FreeRTOS and emWin for i.MX RT platform

Krzysztof Chojnowski

SoMLabs

SOM
System on Module

CB
Carrier Board

DK
Development Kit

Engineering
Since 2003 delivering proven designs

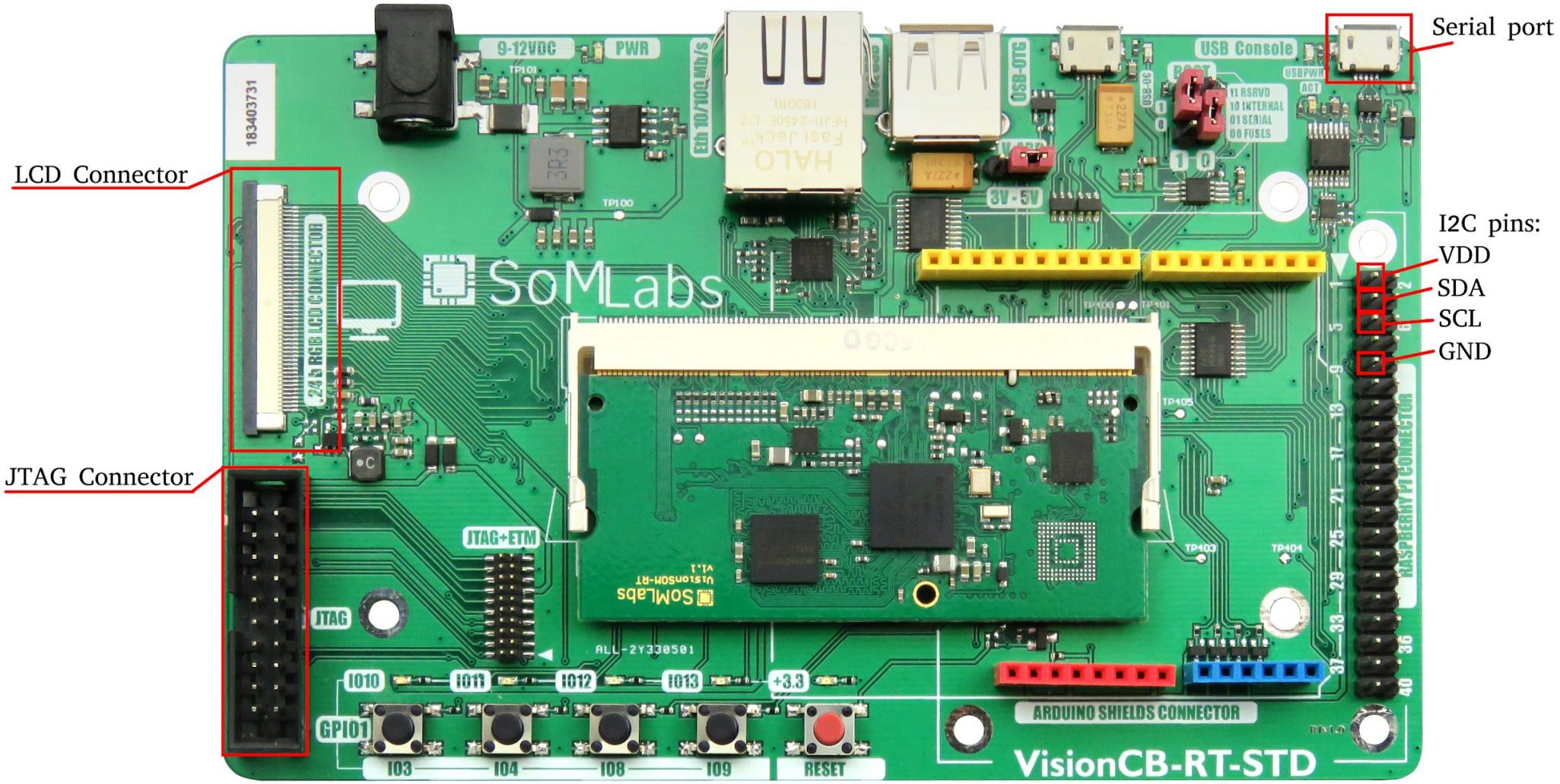
Agenda

- Introduction to the development environment
- Introduction to (Free)RTOS
- Creating and synchronizing FreeRTOS tasks
- Tasks communication
- emWin library

Exercises

- Exercise 0 - building and running example project
- Exercise 1 - creating tasks
- Exercise 2 - handling interrupts
- Exercise 3 - queues
- Exercise 4 - building GUI with emWin
- Exercise 5 - handling GUI events

VisionCB-RT-STD



SDK

i.MX RT1050 Crossover Processor with Arm® Cortex®-M7 core

Follow

OVERVIEW | DOCUMENTATION | TOOLS & SOFTWARE | BUY/PARAMETRICS | PACKAGE/QUALITY | TRAINING & SUPPORT

Recommended Tools and Software

i.MX RT1050 Evaluation Kit

The i.MX RT1050 EVK is an entry-level, low-cost development platform featuring the i.MX RT1050 crossover processor based on Arm Cortex-M7.

Buy Options

MCUXpresso Software Development Kit (SDK)

Software development kit for designing with Kinetis, LPC, i.MX controllers based on Arm Cortex-M cores, with integrated stacks, middleware, examples

Software Development Kits

Download Options

MCUXpresso Integrated Development Environment (IDE)

Easy-to-use software development tools for Kinetis, LPC, i.MX controllers based on Arm Cortex-M cores - GNU, Eclipse, profiling, debugger, trace

Download Options

SDK

MCUXpresso Software Development Kit (SDK)

Follow

OVERVIEW | DOCUMENTATION | DOWNLOADS | DEVELOPMENT TOOLS | TRAINING & SUPPORT

Jump To

Overview

The MCUXpresso SDK is a comprehensive software enablement package designed to simplify and accelerate application development with NXP's LPC and Kinetis® microcontrollers and i.MX RT crossover processors based on Arm® Cortex®-M cores. The MCUXpresso SDK includes production-grade software with integrated RTOS (optional), integrated stacks and middleware, reference software, and more.

Underscoring highest quality, the MCUXpresso SDK is MISRA compliant and checked with Coverity® static analysis tools and is available in custom downloads based on user selections of MCU, evaluation board, and optional software components.

Features

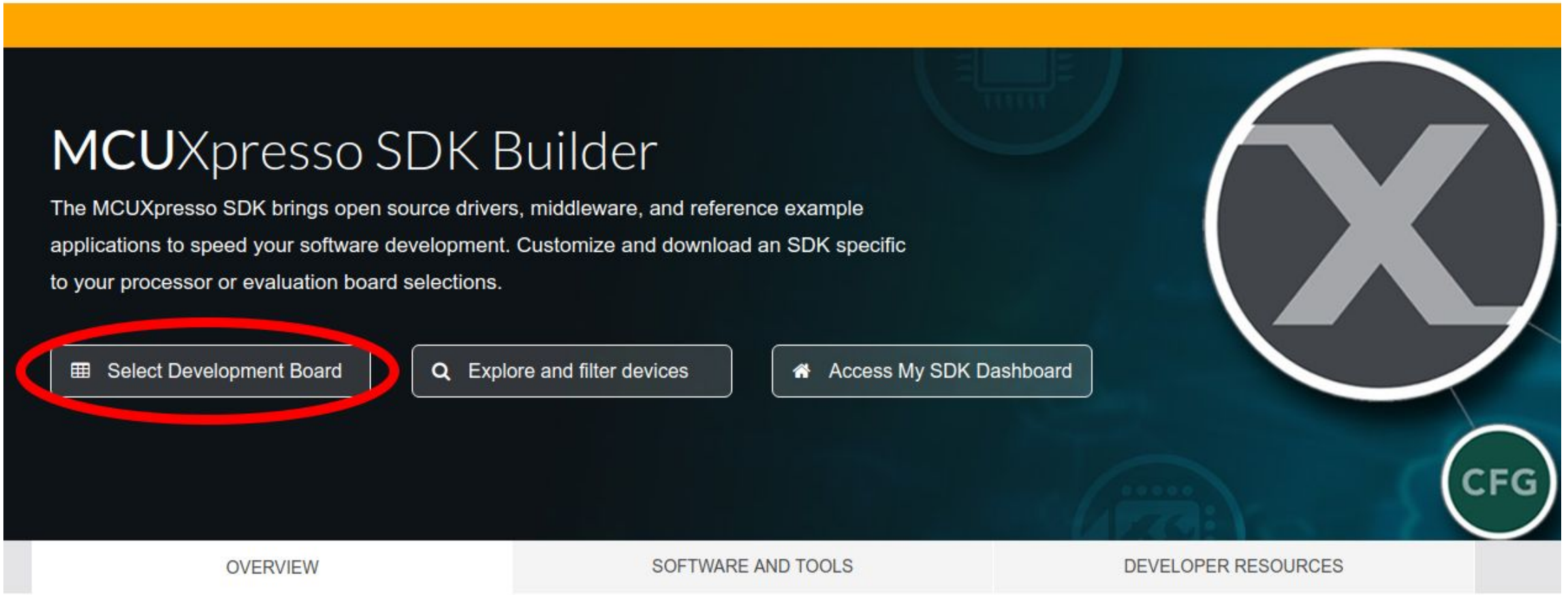
The MCUXpresso SDK consists of the following runtime software components:

- Arm® CMSIS-CORE startup and device header files and CMSIS-DSP standard libraries
- Open-source peripheral drivers that provide stateless, high performance, easy-to-use APIs
- Drivers for communication peripherals also include high-level transactional APIs for high-performance data transfers and RTOS wrappers that leverage native RTOS services to better comply with the RTOS cases
- High-quality software: all drivers and startup code are

More

User Guide | Download

SDK


MCUXpresso SDK Builder
The MCUXpresso SDK brings open source drivers, middleware, and reference example applications to speed your software development. Customize and download an SDK specific to your processor or evaluation board selections.
Select Development Board
Explore and filter devices
Access My SDK Dashboard
CFG
OVERVIEW
SOFTWARE AND TOOLS
DEVELOPER RESOURCES

SDK

Select Development Board

Search for your board or kit to get started.

Search by Name

mimxrt1052

Select a Device, Board, or Kit

- Boards
- Kits
- Processors
  - MIMXRT1052xxxxB
  - MIMXRT1052xxxxx — Deprecated

Name your SDK

SDK_2.5.0_MIMXRT1052xxxxB

Don't use: <, >, :, ", /, |, ?, *, \ in the name of your SDK

Hardware Details

Included Part Numbers	MIMXRT1052CVL5B, MIMXRT1052DVL6B, MIMXRT1052CVJ5B, MIMXRT1052DVJ6B
Board(s)	EVKB-IMXRT1050
Device	MIMXRT1052
Core Type / Max Freq	Cortex-M7F / 600MHz
Device Memory Size	0 KB Flash 512 KB RAM

Actions

- Build MCUXpresso SDK
- Explore selection with Clocks tool
- Explore selection with Pins tool

SDK

SDK Builder

Generate a downloadable SDK archive for use with desktop MCUXpresso Tools.

Developer Environment Settings

Selections here will impact files and examples projects included in the SDK and Generated Projects

Host OS: Linux

Toolchain / IDE: MCUXpresso IDE

Select Optional Middleware

Add middleware, operating systems, and software libraries to your SDK.

Add software component

Click the link below to request this specific MCUXpresso SDK Build

In general, SDK builds should complete within a few minutes.
You will be notified via email and notifications in the upper right corner of this webpage.

Request Build →

Archive Name: SDK_2.5.0_MIMXRT1052xxxxB (3)

Don't use: < > : " / | ? * \ in the name of your SDK

Hardware Details

Included Part Numbers	MIMXRT1052CVL5B, MIMXRT1052DVL6B, MIMXRT1052CVJ5B, MIMXRT1052DVJ6B
Board(s)	EVKB-IMXRT1050
Device	MIMXRT1052
Core Type / Max Freq	Cortex-M7F / 600MHz
Device Memory Size	0 KB Flash 512 KB RAM

SDK Details

SDK Version:	2.5.0 (released 2018-12-17)
Host OS:	Linux
Toolchain:	MCUXpresso IDE **SDK v2.5.x requires MCUXpresso IDE v10.3.x or later**
Middleware:	**emWin, Amazon-FreeRTOS Kernel**

Documentation

Base SDK: MCUXpresso SDK API Reference Manual

SDK

MCUXpresso SDK Dashboard

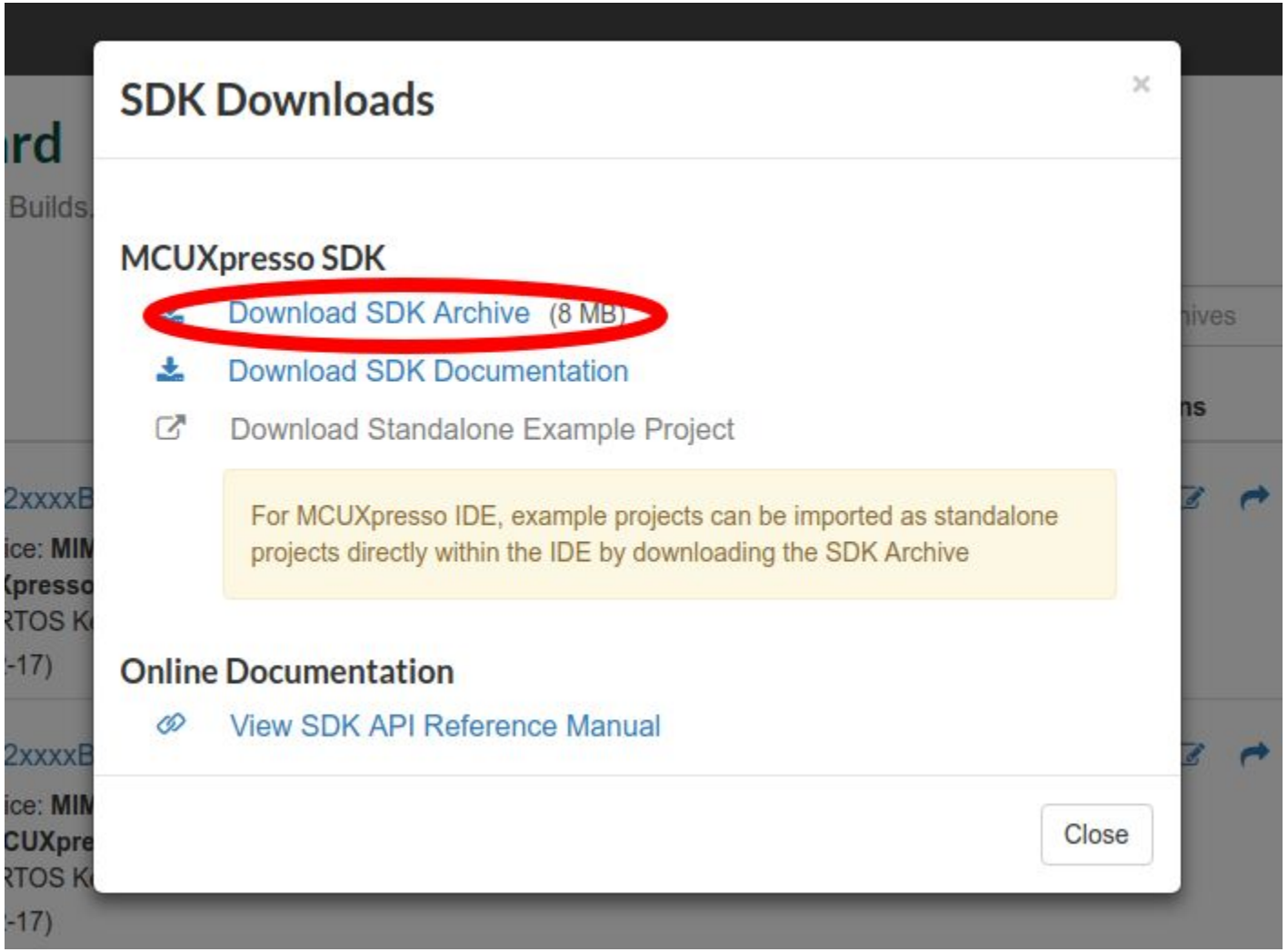
Access, Download, and Share your requested SDK Builds.

My Recent SDKs

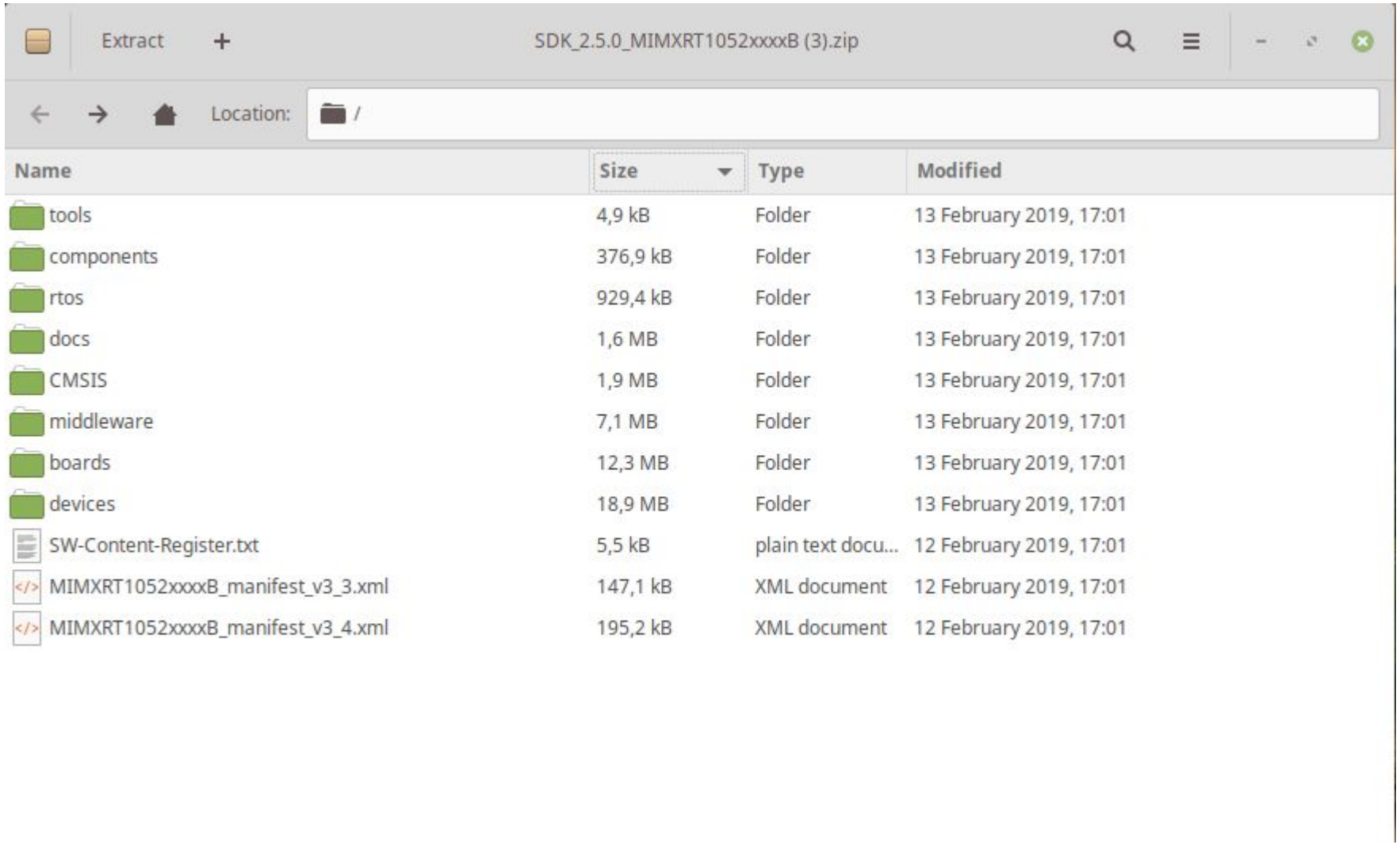
Search all archives

	SDK Archive Details	Actions
	SDK_2.5.0_MIMXRT1052xxxxB (3) NEW Build Date: **2019-02-13**, Device: **MIMXRT1052xxxxB** OS: **Linux**, Toolchain: **MCUXpresso IDE** Components: Amazon-FreeRTOS Kernel, emWin SDK Version: **2.5.0** (2018-12-17)	
	SDK_2.5.0_MIMXRT1052xxxxB Build Date: **2019-01-09**, Device: **MIMXRT1052xxxxB** OS: **Windows**, Toolchain: **MCUXpresso IDE** Components: Amazon-FreeRTOS Kernel, emWin SDK Version: **2.5.0** (2018-12-17)	
	EVKB-IMXRT1050 Build Date: **2018-09-01**, Board: **EVKB-IMXRT1050** OS: **Linux**, Toolchain: **MCUXpresso IDE** Components: CMSIS DSP Library, Amazon-FreeRTOS Kernel, emWin, lwIP, FatFS, USB stack SDK Version: **KSDK 2.4.2** (2018-08-02)	Update Available

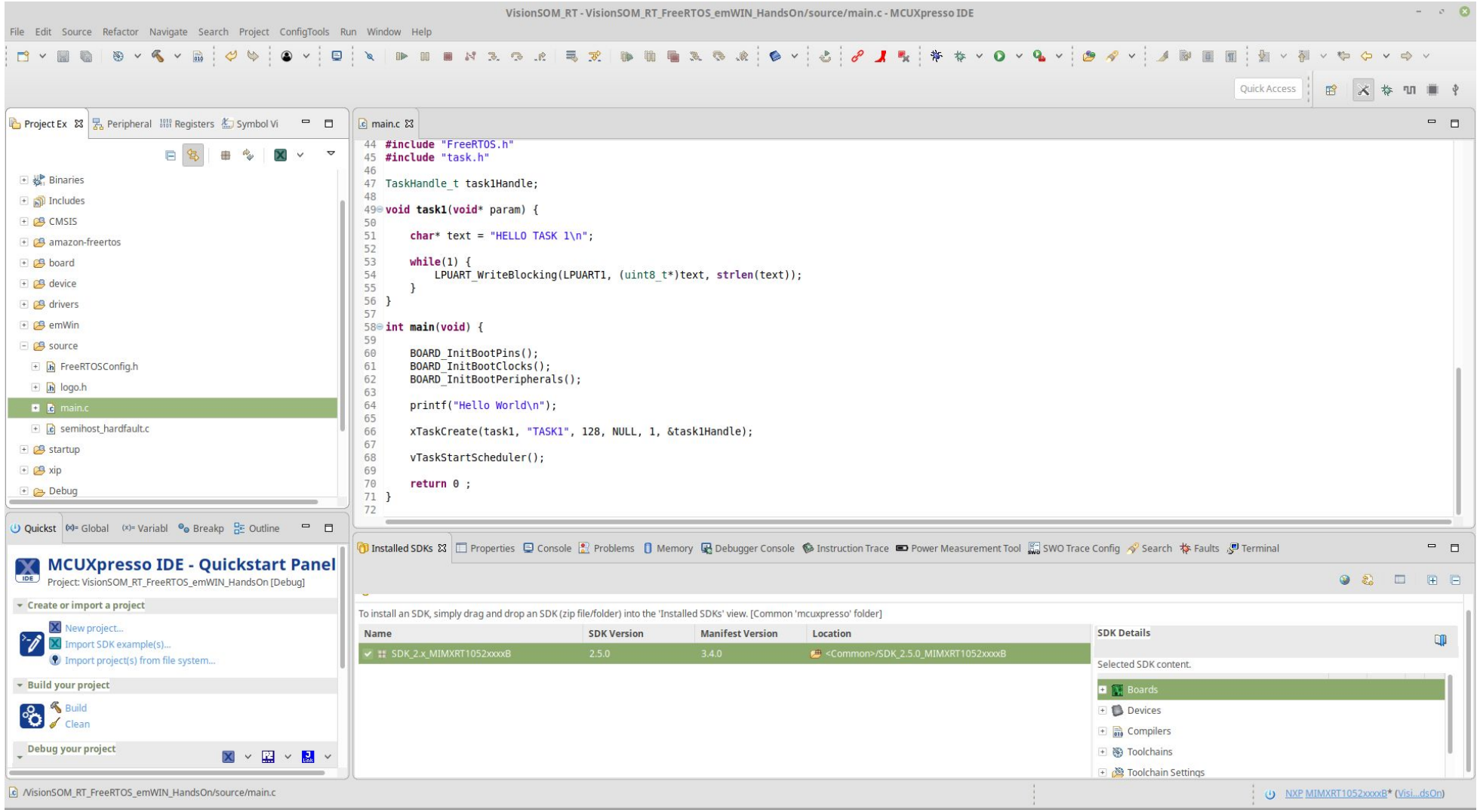
SDK


SDK Downloads
MCUXpresso SDK
Download SDK Archive (8 MB)
Download SDK Documentation
Download Standalone Example Project
For MCUXpresso IDE, example projects can be imported as standalone projects directly within the IDE by downloading the SDK Archive
Online Documentation
View SDK API Reference Manual
Close

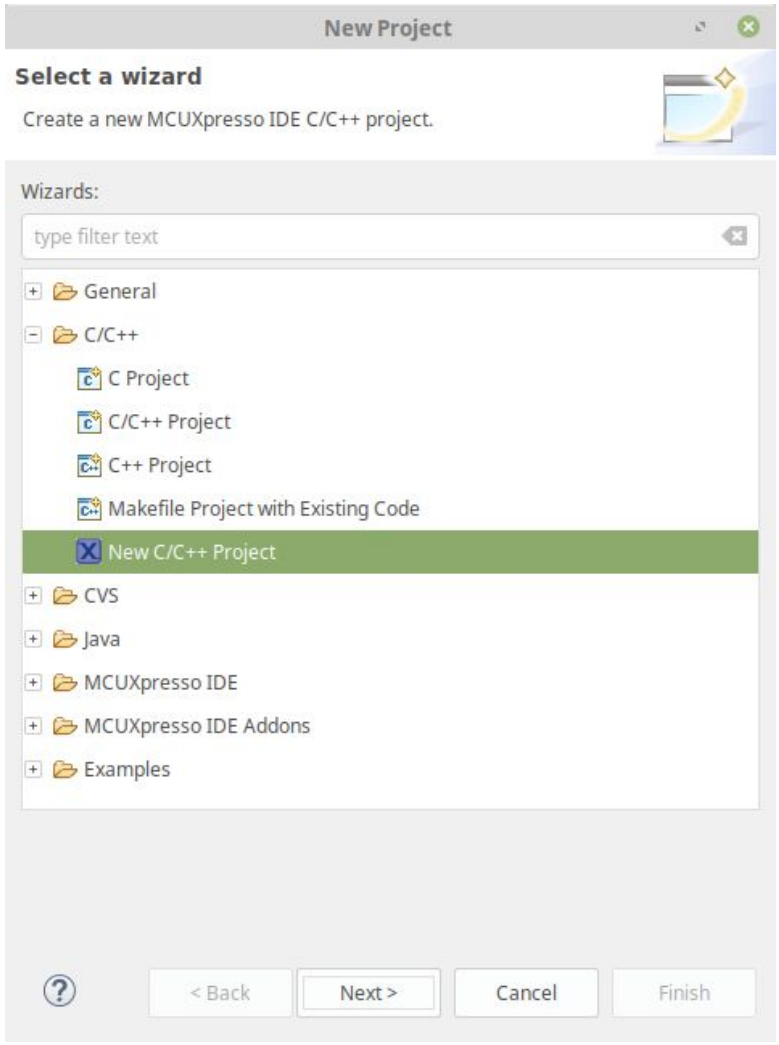
SDK

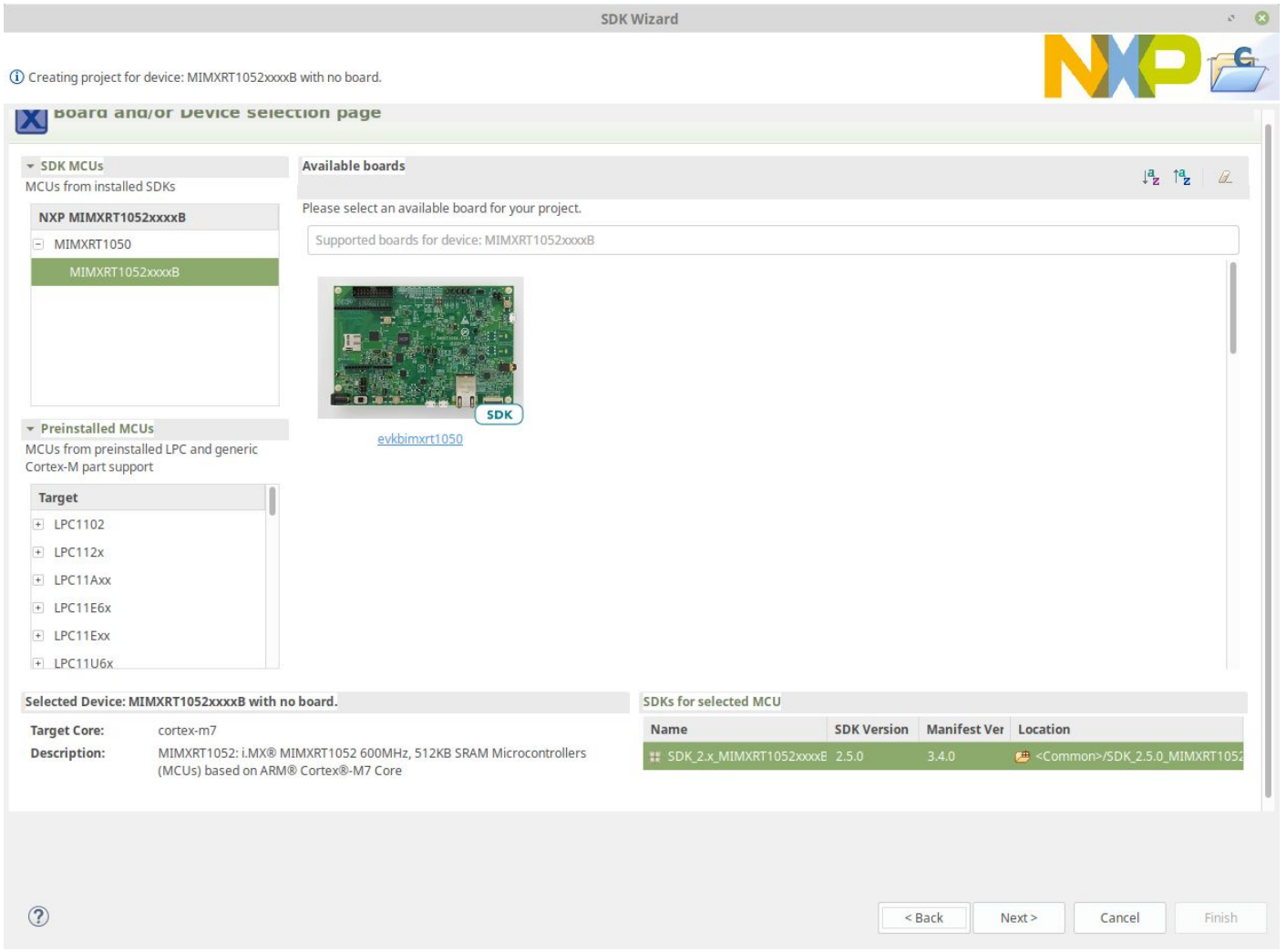

Extract
SDK_2.5.0_MIMXRT1052xxxxB (3).zip
Location: /
Name
Size
Type
Modified
tools 4,9 kB Folder 13 February 2019, 17:01
components 376,9 kB Folder 13 February 2019, 17:01
rtos 929,4 kB Folder 13 February 2019, 17:01
docs 1,6 MB Folder 13 February 2019, 17:01
CMSIS 1,9 MB Folder 13 February 2019, 17:01
middleware 7,1 MB Folder 13 February 2019, 17:01
boards 12,3 MB Folder 13 February 2019, 17:01
devices 18,9 MB Folder 13 February 2019, 17:01
SW-Content-Register.txt 5,5 kB plain text docu... 12 February 2019, 17:01
MIMXRT1052xxxxB_manifest_v3_3.xml 147,1 kB XML document 12 February 2019, 17:01
MIMXRT1052xxxxB_manifest_v3_4.xml 195,2 kB XML document 12 February 2019, 17:01

MCUXpresso

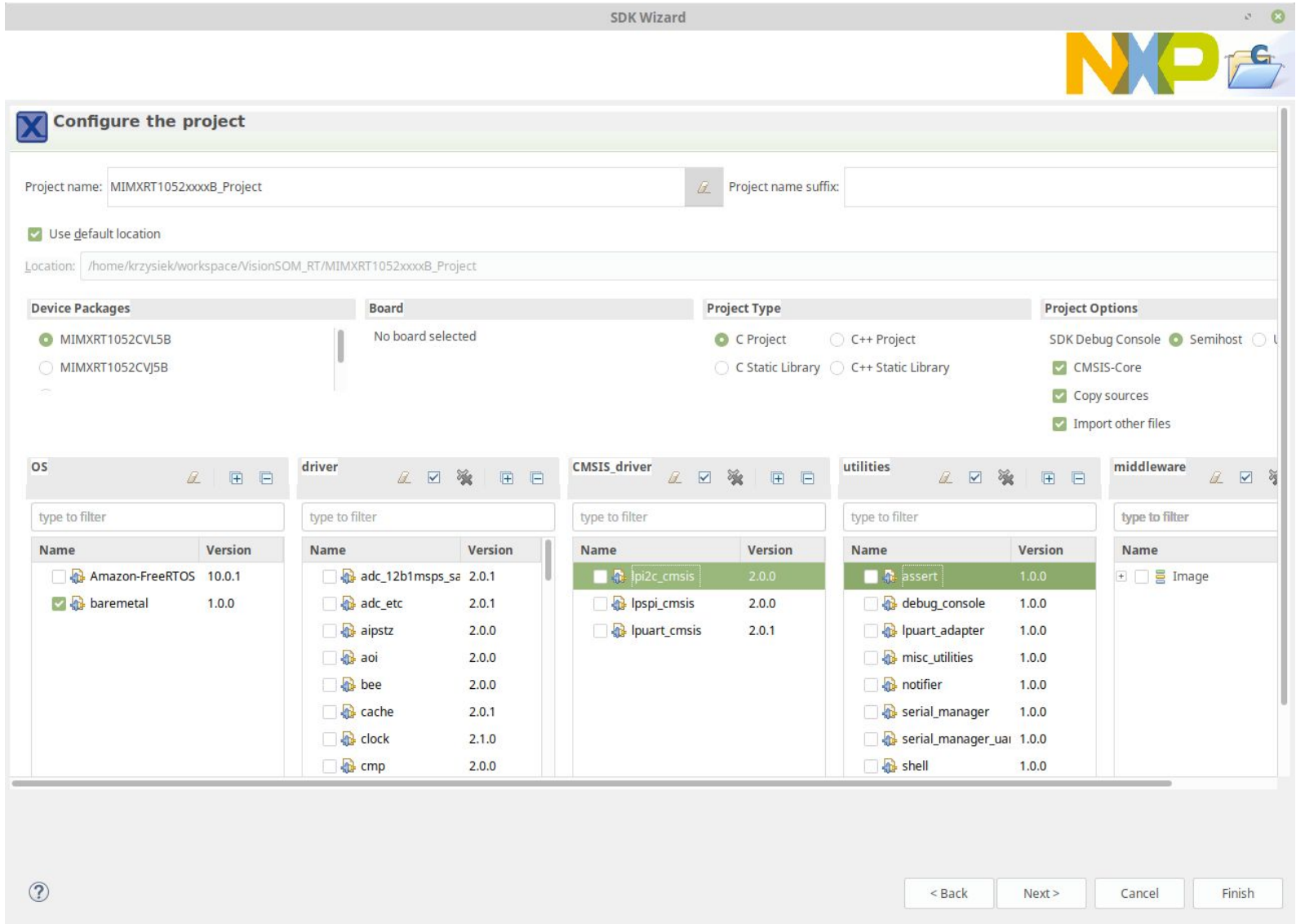


MCUXpresso

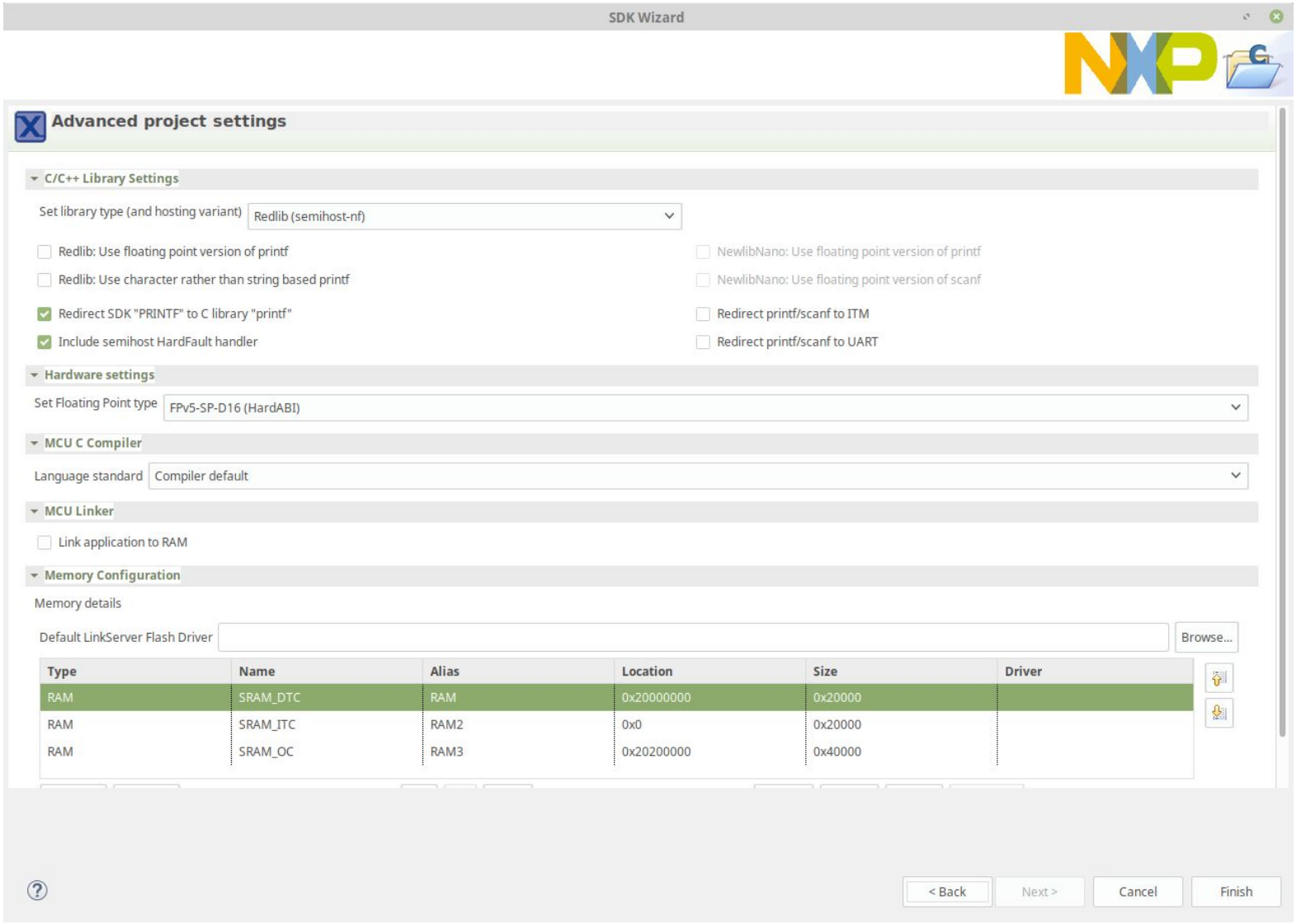

New Project
Select a wizard
Create a new MCUXpresso IDE C/C++ project.
Wizards:
type filter text
General
C/C++
C Project
C/C++ Project
C++ Project
Makefile Project with Existing Code
New C/C++ Project
CVS
Java
MCUXpresso IDE
MCUXpresso IDE Addons
Examples
< Back
Next >
Cancel
Finish


SDK Wizard
Creating project for device: MIMXRT1052xxxxB with no board.
Board and/or Device selection page
SDK MCUs
MCUs from installed SDKs
NXP MIMXRT1052xxxxB
MIMXRT1050
MIMXRT1052xxxxB
Preinstalled MCUs
MCUs from preinstalled LPC and generic Cortex-M part support
Target
LPC1102
LPC112x
LPC11Axx
LPC11E6x
LPC11Exx
LPC11U6x
Available boards
Please select an available board for your project.
Supported boards for device: MIMXRT1052xxxxB
SDK
evkbimxrt1050
Selected Device: MIMXRT1052xxxxB with no board.
Target Core: cortex-m7
Description: MIMXRT1052: i.MX® MIMXRT1052 600MHz, 512KB SRAM Microcontrollers (MCUs) based on ARM® Cortex®-M7 Core
SDKs for selected MCU
Name
SDK Version
Manifest Ver
Location
SDK_2.x_MIMXRT1052xxxxB
2.5.0
3.4.0
<Common>/SDK_2.5.0_MIMXRT1052
< Back
Next >
Cancel
Finish

MCUXpresso



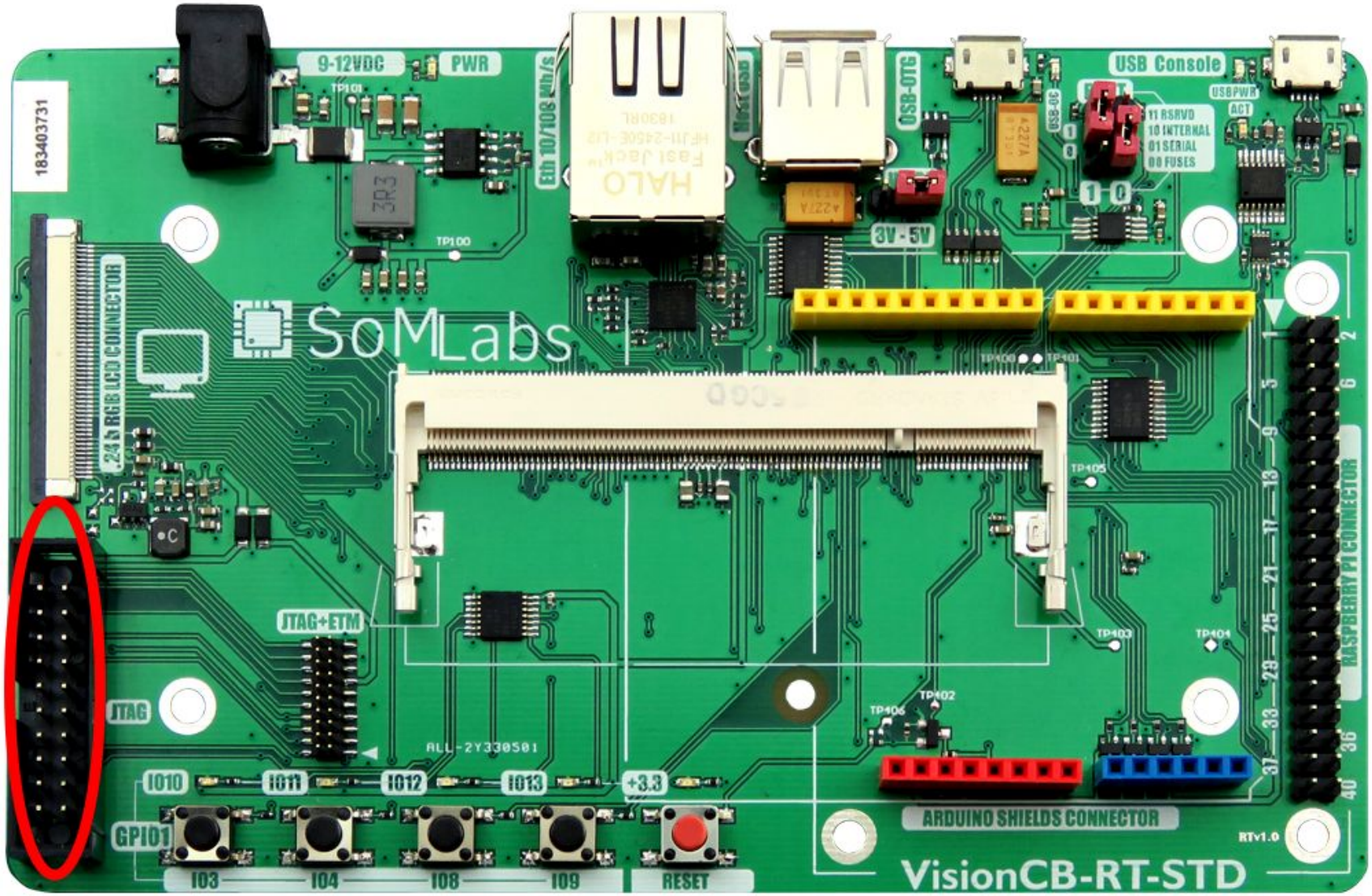
MCUXpresso



Exercise 0

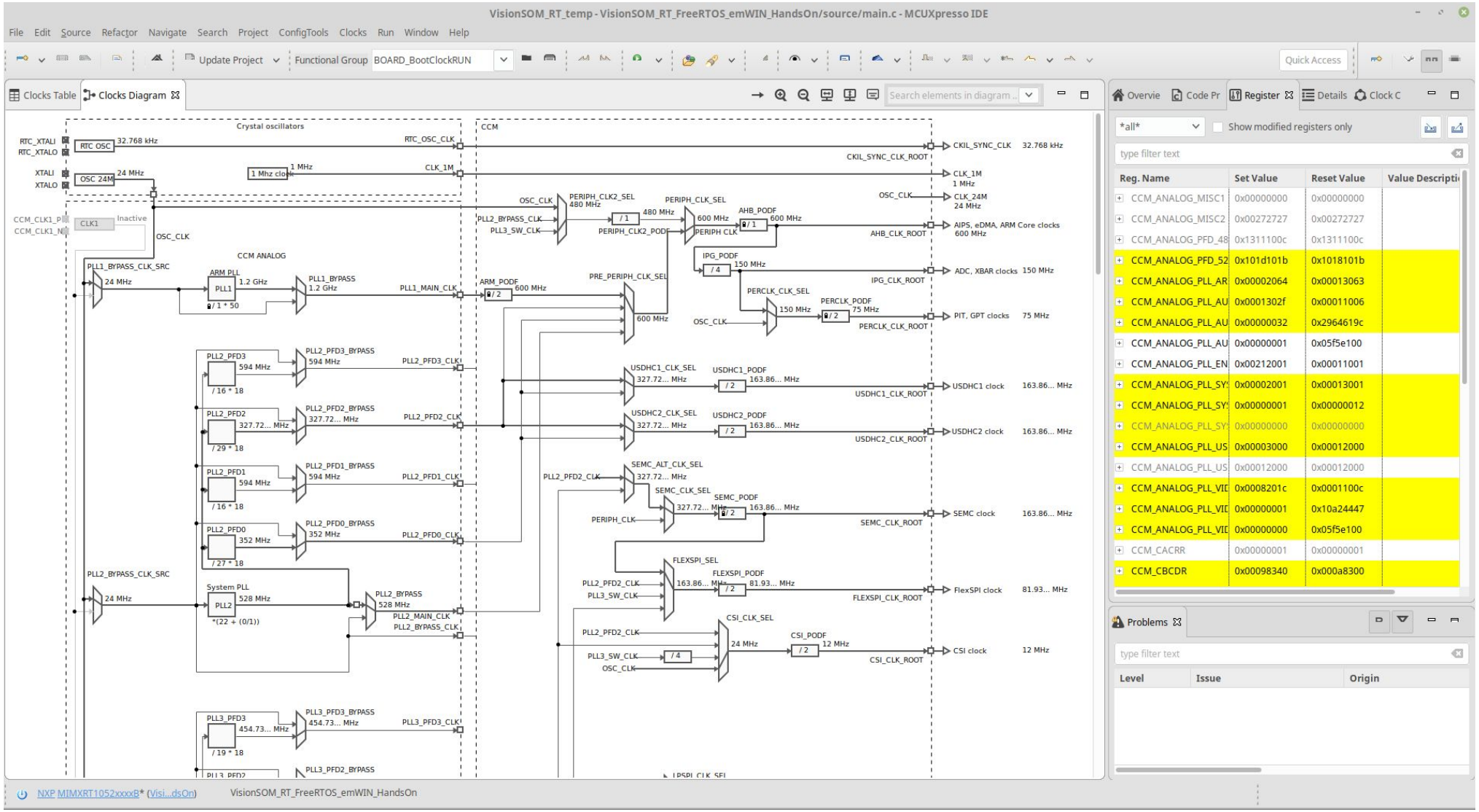
- Import SDK
- Import a template project
- build project

Exercise 0

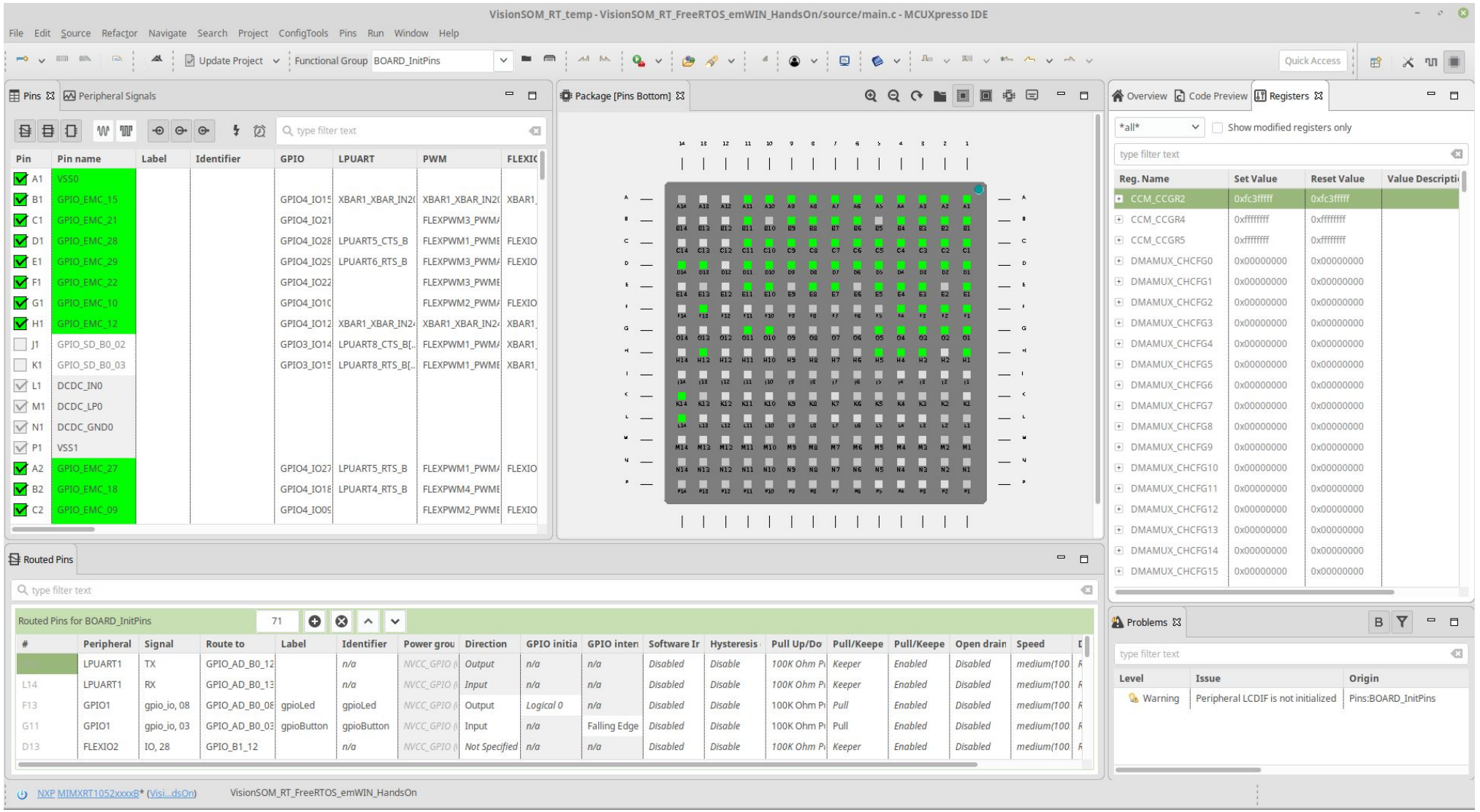


Proven Partner NXP

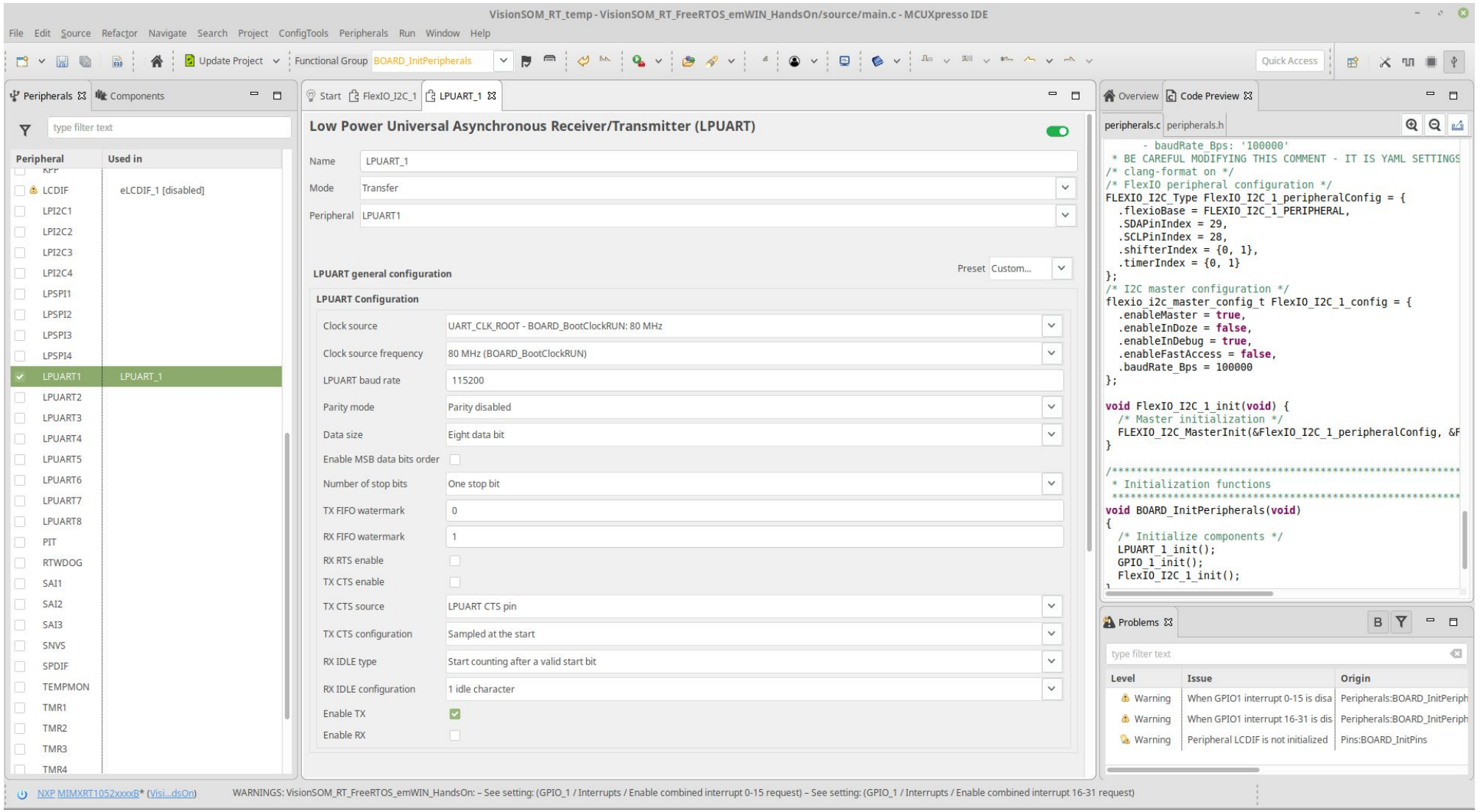
MCUXpresso



MCUXpresso



MCUXpresso



SEGGER J-Link

- Modification of the JLinkDevices.xml script for QSPI programming

```
/opt/SEGGER/JLink
C:\Program Files (x86)\SEGGER\JLink_V640
```

- Section

```
<!--                    -->
<!-- NXP (iMXRT105x)    -->
<!--                    -->
```

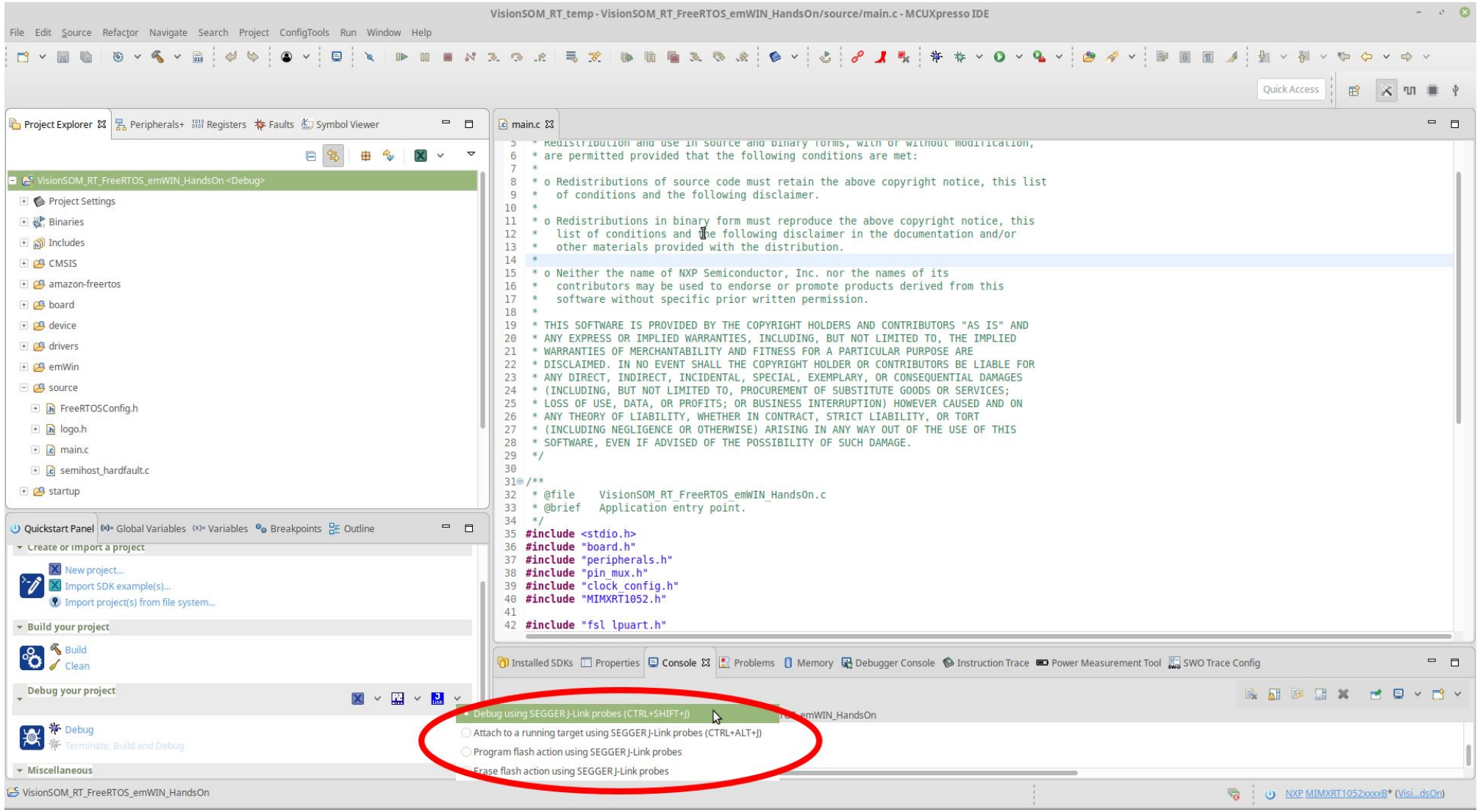
- Loader value

```
Devices/NXP/iMXRT105x/NXP_iMXRT105x_HyperFlash.elf
```

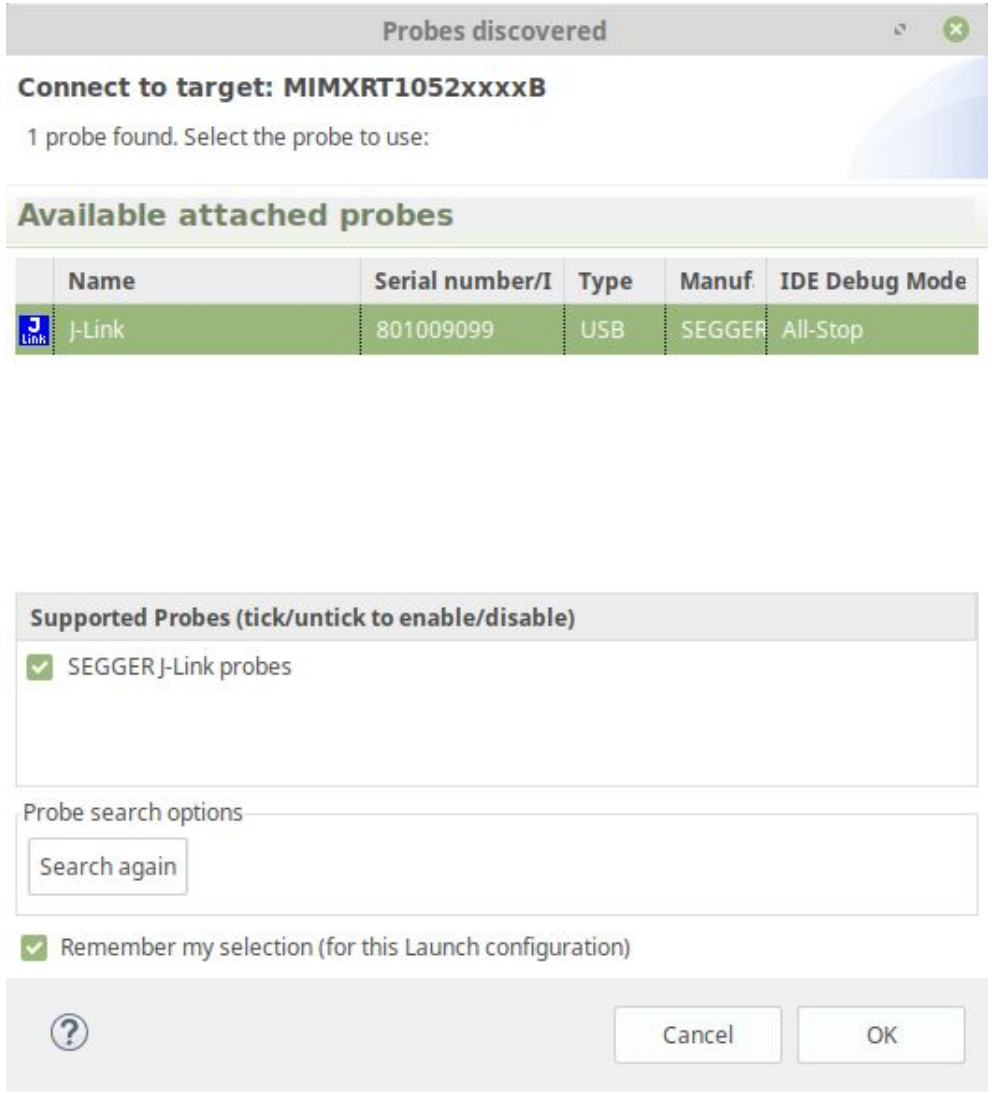
should be:

```
Devices/NXP/iMXRT105x/NXP_iMXRT105x_QSPI.elf
```

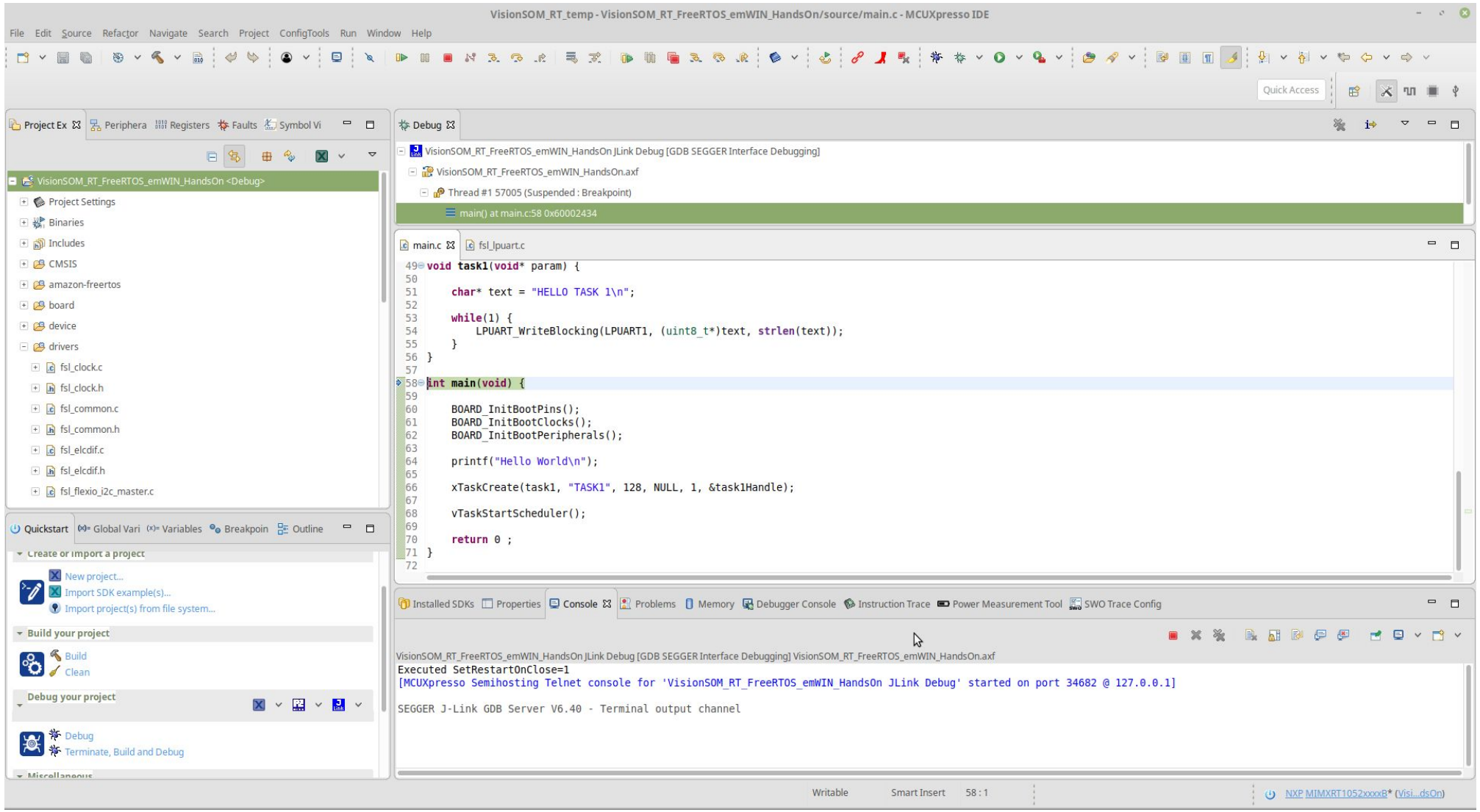
SEGGER J-Link



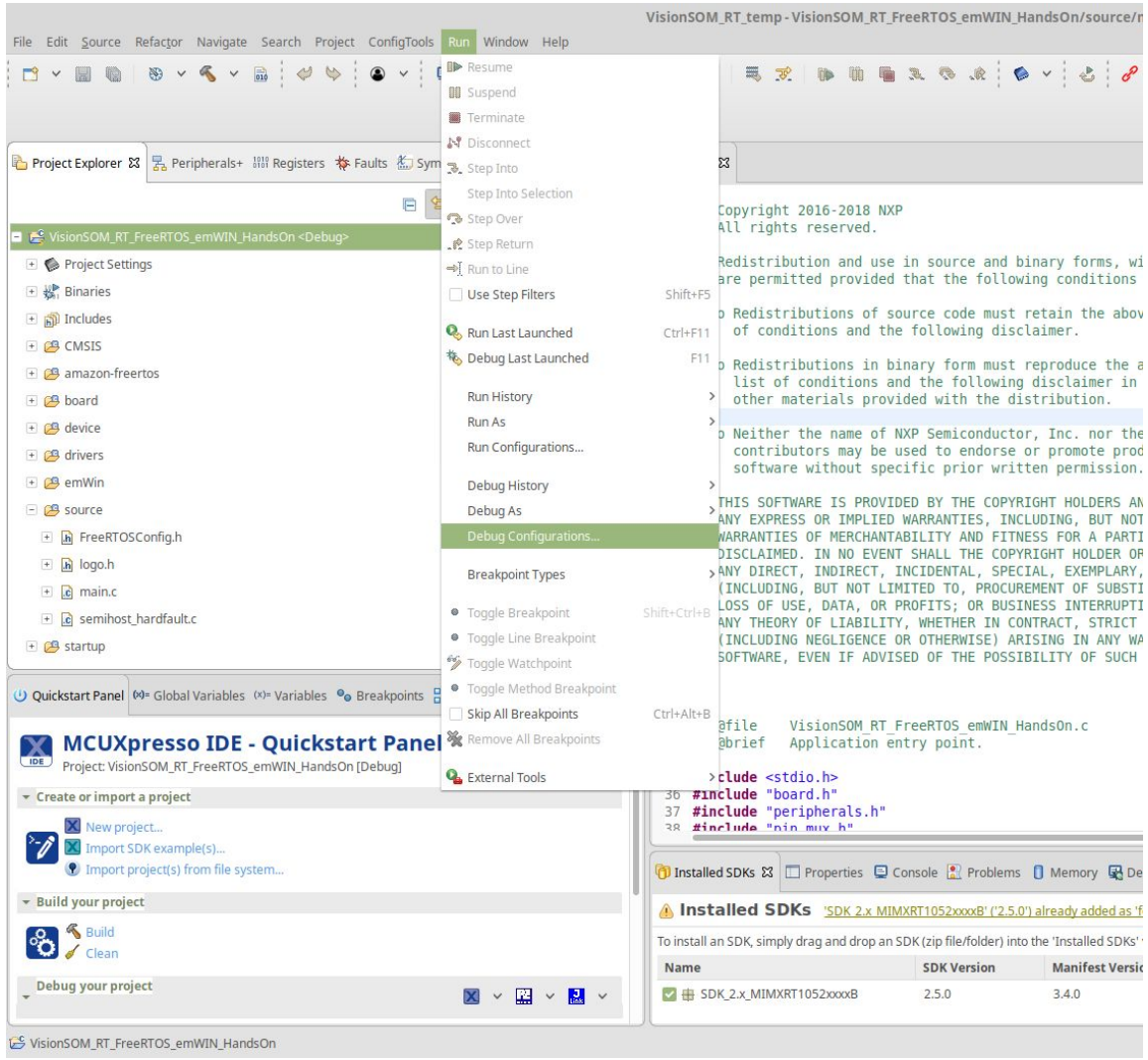
SEGGER J-Link



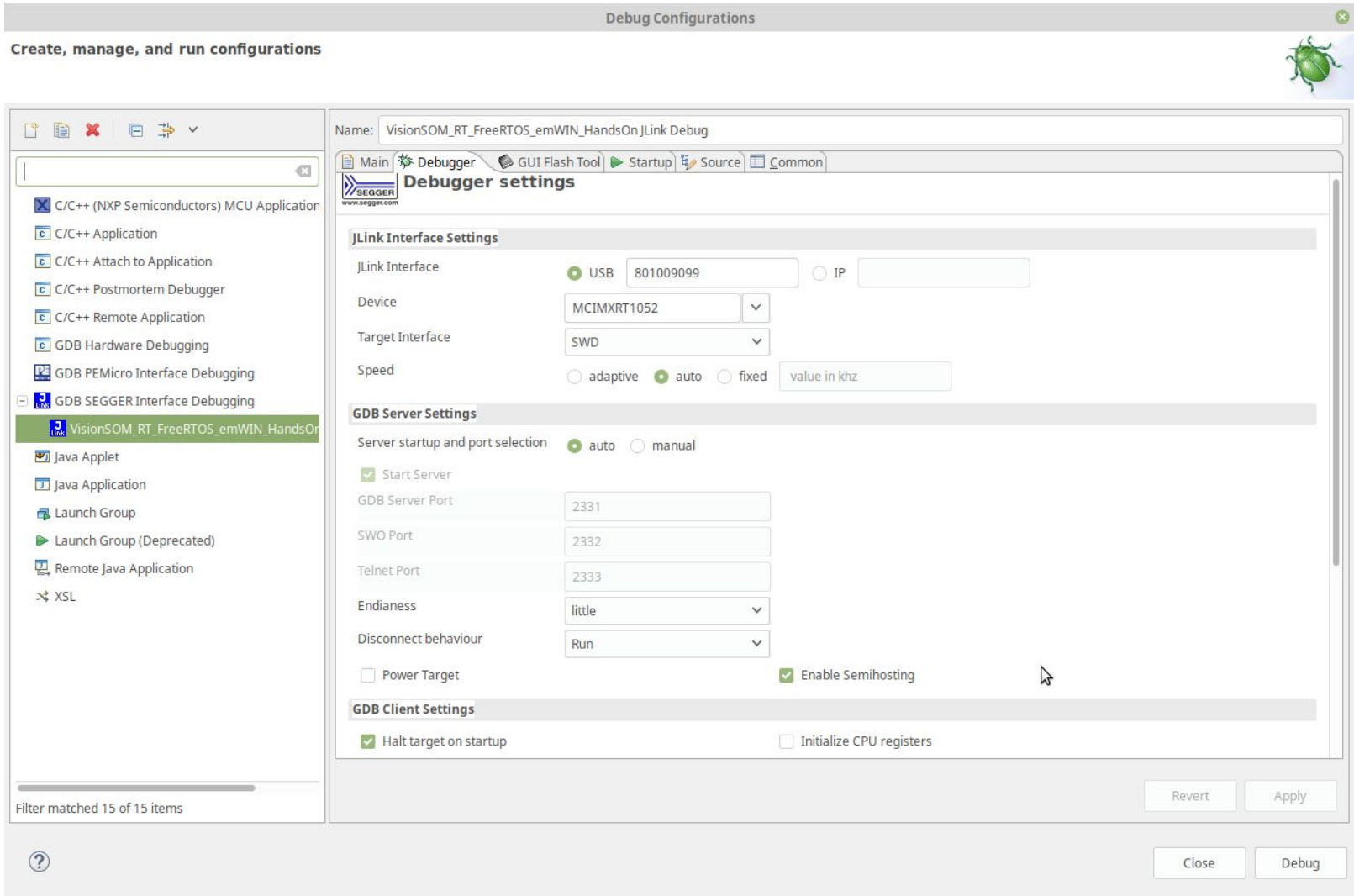
SEGGER J-Link



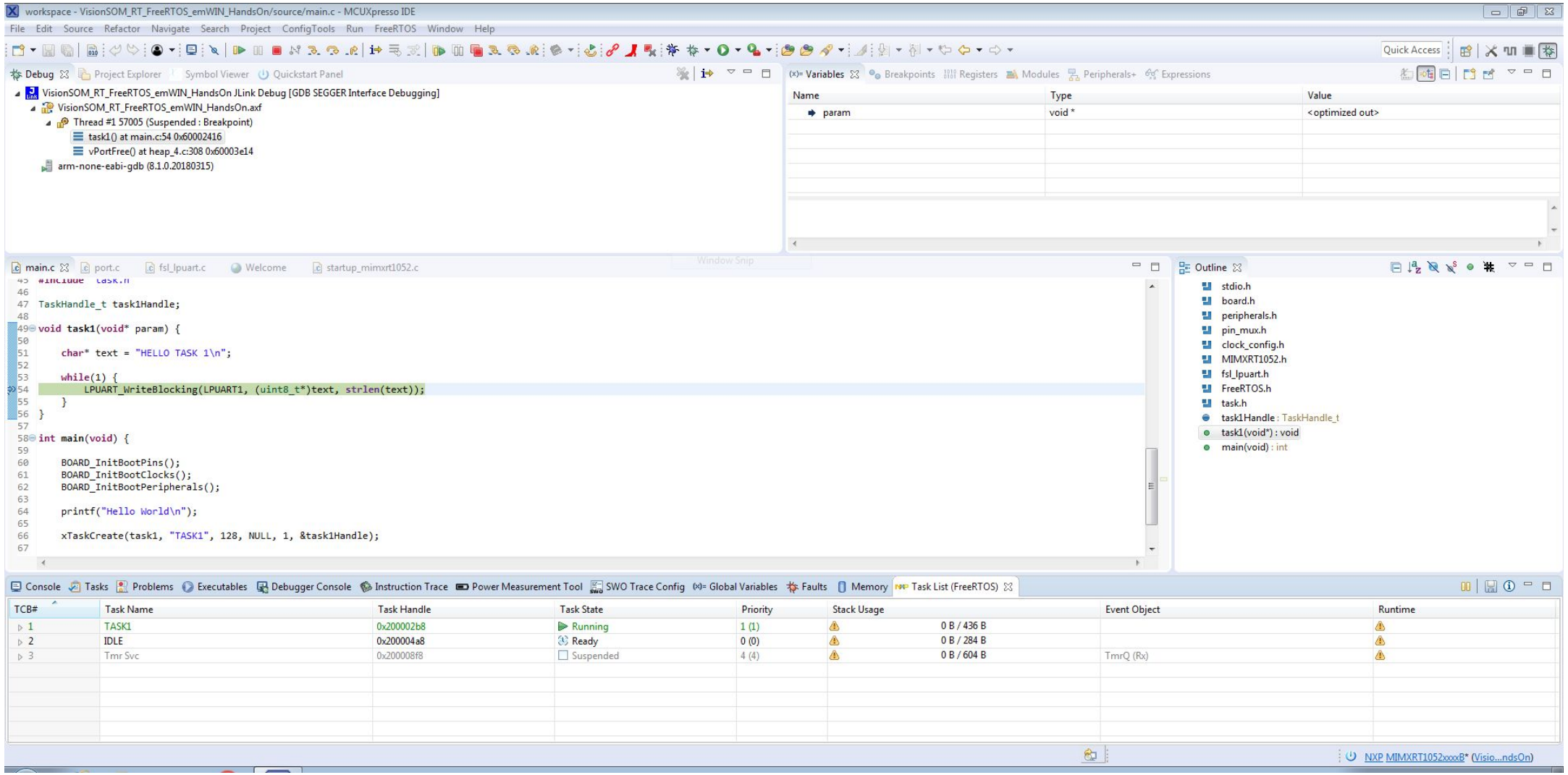
SEGGER J-Link



SEGGER J-Link



SEGGER J-Link



FreeRTOS

- RTOS (Real Time Operating System)
- RT - Ensures the compliance with time constraints
  deterministic scheduler
  fixed priorities
- OS - Kernel executing multiple tasks
  independent tasks
  synchronization and communication

FreeRTOS - tasks

RTOS approach

```
int main(void) {
  // create tasks
  // run scheduler
}

void task1(void*) {
  // initialization
  while(1) {
    // processing
  }
}
void task2(void*) {
  // initialization
  while(1) {
    // processing
  }
}
```

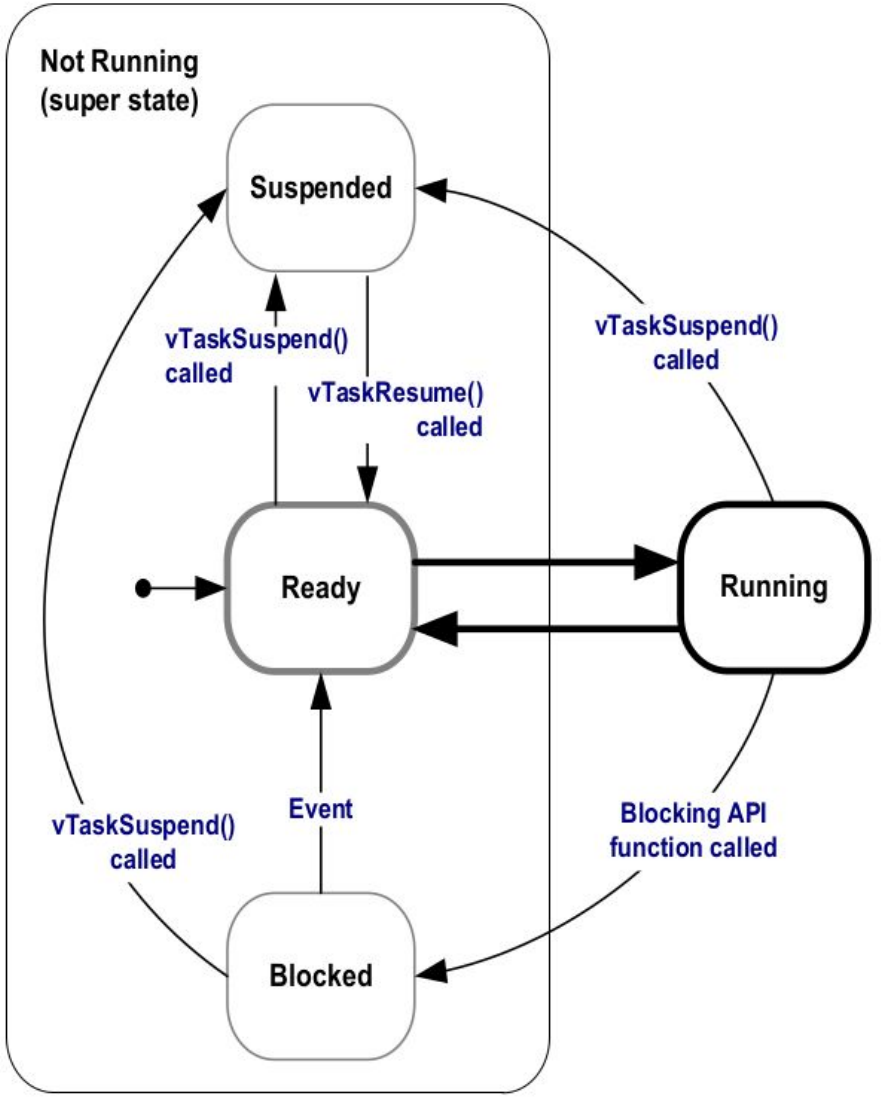
Classic approach

```
int main(void) {

  // initialization

  while(1) {
    // processing
  }
}
```

FreeRTOS - tasks



FreeRTOS - scheduling

	configUSE_TIME_SLICING = 0	configUSE_TIME_SLICING = 1
configUSE_PREEMPTION = 0	Co-operative Scheduling	
configUSE_PREEMPTION = 1	Prioritized Pre-emptive Scheduling	Prioritized Pre-emptive Scheduling with Time Slicing

FreeRTOS - memory management

- static memory allocation
- dynamic allocation algorithms
  - heap_1 - simple allocation without freeing
  - heap_2 - allocation and freeing without blocks combining
  - heap_3 - malloc() and free()
  - heap_4 - allocation and freeing with blocks combining
  - heap_5 - allocation and freeing with blocks combining with multiple regions support

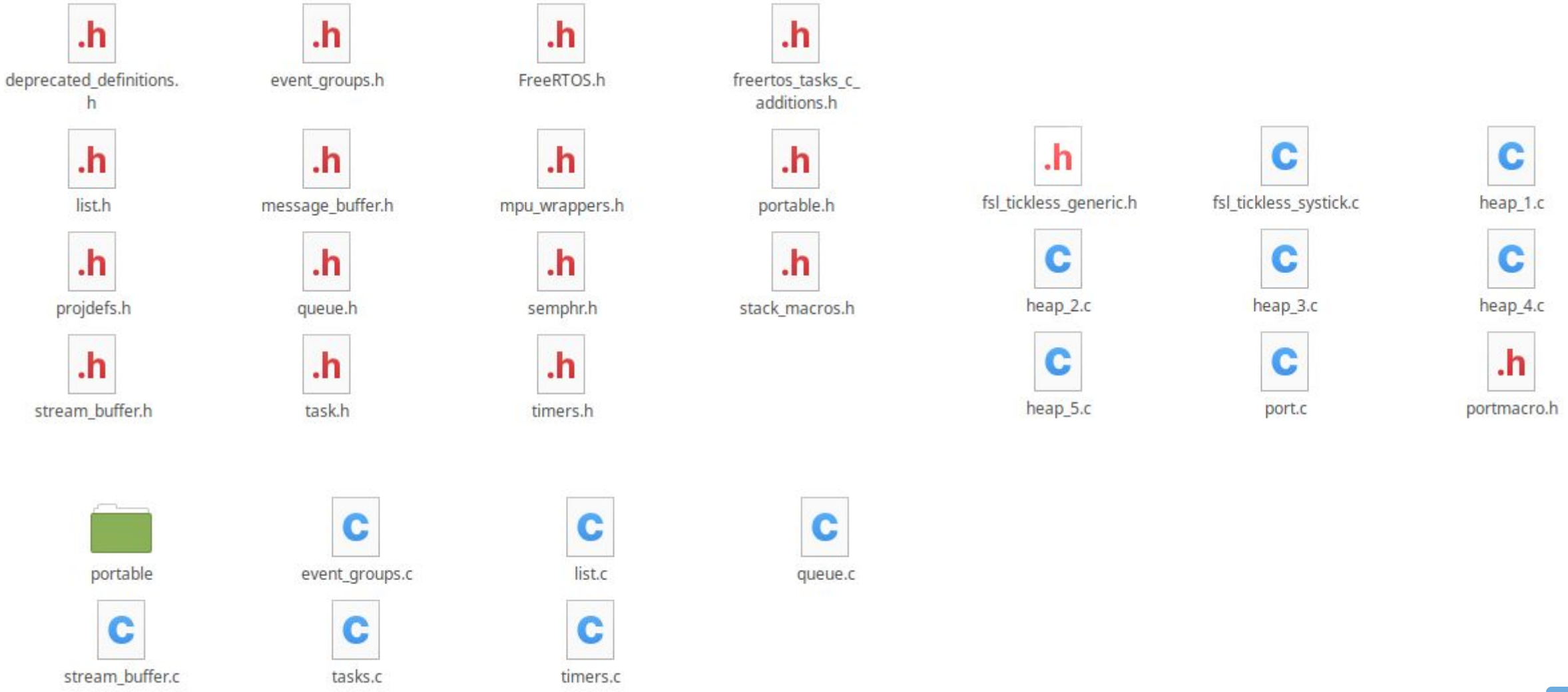
FreeRTOS - Idle task

- created automatically
- lowest priority
- responsible for freeing memory
- must not be starved!

FreeRTOS - components

- Semaphores
- Mutexes
- Queues
- Notifications
- Software timers

FreeRTOS - file structure



FreeRTOS - configuration

- FreeRTOSConfig.h

```
#define configUSE_PREEMPTION                    1
#define configUSE_TICKLESS_IDLE                 0
#define configCPU_CLOCK_HZ                      (SystemCoreClock)
#define configTICK_RATE_HZ                      ((TickType_t)200)
#define configMAX_PRIORITIES                    5
#define configMINIMAL_STACK_SIZE                ((unsigned short)90)
#define configMAX_TASK_NAME_LEN                 20
#define configUSE_16_BIT_TICKS                  0
#define configIDLE_SHOULD_YIELD                 1
#define configUSE_TASK_NOTIFICATIONS            1
#define configUSE_MUTEXES                       1
#define configUSE_RECURSIVE_MUTEXES             1
#define configUSE_COUNTING_SEMAPHORES           1
#define configUSE_ALTERNATIVE_API               0 /* Deprecated! */
#define configQUEUE_REGISTRY_SIZE               8
#define configUSE_QUEUE_SETS                    0
#define configUSE_TIME_SLICING                  1
#define configUSE_NEWLIB_REENTRANT              0
#define configENABLE_BACKWARD_COMPATIBILITY     1
#define configNUM_THREAD_LOCAL_STORAGE_POINTERS 5
#define configUSE_APPLICATION_TASK_TAG          0
```

FreeRTOS - memory overhead

- IAR STR71x ARM7 port
- Full optimisation
- Four priorities
- https://www.freertos.org/FAQMem.html

Scheduler	236 bytes
Queue	76 bytes + queue storage area
Task	64 bytes + the task stack size

\+ 5-10 kB Flash

FreeRTOS - time overhead

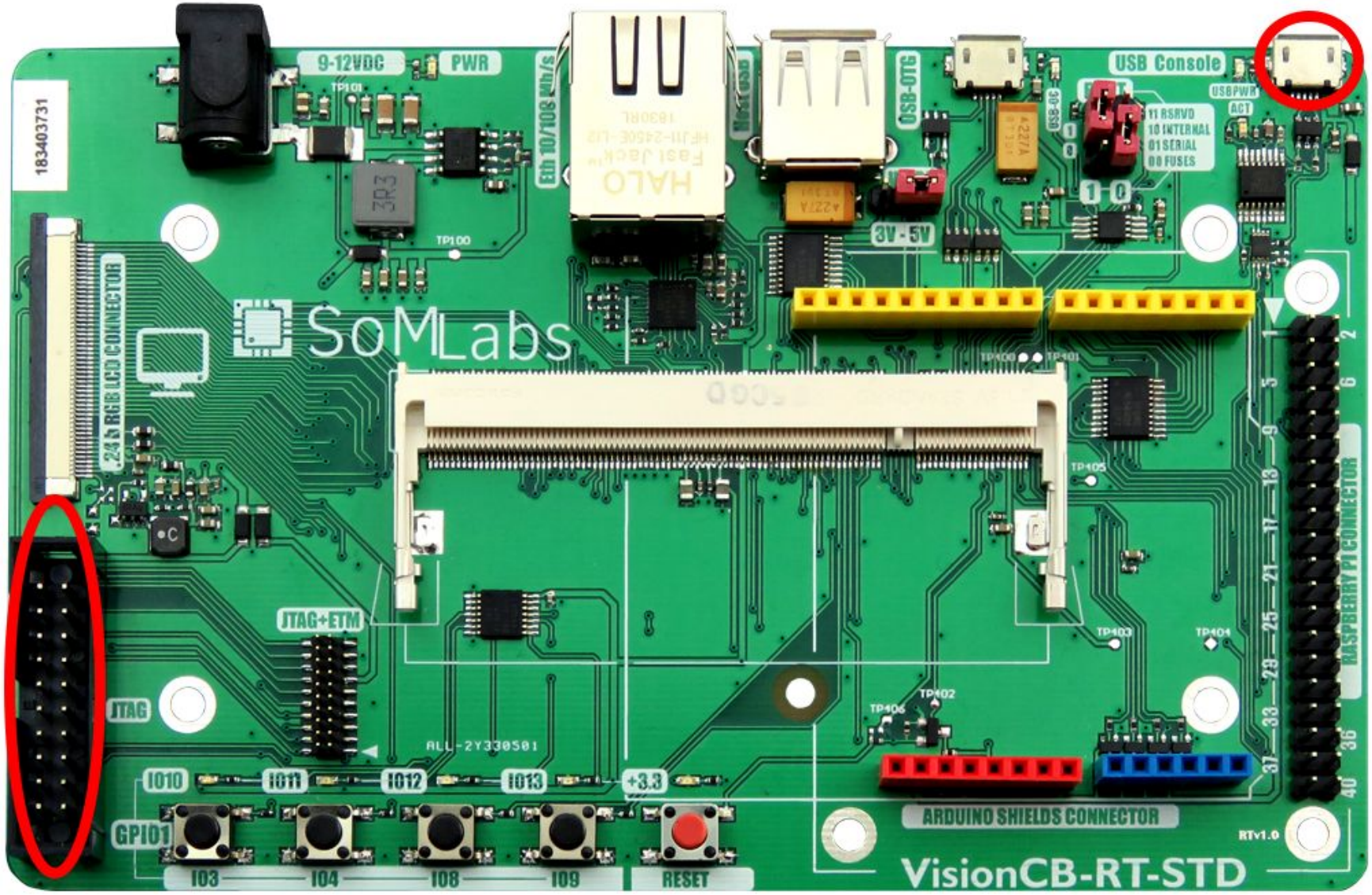
- FreeRTOS ARM Cortex-M3 port for the Keil compiler
- Stack overflow checking turned off
- Trace features turned off
- Run-time stats feature turned off
- Compiler set to optimise for speed
- https://www.freertos.org/FAQMem.html

Context switching: 84 CPU cycles

Exercise 1

- Create a new task
- Synchronize access to the common resource (UART port) using a mutex
- Use the same code for multiple tasks
- Synchronize access to the common resource (UART port) by changing the scheduling algorithm

Exercise 1


9-12VDC
PWR
USB Console
SoMLabs
3V - 5V
JTAG+ETM
JTAG
GPIO1
IO10
IO11
IO12
IO13
+3.3
IO3
IO4
IO8
IO9
RESET
ARDUINO SHIELDS CONNECTOR
RASPBERRY PI CONNECTOR
VisionCB-RT-STD

Exercise 1

- Creating a new task

```
BaseType_t xTaskCreate(  TaskFunction_t pvTaskCode,
                         const char * const pcName,
                         unsigned short usStackDepth,
                         void *pvParameters,
                         UBaseType_t uxPriority,
                         TaskHandle_t *pxCreatedTask
                       );

void vTaskCode( void * pvParameters );
```

Exercise 1

- Using the mutex

```
SemaphoreHandle_t xSemaphoreCreateMutex( void );

BaseType_t xSemaphoreTake( SemaphoreHandle_t xSemaphore,
                           TickType_t xTicksToWait );

BaseType_t xSemaphoreGive( SemaphoreHandle_t xSemaphore );
```

Exercise 1

- Changing scheduling algorithm (FreeRTOSConfig.h)

```
configUSE_PREEMPTION

configUSE_TIME_SLICING
```

FreeRTOS - interrupts

- No requirements regarding the interrupts handling
- Functions dedicated for the interrupts - **FromISR*
- Task-interrupt cooperation
  - Synchronization
  - Processing deferring

Exercise 2

- Send a notification from the button interrupt
- Add a higher priority ‘busy task’
- Observe the button-LED responsiveness
- Switch the priorities
- Change the scheduling algorithm

Exercise 2

- Using notifications

```
BaseType_t xTaskNotifyFromISR(
    TaskHandle_t xTaskToNotify,
    uint32_t ulValue,
    eNotifyAction eAction,
    BaseType_t *pxHigherPriorityTaskWoken );

BaseType_t xTaskNotifyWait(
    uint32_t ulBitsToClearOnEntry,
    uint32_t ulBitsToClearOnExit,
    uint32_t *pulNotificationValue,
    TickType_t xTicksToWait );
```

Exercise 2

- Busy task

```
void busyTask(void* param) {

    const TickType_t delayMs = 10 / portTICK_PERIOD_MS;
    char* text = "Calculation finished\n";

    while(1) {
        for(int i = 0; i < 0xFFFFFFF; i++)
            ;

        LPUART_WriteBlocking(LPUART1, (uint8_t*)text, strlen(text));
        vTaskDelay(delayMs);
    }
}


xTaskCreate(busyTask, "BUSY_TASK", 128, busyTask, 2, &busyTaskHandle);
```

Exercise 2

- Switching priorities

```
xTaskCreate(ledTask, "LED_TASK", 128, ledTask, 2, &ledTaskHandle);
xTaskCreate(busyTask, "BUSY_TASK", 128, busyTask, 1, &busyTaskHandle);
```

- Changing scheduling algorithm

```
#define configUSE_PREEMPTION    0
#define configUSE_TIME_SLICING  1
```

FreeRTOS - queues

- FIFO queues used for sending data
  - task - task
  - task - interrupt

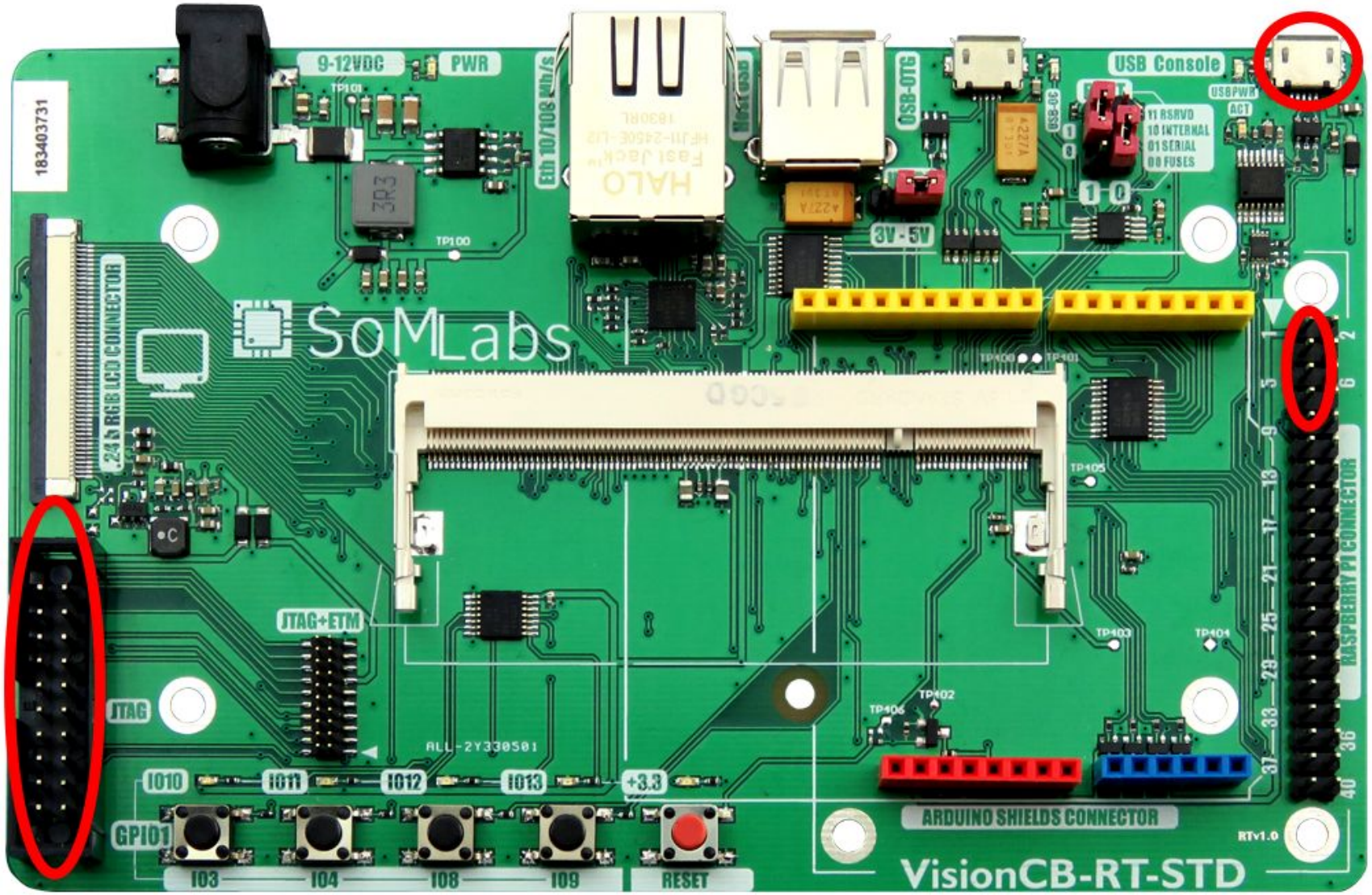
- Data type defined by the programmer:
  - simple type
  - structure
  - pointer

Exercise 3

- Read temperature and pressure from MPL3115A2
- Create a queue for sending:
  - simple types
  - structures
  - pointers

  between two tasks.

Exercise 3



PIN	
1	VCC
3	SDA
5	SCL
9	GND

Exercise 3

- Using a queue

```
QueueHandle_t xQueueCreate(  UBaseType_t uxQueueLength,
                             UBaseType_t uxItemSize  );

BaseType_t xQueueSend(  QueueHandle_t xQueue,
                        const void * pvItemToQueue,
                        TickType_t xTicksToWait  );

BaseType_t xQueueReceive(  QueueHandle_t xQueue,
                           void *pvBuffer,
                           TickType_t xTicksToWait  );
```

emWin

- GUI library based on windows and widgets
- Touch panel support
- Easy to use with any RTOS or bare-metal application
- Examples for multiple microcontrollers
  https://www.segger.com/evaluate-our-software/

emWin - configuration

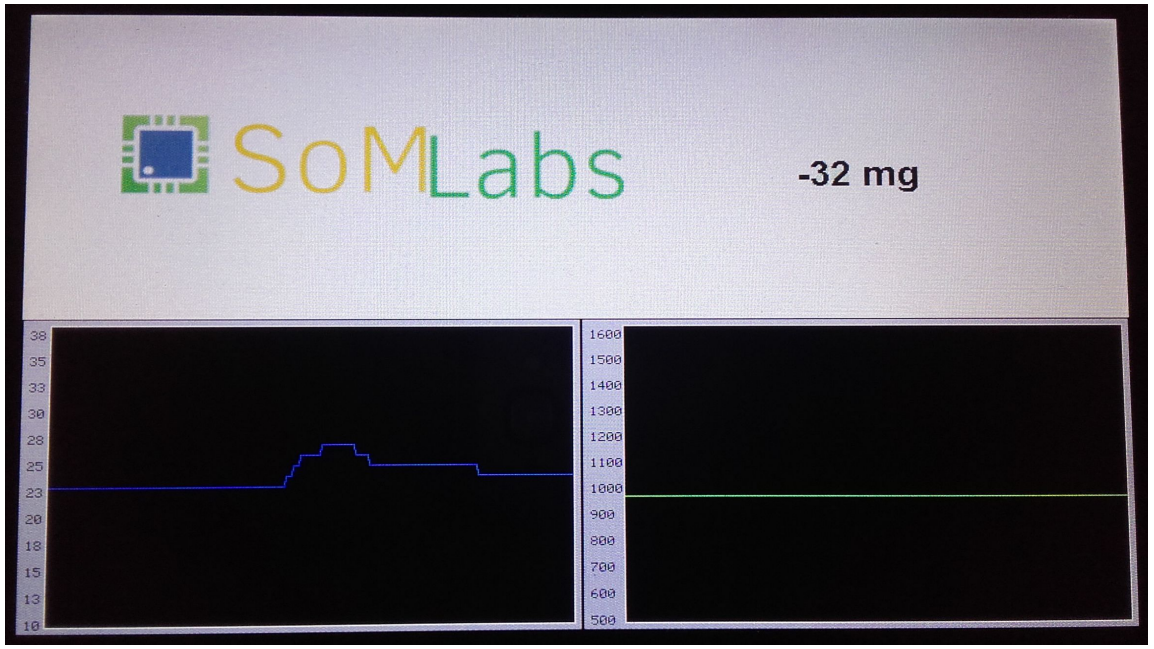
Configuration files:

- GUIConf.h
- LCDConf.h
- emwin_support.c
- emwin_support.h

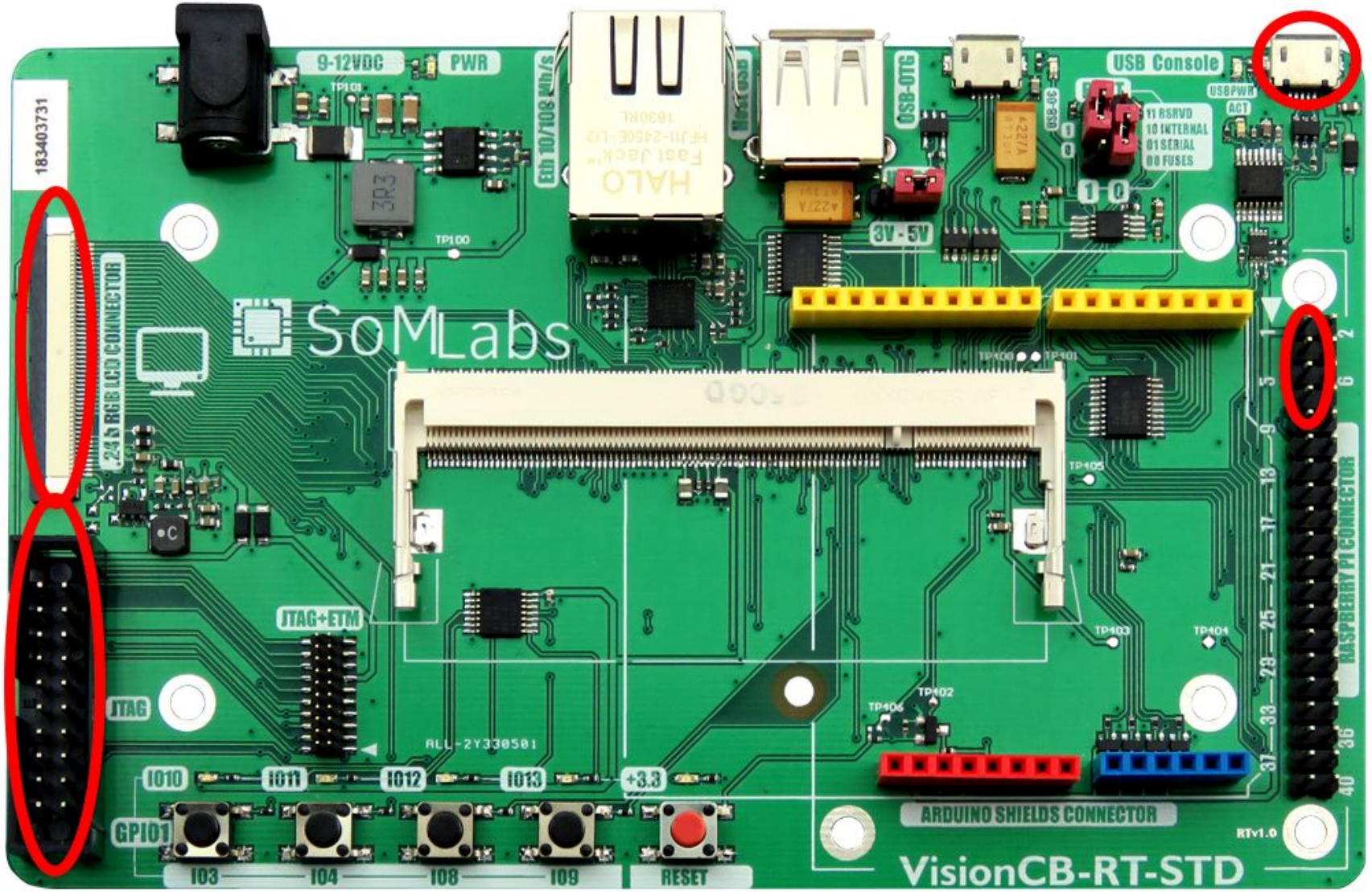
```
#define GUI_MEMORY_ADDR ((uint32_t)s_gui_memory)
#define VRAM_ADDR ((uint32_t)s_vram_buffer)
```

Exercise 4

- Analyze the GUI building code in the *guiTask*
- Create another GRAPH component for pressure data



Exercise 4



PIN	
1	VCC
3	SDA
5	SCL
9	GND

Exercise 4

- Creating the GRAPH widget

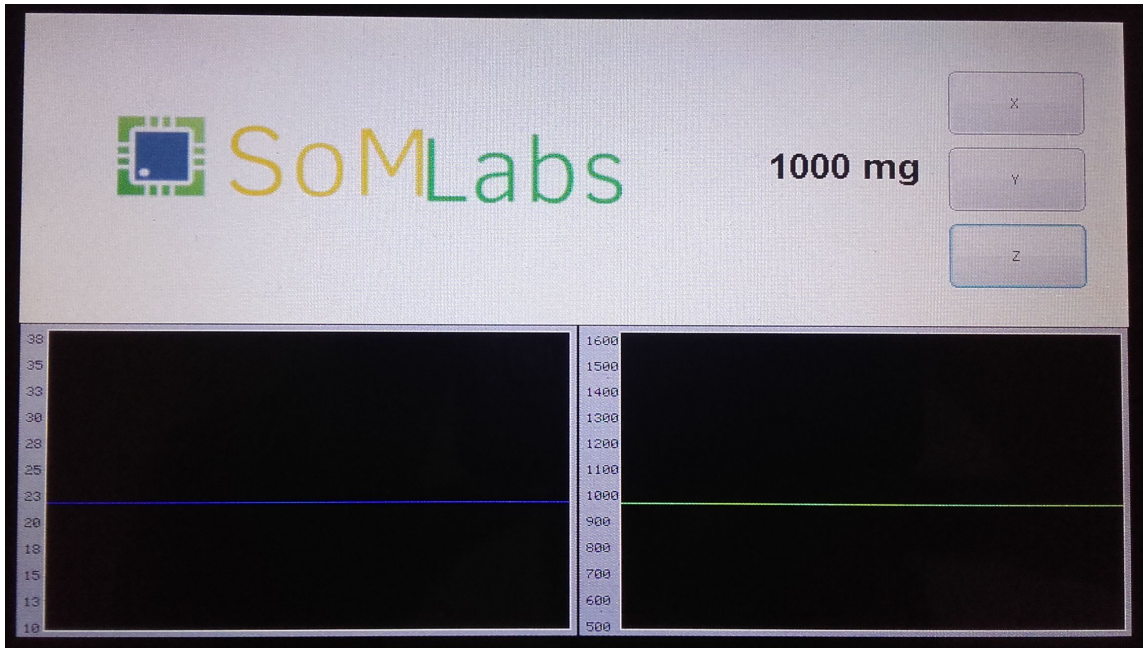
```
GRAPH_Handle GRAPH_CreateEx( int x0, int y0,
                             int xSize, int ySize,
                             WM_HWIN hParent,
                             int WinFlags,
                             int ExFlags, int Id);
```

emWin - touch panel

- PID (Pointer Input Device) and MultiTouch support
- Providing touch events to the library
- Handling the input events
- Windows notifications

Exercise 5

- Analyze handling the touch events in the software timer code
- Create buttons
- Use the buttons for acceleration axis selection
- Handle the input events



Exercise 5

- Creating buttons

```
BUTTON_Handle BUTTON_Create( int x0, int y0,
                             int xSize, int ySize,
                             WM_HWIN hParent,
                             int WinFlags,
                             int ExFlags,
                             int Id);
```

Exercise 5

- Handling events

```
void eventCallback(WM_MESSAGE * pMsg) {
  if(pMsg->MsgId == WM_NOTIFY_PARENT) {
    if (pMsg->hWinSrc == buttonX) {
      int code = pMsg->Data.v;
      if (code == WM_NOTIFICATION_RELEASED) {
        activeAxis = ACTIVE_AXIS_X;
      }
    }
    return;
  }
```

Why to use RTOS?

- Easy timing management
- Application modularity
- Team development
- Schedulability design and analysis
  - MAST (mast.unican.es)
  - SimSo (projects.laas.fr/simso)

Why to use FreeRTOS?

- One of the most popular RTOS
- Very well documented
- Open-source
- FreeRTOS+ Ecosystem
- OpenRTOS an SafeRTOS

- Connecting the Future
- Customer Centric