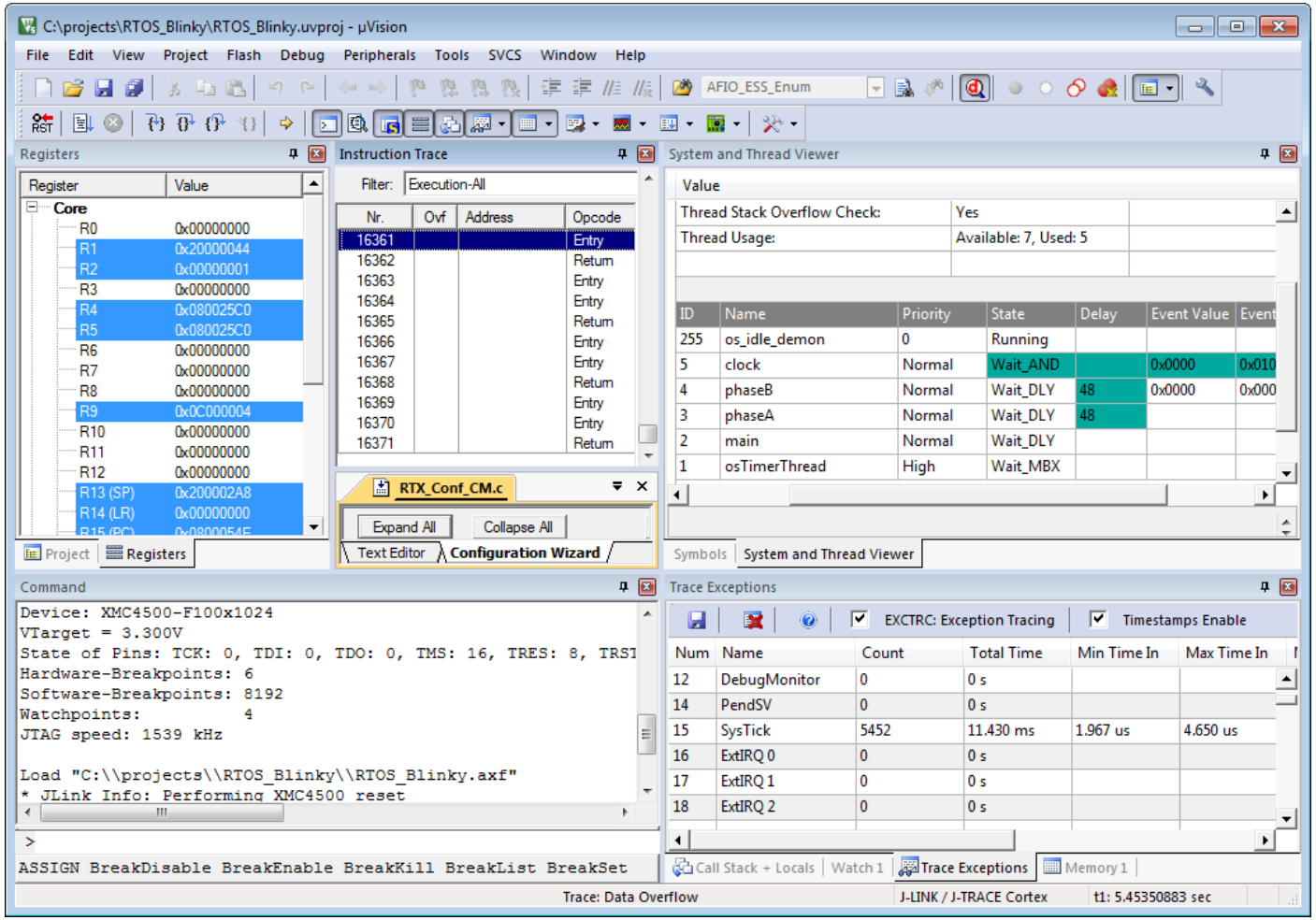
Start a Debug Session

- 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



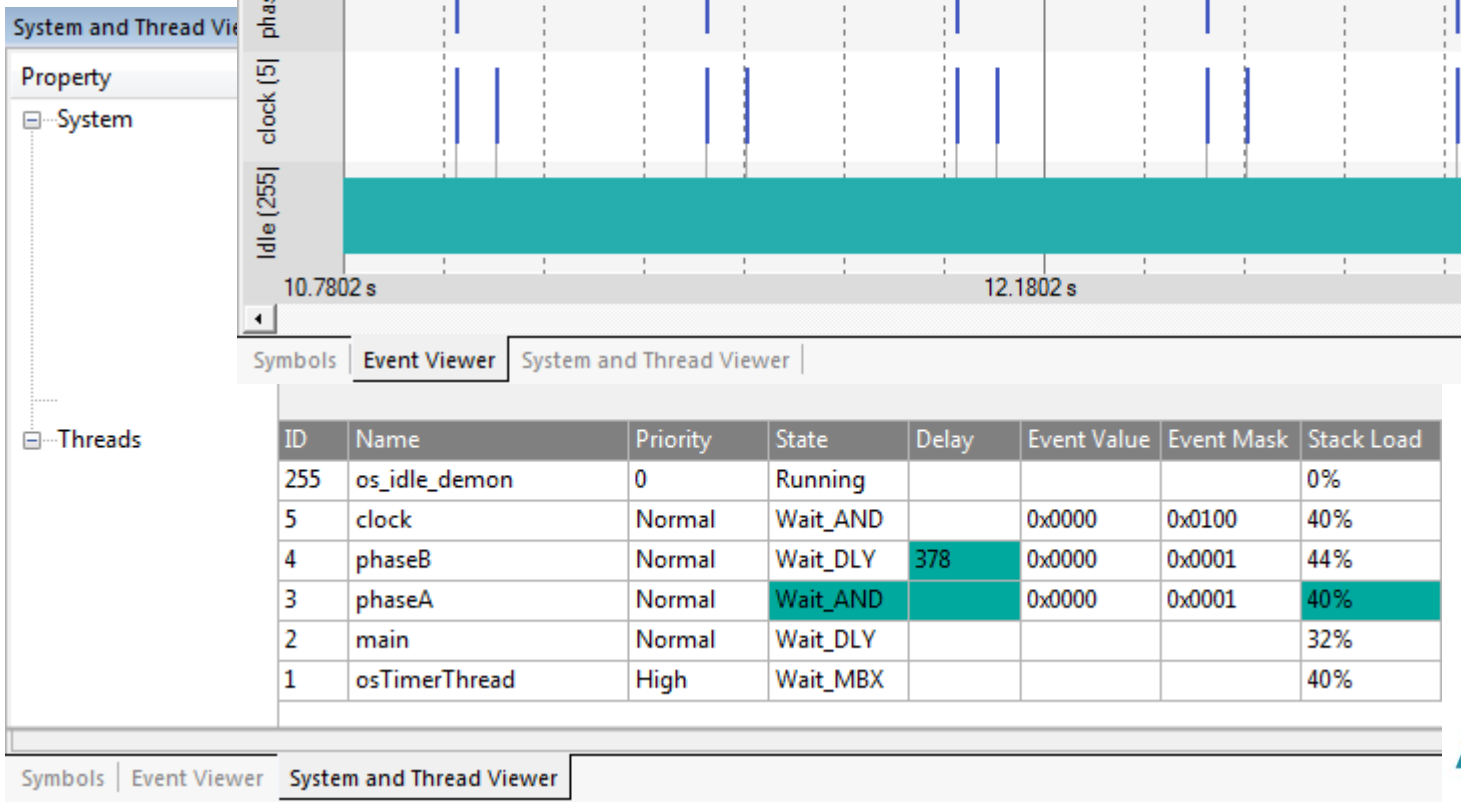
Add more Debug Information

- 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 → PORT1**



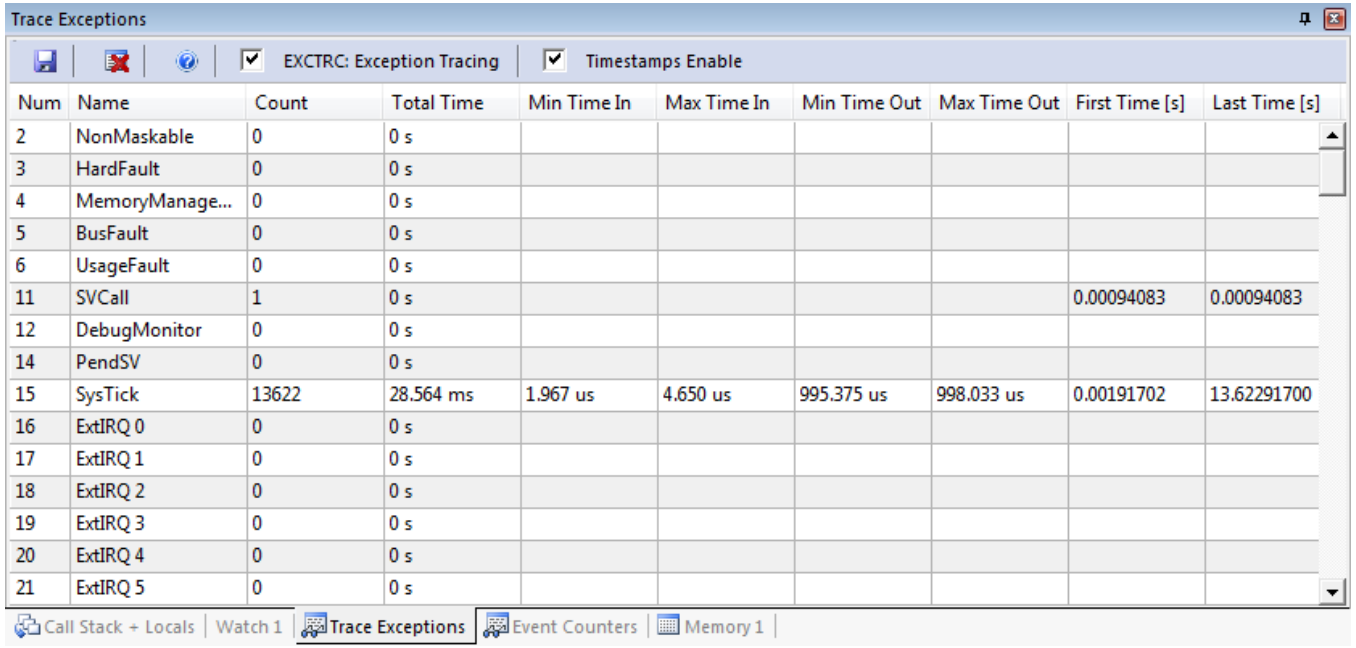
Kernel-Aware Debugging

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



Trace Exceptions

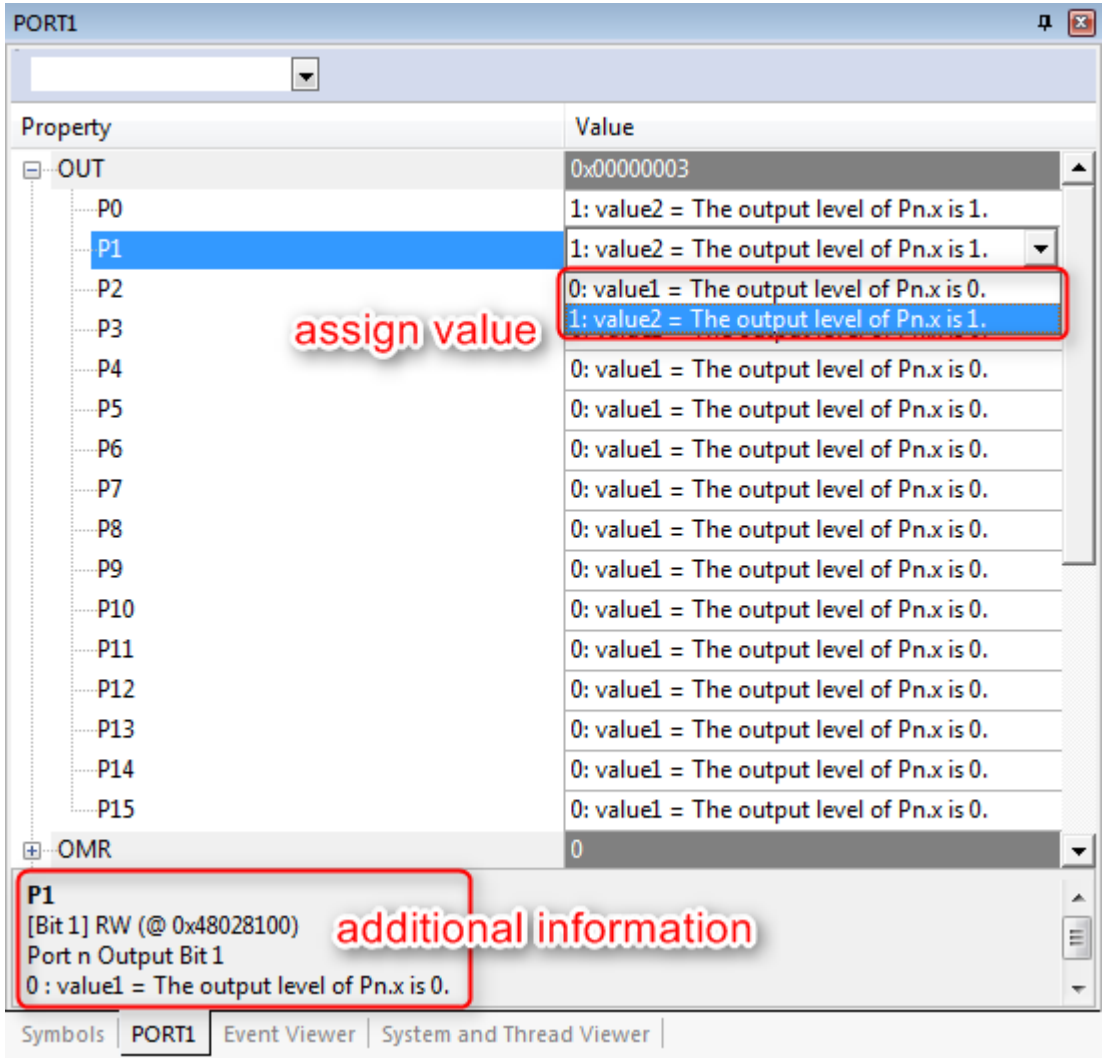
- Displays statistical information about program exceptions
- Exception name and number, number of times entered
- Max and min time spent in and out of exceptions
- First and last time entered



ARM

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)

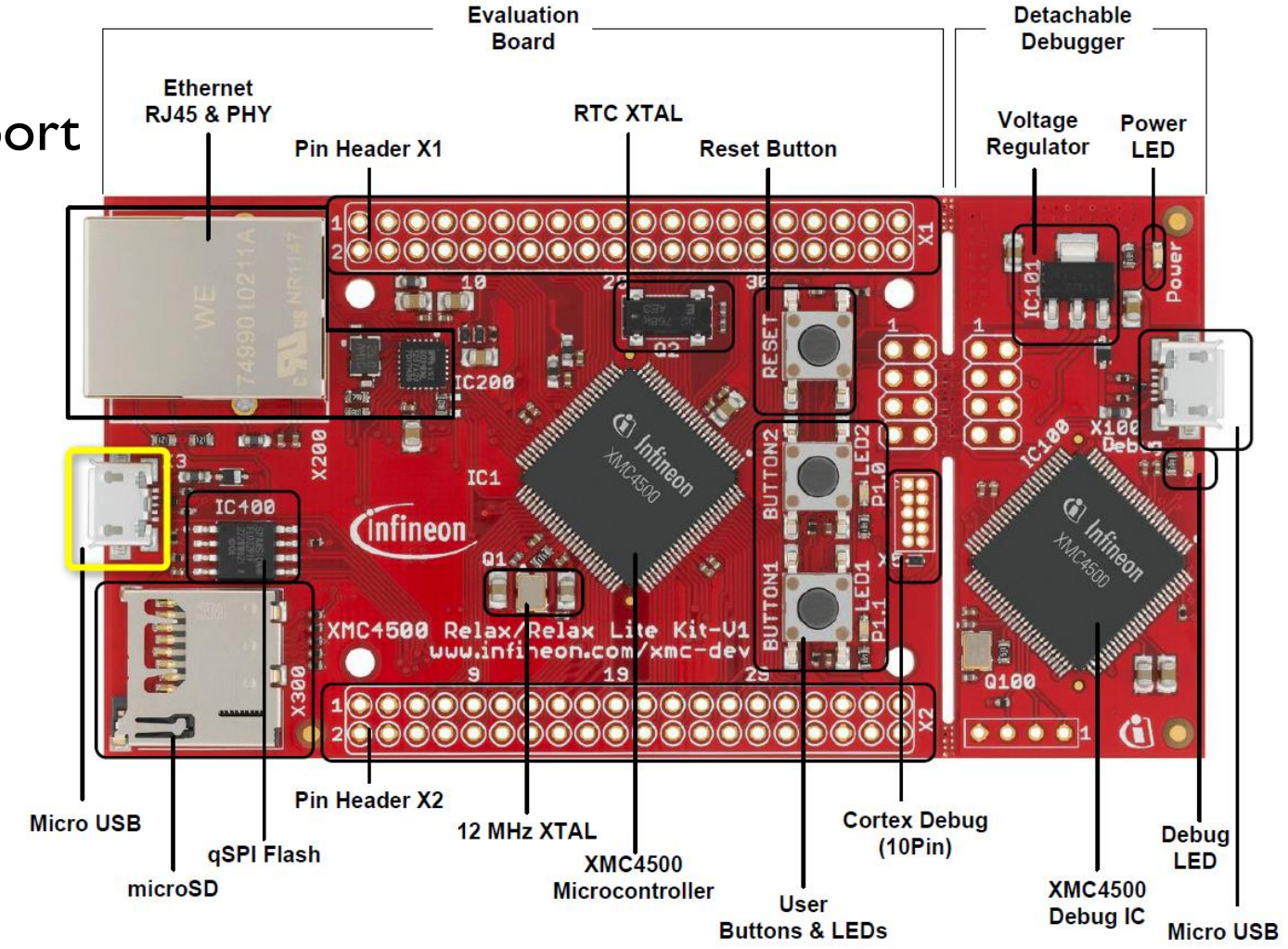


Extending the Project with USB

ARM

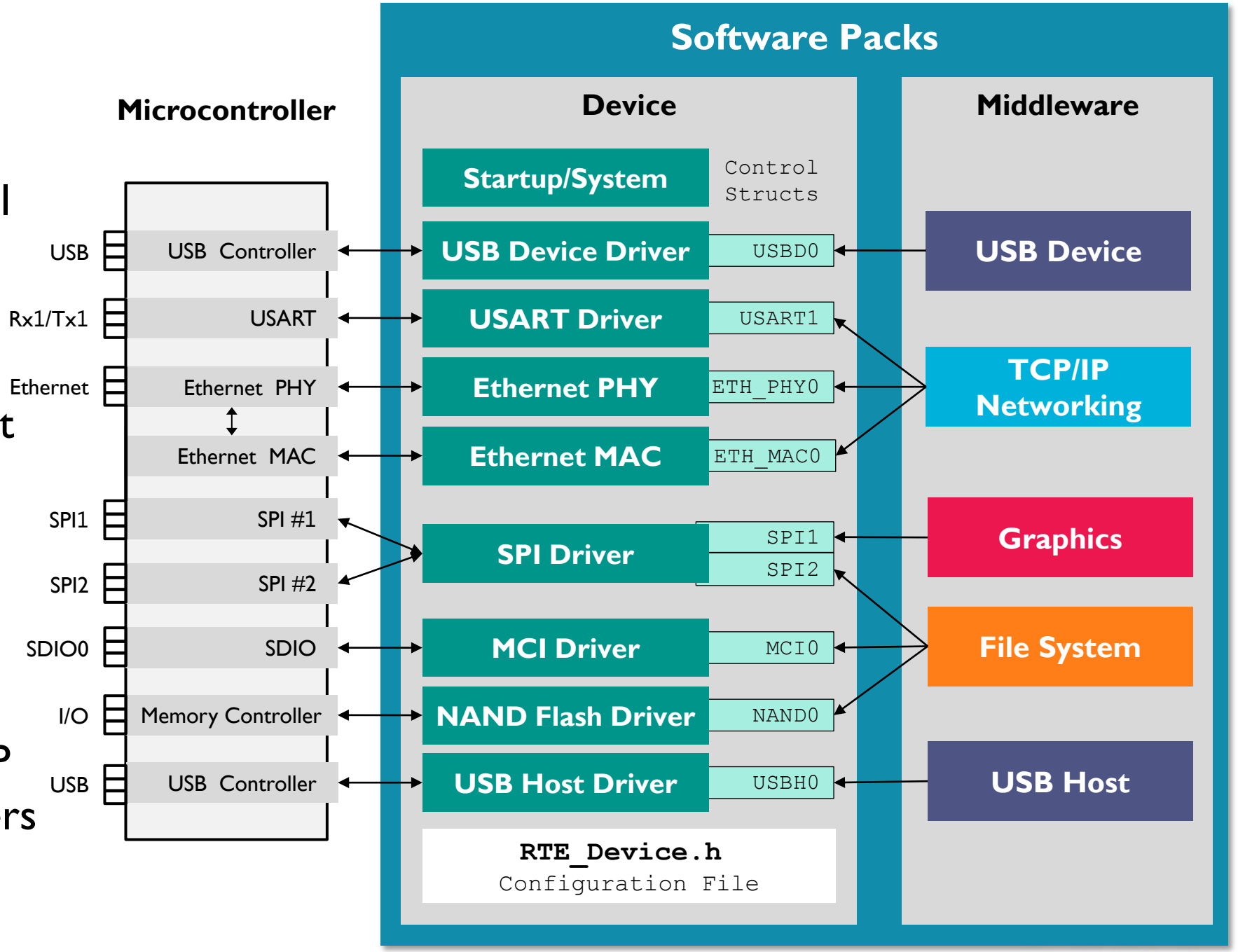
Adding USB Functionality to the Project

- 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

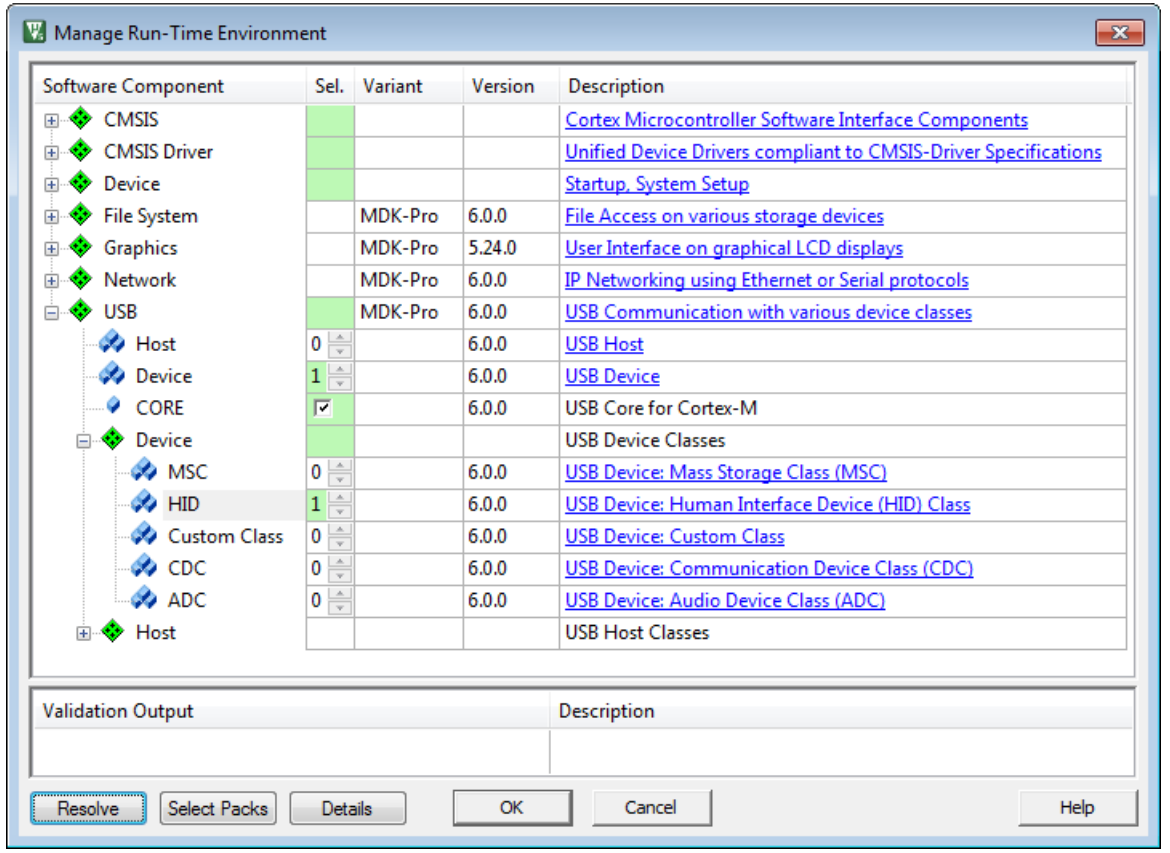
- API describing peripheral driver interfaces for middleware stacks and user applications
- Generic and independent of a specific RTOS
- Covers a wide range of use cases for the supported peripheral types
- Infineon::XMC4000_DFP provides compliant drivers for USB



Manage Run-Time Environment

Select USB Middleware Components

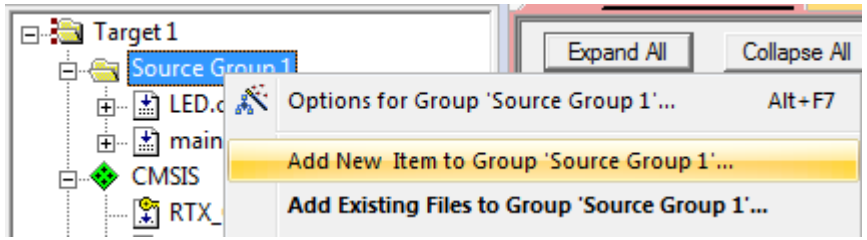
- Stop the Debug session
- Click on 
- Expand **USB**
  - Select **CORE**
- Expand **USB:Device**
  - Set **HID** to **1**
- Click **Resolve**
- Click **OK**

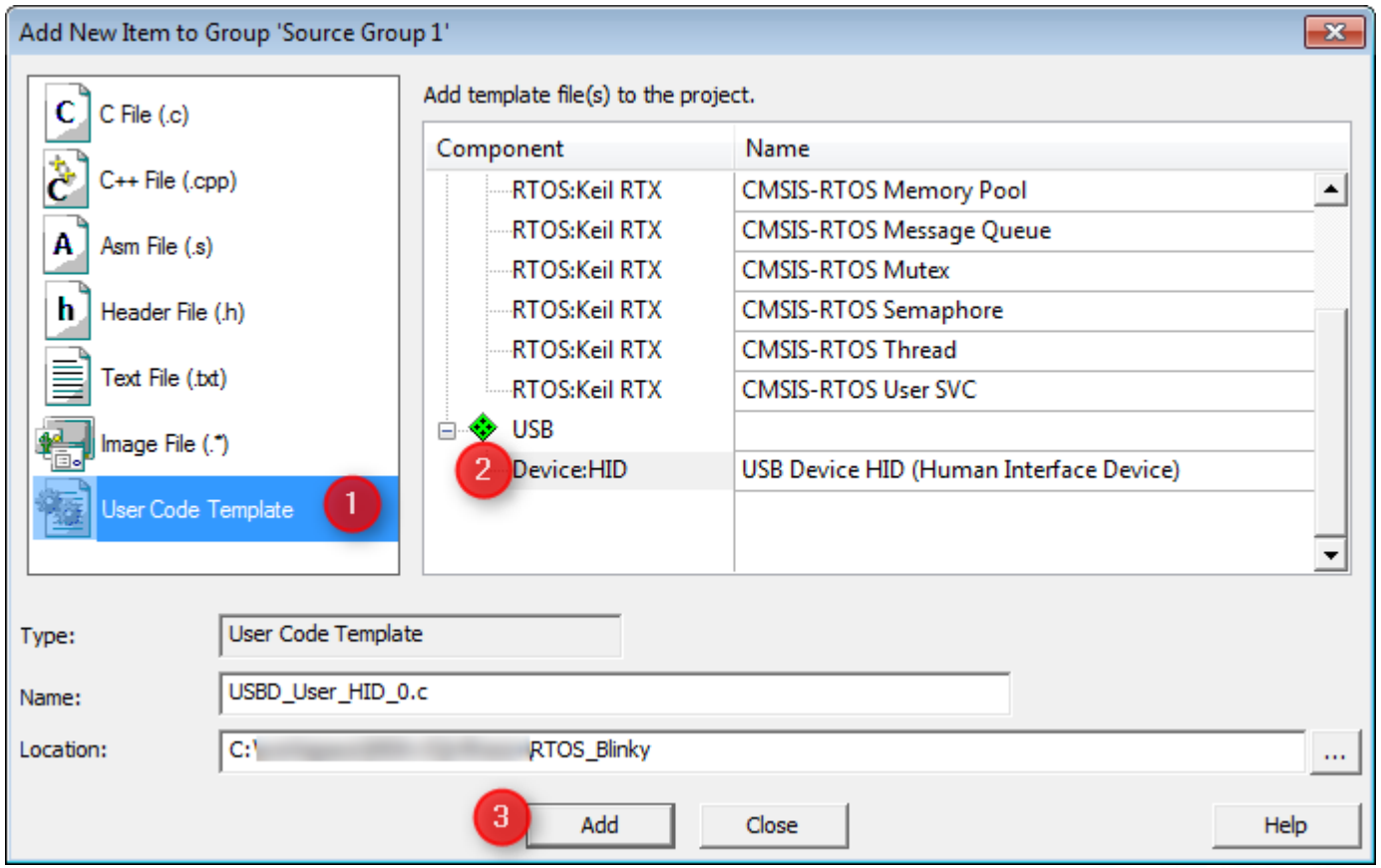


Add USB HID Template File

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

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





USBDUser_HID_0.c Template File (1)

- 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)

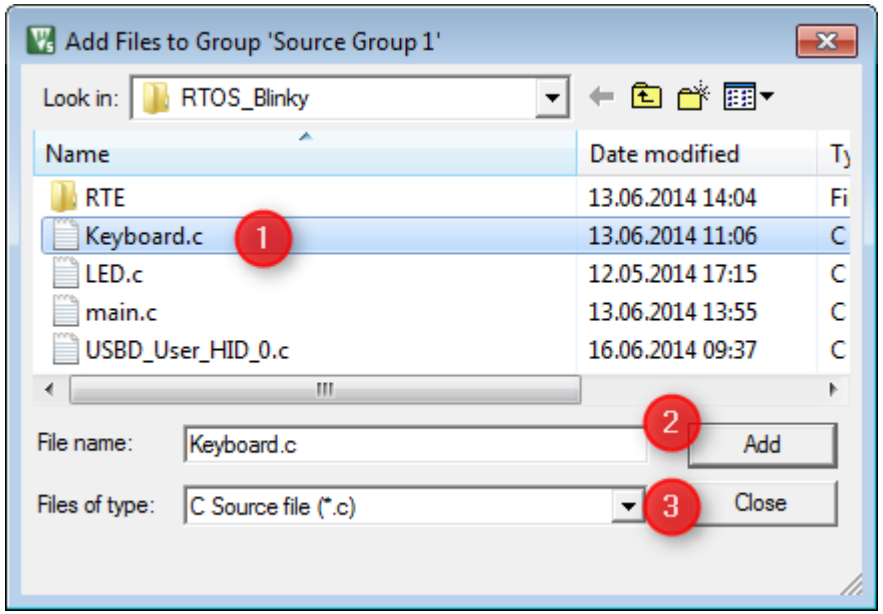
- It contains function prototypes that need to be adapted to the user application:
  - bool USBD_HID0_SetReport (uint8_t rtype, uint8_t req, uint8_t rid, const uint8_t *buf, int32_t len) {

```
  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**

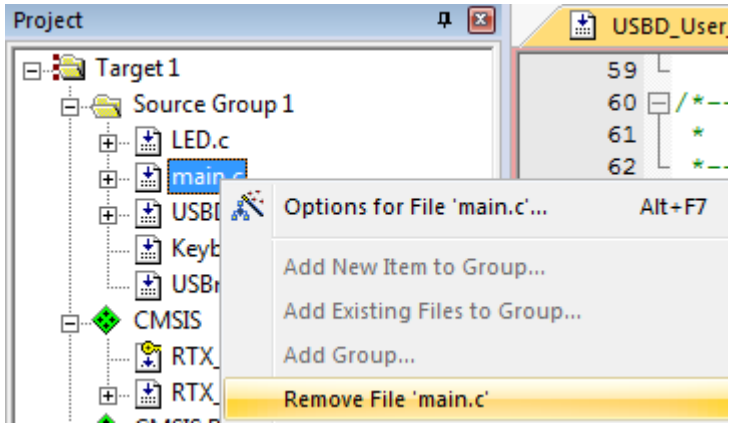


Implement the Application

- 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'**



ARM

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 (

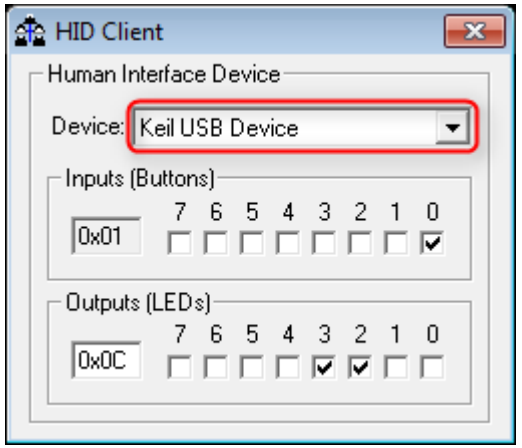
)
- Unplug the USB cable
- Plug it into the Micro-USB port **X3** next to the RJ45
- You might see a notification from Windows that drivers are being installed

ARM

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



ARM

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

ARM

Context Sensitive F1 Help

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

```
/*----------------------------------------------------------------------
 *      Main: Initialize and start RTX Kernel
 *----------------------------------------------------------------------
int main (void) {

  SystemCoreClockUpdate();
  LED_Initialize();                     /* Initialize the I

  tid_phaseA = osThreadCreate(osThread(phaseA), NULL);
  tid_phaseB =
  tid_clock  = osThreadCreate(osThread(clock),  NULL);

  osSignalSet(tid_phaseA, 0x0001);      /* set signal to ph
```

Cursor on SystemCoreClockUpdate

CMSIS COMPLIANT

CMSIS-CORE Version 3.30

CMSIS-CORE support for Cortex-M processor-based devices

CMSIS | CORE | Driver | DSP | RTOS API | Pack | SVD

Main Page | Usage and Description | Reference | Search

- CMSIS-CORE
  - Overview
  - Using CMSIS in Embedded Applicatior
  - Template Files
  - MISRA-C:2004 Compliance Exceptions
  - Register Mapping
  - Reference
    - Peripheral Access
    - System and Clock Configuration
      - SystemCoreClockUpdate
      - SystemInit
      - SystemCoreClock
    - Interrupts and Exceptions (NVIC)
    - Core Register Access
    - Intrinsic Functions for CPU Instructi
    - Intrinsic Functions for SIMD Instructi
    - Systick Timer (SYSTICK)
    - Debug Access
  - Data Structures
  - Data Fields

void SystemCoreClockUpdate (void)

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.

void SystemInit (void)

Initializes the microcontroller system. Typically, this function configures the oscillator (PLL) that is part of the microcontroller device. For systems with a variable clock speed, it updates the variable **SystemCoreClock**. SystemInit is called from the file ***startup_device***.

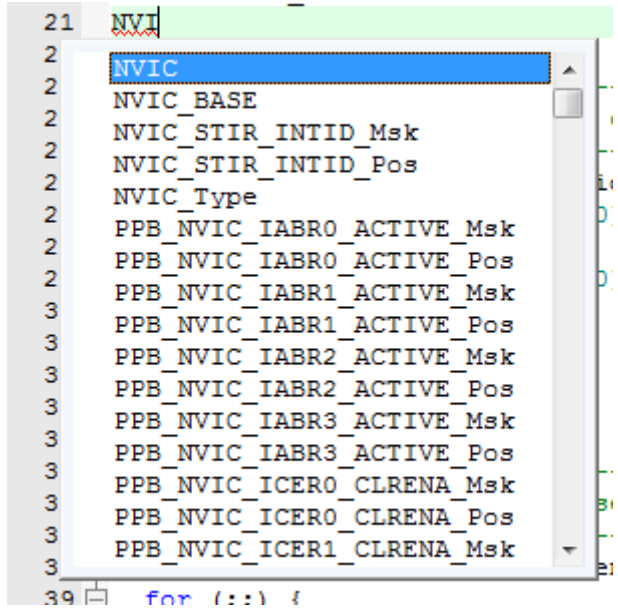
Variable Documentation

uint32_t SystemCoreClock

Holds the system core clock, which is the system clock frequency supplied to the SysTick timer and the processor core clock. This variable can be used by debuggers to query the frequency of the

Code Completion

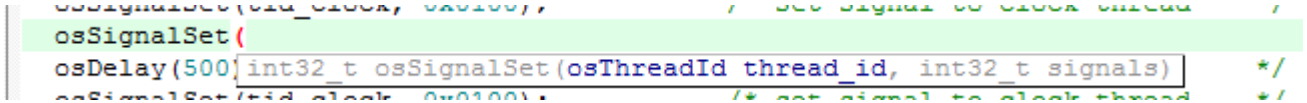
- List showing all program symbols that contain the currently typed characters
- List appears after typing:
  - 3 characters (default)
  - A **trigger character**:
    - . For structures or classes
    - -> For pointer structures
    - :: For symbols within a specific scope
- **CTRL+<space>**
- Insert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



ARM

Parameter Information

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

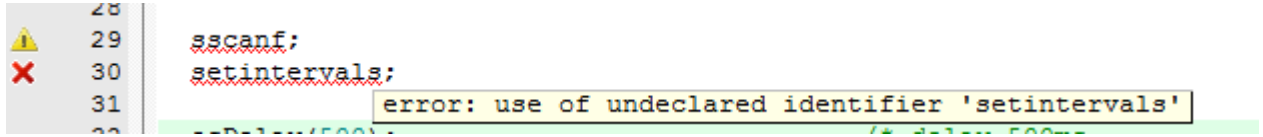


- While typing, the bolded text indicates the next required parameter

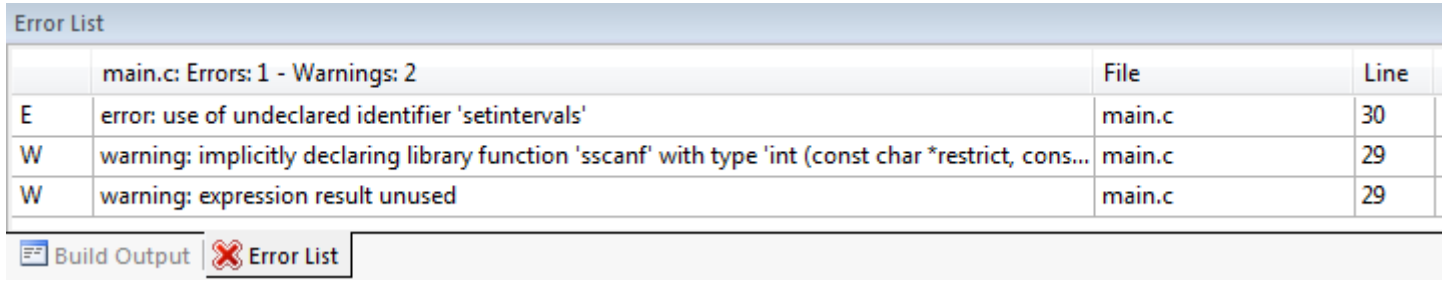
ARM

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



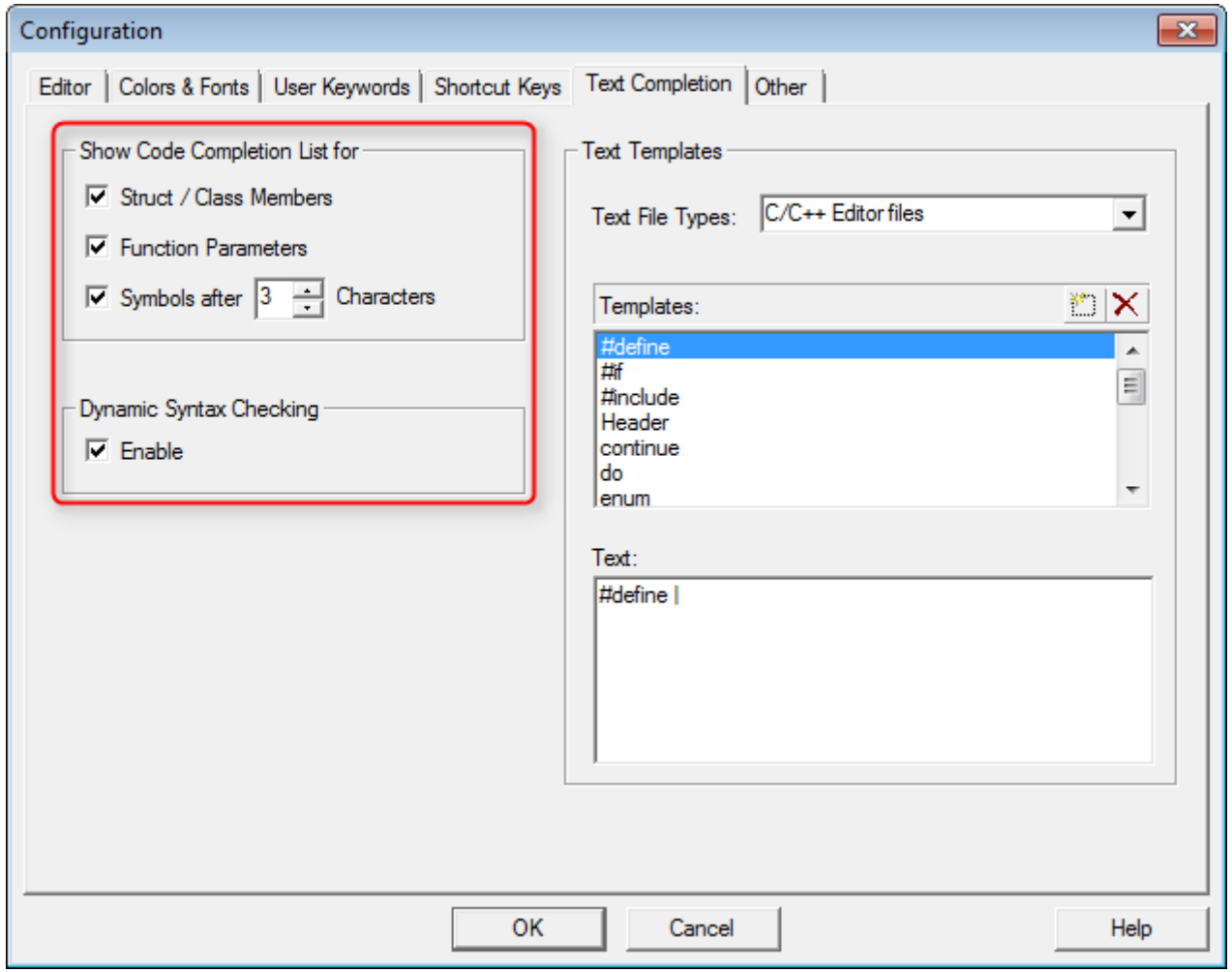
- Hover the mouse on an icon for details about the syntax violation
- Also in:
  - View → Error List Window



ARM

Feature Configuration

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



ARM

Collaterals on USB Stick

- AN258 - Using DAVE3 with MDK Version 5
- AN260 - Infineon XMC1100 2Go Lab for uVision V5
- AN263 - Infineon XMC1200 Boot Kit Lab for uVision V5
- Getting Started – Create Applications with MDK Version 5
- µVision 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

ARM