



FreeRTOS and emWin for i.MX RT platform

Krzysztof Chojnowski



SOM

System on Module

CB

Carrier Board

DK

Development Kit

Engineering

Since 2003 delivering proven designs

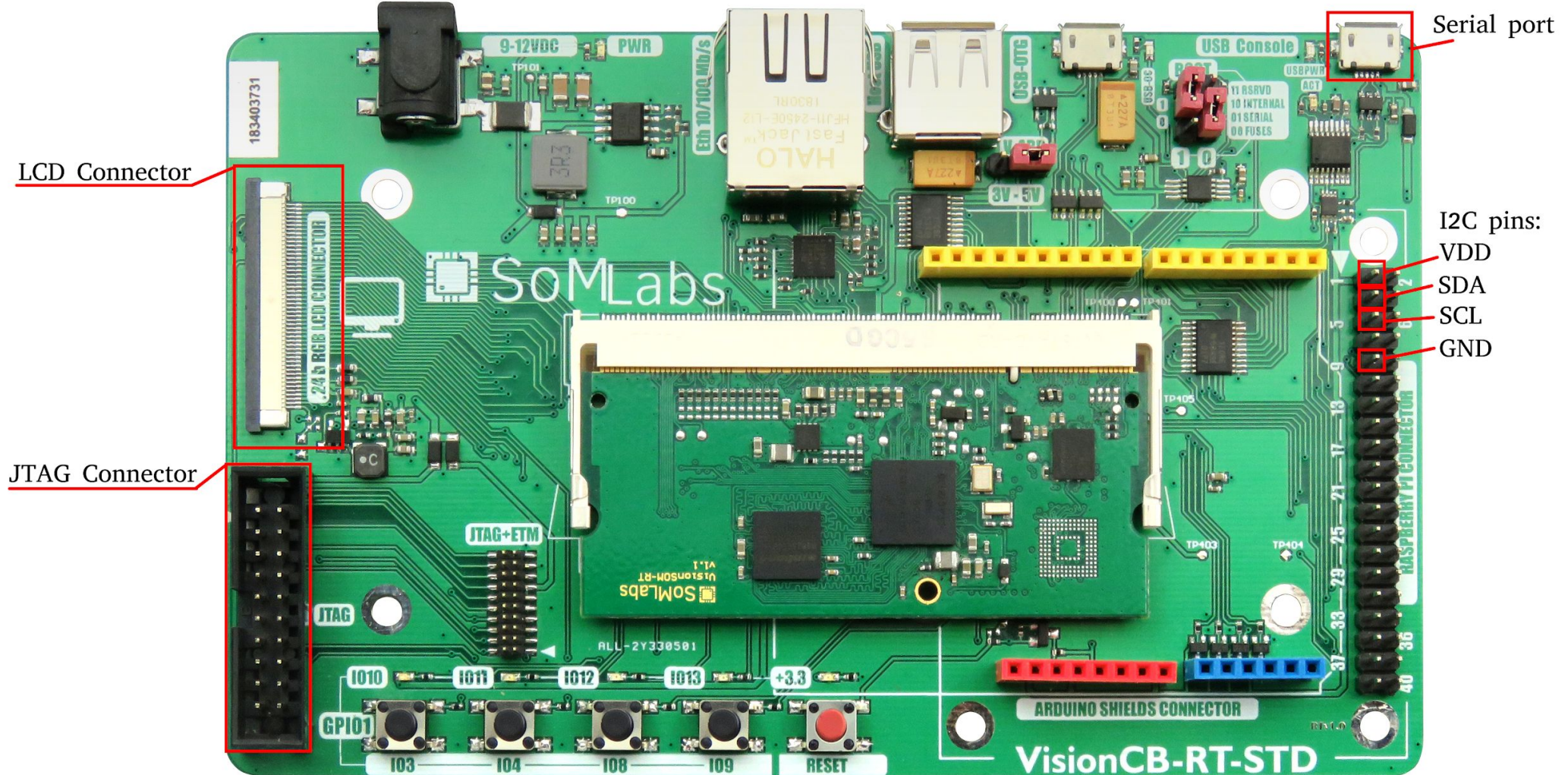
Agenda

- 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 & Features](#)

[Supported Devices](#)

[Target Applications](#)

[System Requirements](#)

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.

[More ▾](#)

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

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



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

i 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

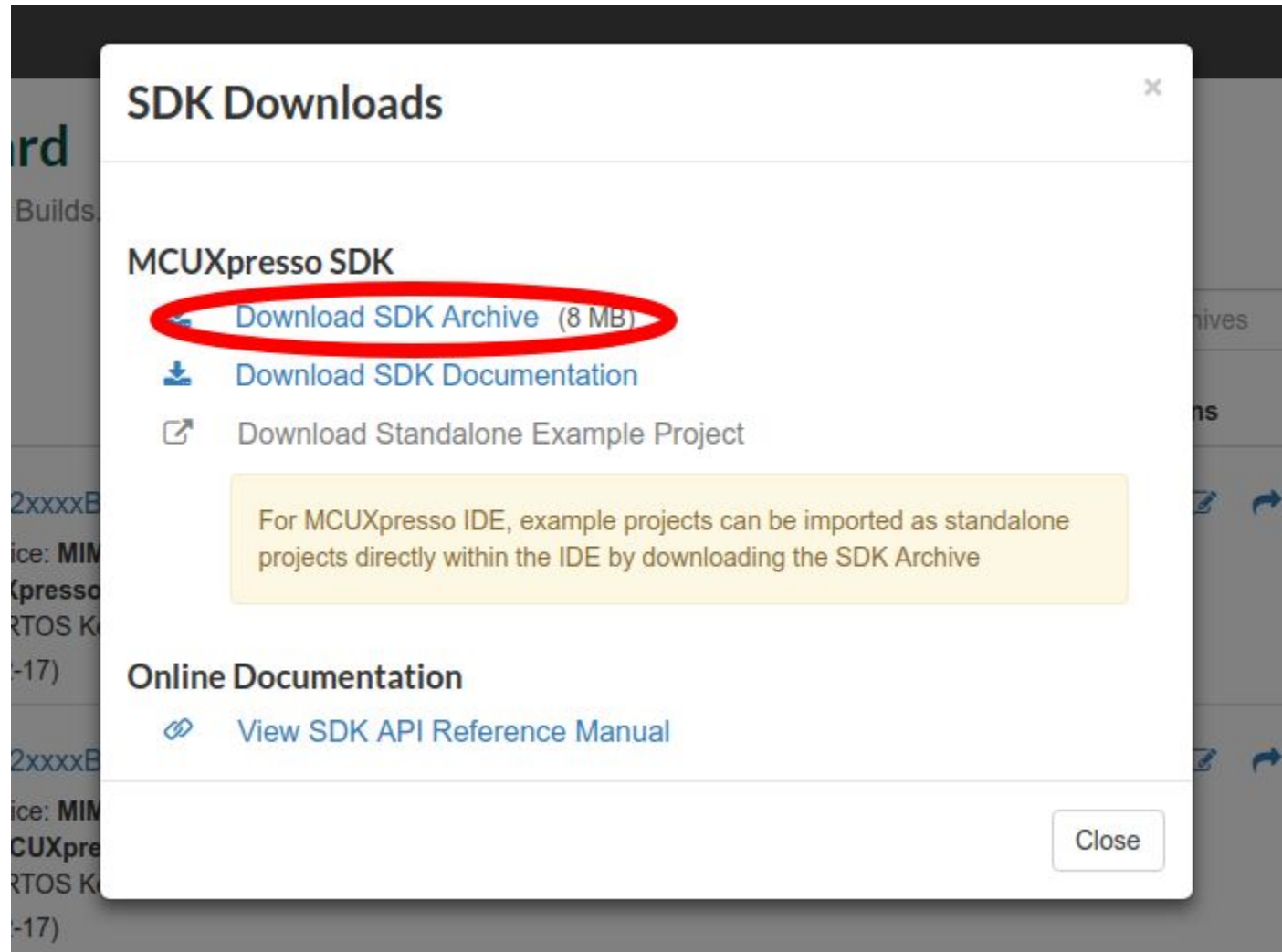
Access, Download, and Share your requested SDK Builds.

My Recent SDKs

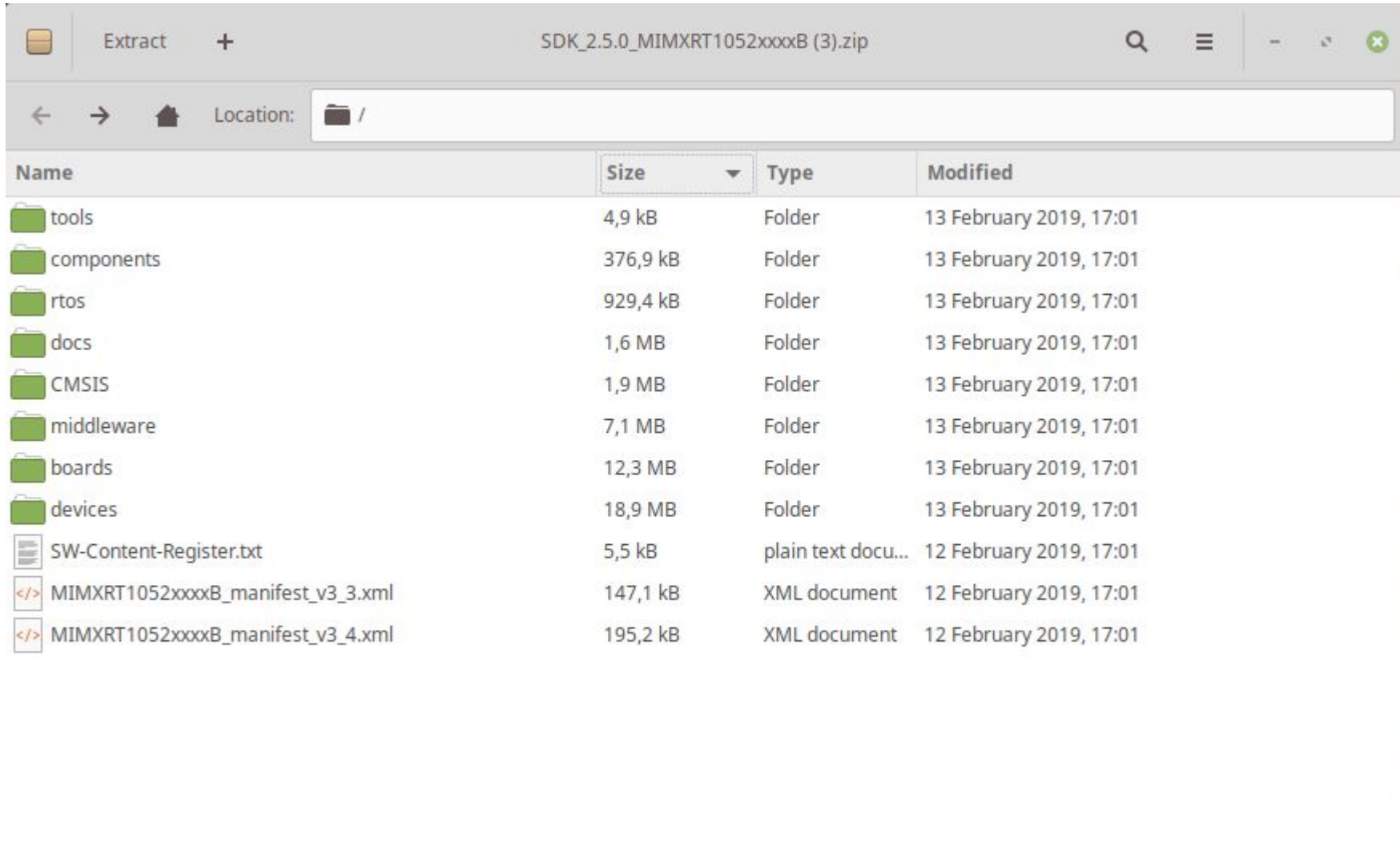
Q

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

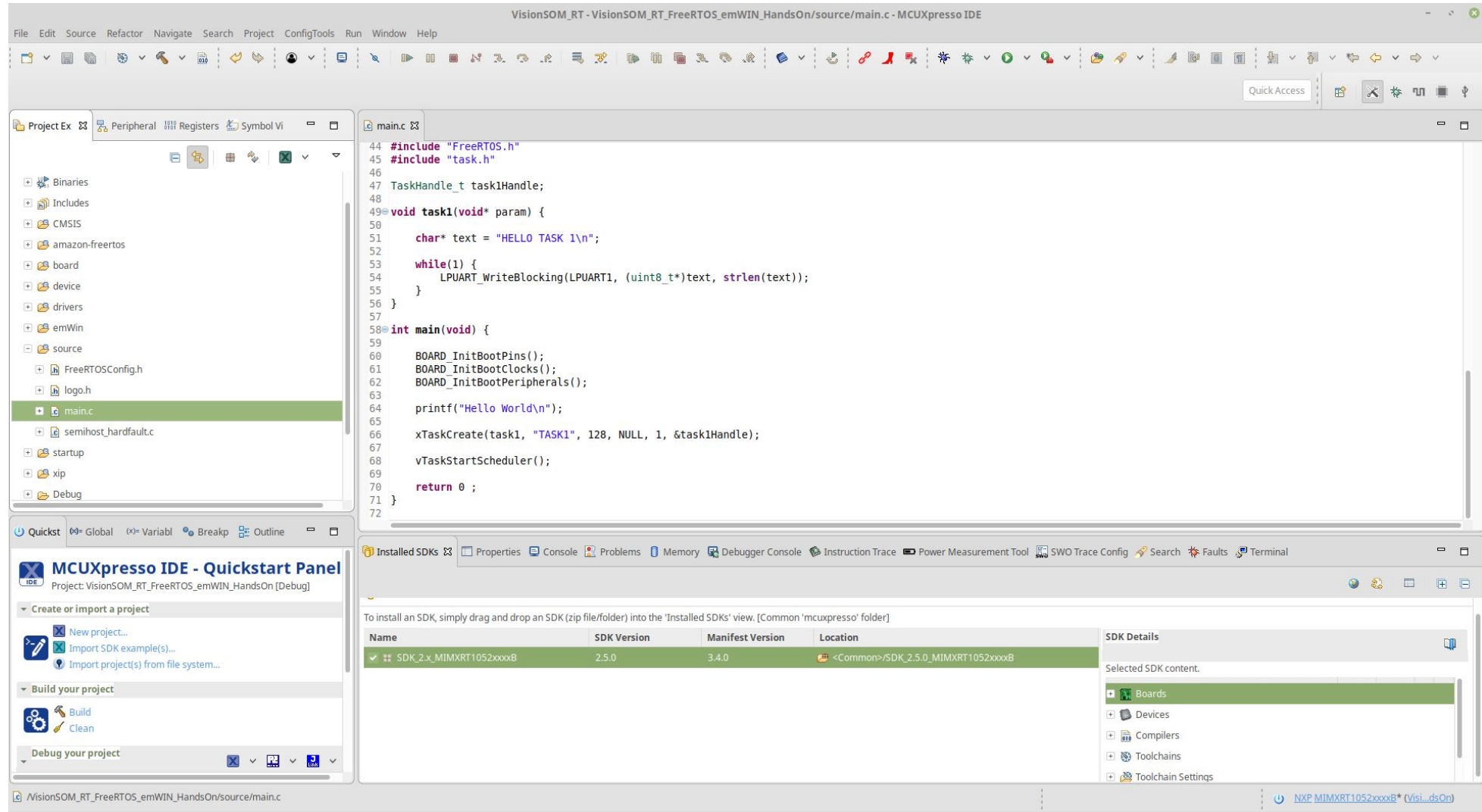


SDK

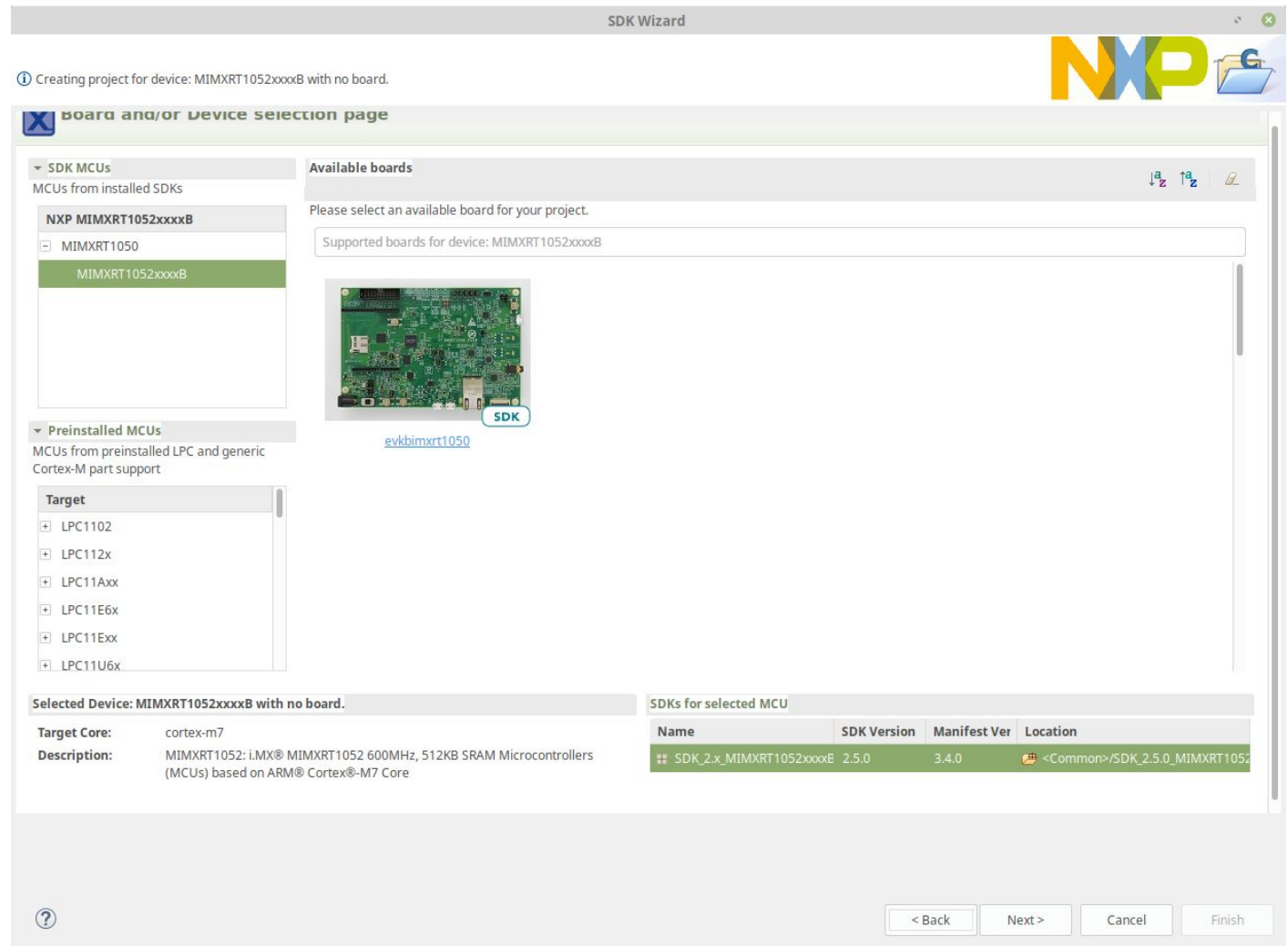
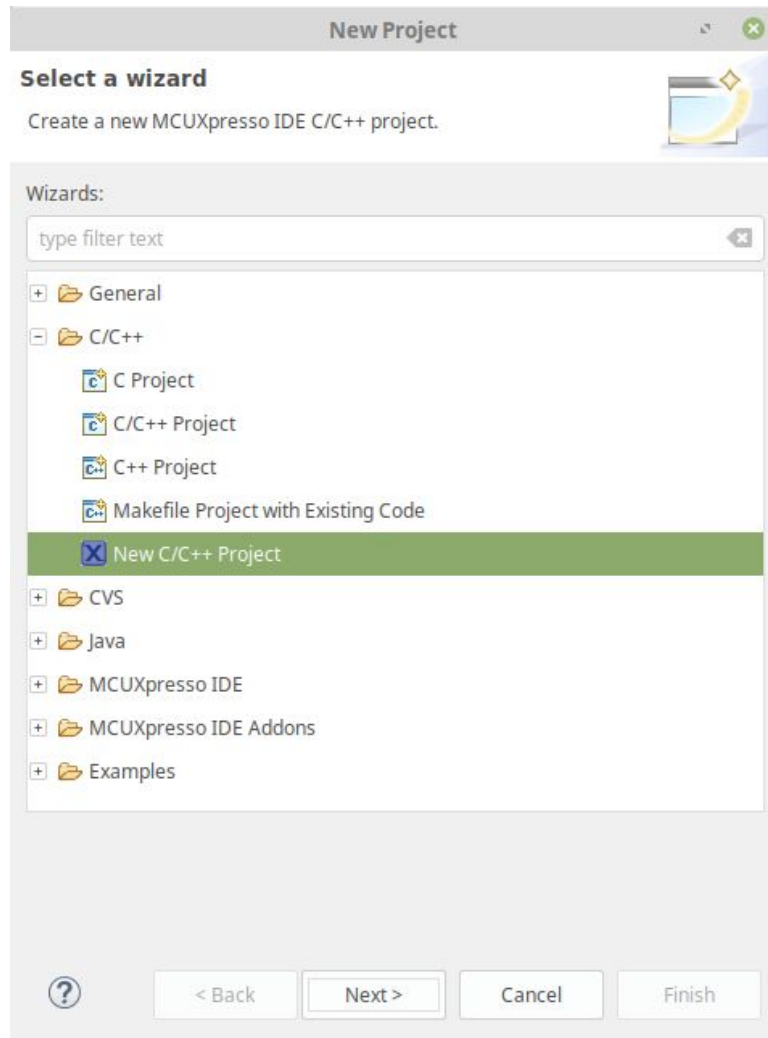


Name	Size	Type	Modified
tools	4,9 kB	Folder	13 February 2019, 17:01
components	376,9 kB	Folder	13 February 2019, 17:01
rtos	929,4 kB	Folder	13 February 2019, 17:01
docs	1,6 MB	Folder	13 February 2019, 17:01
CMSIS	1,9 MB	Folder	13 February 2019, 17:01
middleware	7,1 MB	Folder	13 February 2019, 17:01
boards	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



MCUXpresso

SDK Wizard

Configure the project

Project name: MIMXRT1052xxxxB_Project Project name suffix:

☒ Use default location

Location: /home/krzysiek/workspace/VisionSOM_RT/MIMXRT1052xxxxB_Project

Device Packages

- ☒ MIMXRT1052CVLSB
- ☐ MIMXRT1052CVJ5B

Board

No board selected

Project Type

- ☒ C Project ☐ C++ Project
- ☐ C Static Library ☐ C++ Static Library

Project Options

- SDK Debug Console ☒ Semihost ☐
- ☒ CMSIS-Core
- ☒ Copy sources
- ☒ Import other files

OS

Name	Version
☐ Amazon-FreeRTOS	10.0.1
☒ baremetal	1.0.0

driver

Name	Version
☐ adc_12b1mpps_sa	2.0.1
☐ adc_etc	2.0.1
☐ aipstz	2.0.0
☐ aoi	2.0.0
☐ bee	2.0.0
☐ cache	2.0.1
☐ clock	2.1.0
☐ cmp	2.0.0

CMSIS_driver

Name	Version
☒ pi2c_cmsis	2.0.0
☐ lpspi_cmsis	2.0.0
☐ lpuart_cmsis	2.0.1

utilities

Name	Version
☒ assert	1.0.0
☐ debug_console	1.0.0
☐ lpuart_adapter	1.0.0
☐ misc_utilities	1.0.0
☐ notifier	1.0.0
☐ serial_manager	1.0.0
☐ serial_manager_uai	1.0.0
☐ shell	1.0.0

middleware

Name
☐ Image

< Back Next > Cancel Finish

MCUXpresso

SDK Wizard



Advanced project settings

C/C++ Library Settings

Set library type (and hosting variant) Redlib (semihost-nf)

☐ Redlib: Use floating point version of printf

☐ Redlib: Use character rather than string based printf

☒ Redirect SDK "PRINTF" to C library "printf"

☒ Include semihost HardFault handler

☐ NewlibNano: Use floating point version of printf

☐ NewlibNano: Use floating point version of scanf

☐ Redirect printf/scanf to ITM

☐ Redirect printf/scanf to UART

Hardware settings

Set Floating Point type FPU5-SP-D16 (HardABI)

MCU C Compiler

Language standard Compiler default

MCU Linker

☐ Link application to RAM

Memory Configuration

Memory details

Default LinkServer Flash Driver Browse...

Type	Name	Alias	Location	Size	Driver
RAM	SRAM_DTC	RAM	0x20000000	0x20000	
RAM	SRAM_ITC	RAM2	0x0	0x20000	
RAM	SRAM_OC	RAM3	0x20200000	0x40000	

?

< Back

Next >

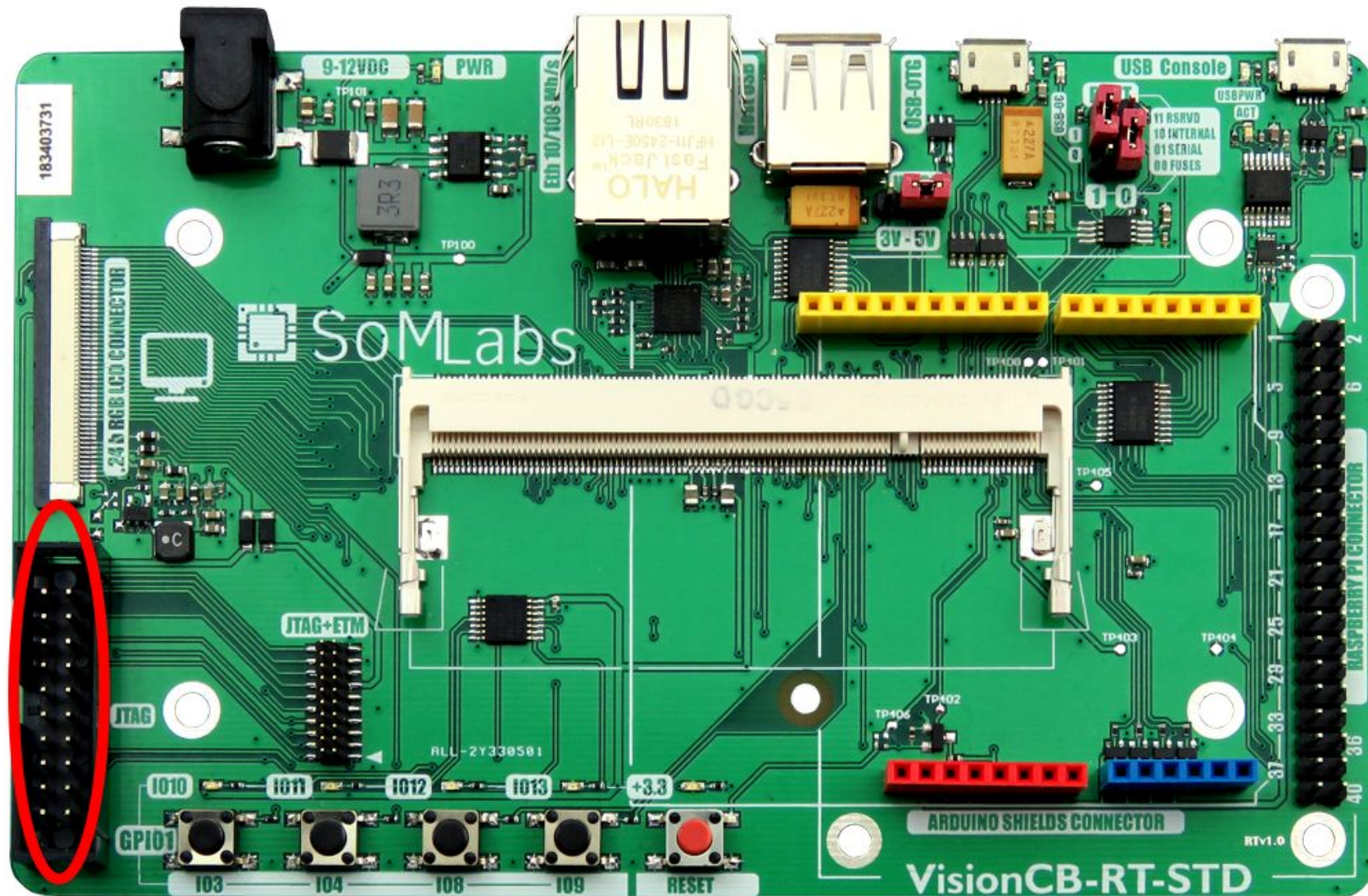
Cancel

Finish

Exercise 0

- Import SDK
- Import a template project
- build project

Exercise 0



MCUXpresso

MCUXpresso IDE interface showing the VisionSOM_RT_freertos_emWIN_HandsOn project. The main window displays the Clocks Diagram, illustrating the system clock configuration. The diagram shows the flow from crystal oscillators (RTC_XTALI, XTALI) through various PLLs (PLL1, PLL2, PLL3) and dividers to generate clocks for different system components like ARM, PERIPH, AHB, IPG, PIT, GPT, USDHC, SEMC, FLEXSPI, and CSI.

The right sidebar shows the Register view, displaying a list of registers and their values. The registers are filtered by the search term "all".

Reg. Name	Set Value	Reset Value	Value Description
CCM_ANALOG_MISC1	0x00000000	0x00000000	
CCM_ANALOG_MISC2	0x00272727	0x00272727	
CCM_ANALOG_PFD_48	0x1311100c	0x1311100c	
CCM_ANALOG_PFD_52	0x101d101b	0x1018101b	
CCM_ANALOG_PLL_AR	0x00002064	0x00013063	
CCM_ANALOG_PLL_AU	0x0001302f	0x00011006	
CCM_ANALOG_PLL_AU	0x00000032	0x2964619c	
CCM_ANALOG_PLL_AU	0x00000001	0x05f5e100	
CCM_ANALOG_PLL_EN	0x00212001	0x00011001	
CCM_ANALOG_PLL_SY	0x00002001	0x00013001	
CCM_ANALOG_PLL_SY	0x00000001	0x00000012	
CCM_ANALOG_PLL_SY	0x00000000	0x00000000	
CCM_ANALOG_PLL_US	0x00000300	0x00012000	
CCM_ANALOG_PLL_US	0x00012000	0x00012000	
CCM_ANALOG_PLL_VIC	0x00008201c	0x0001100c	
CCM_ANALOG_PLL_VIC	0x00000001	0x10a24447	
CCM_ANALOG_PLL_VIC	0x00000000	0x05f5e100	
CCM_CACRR	0x00000001	0x00000001	
CCM_CBCDR	0x000098340	0x000a8300	

The bottom status bar shows the project path: VisionSOM_RT_freertos_emWIN_HandsOn.

MCUXpresso

MCUXpresso IDE interface showing project configuration and pin management.

Project: VisionSOM_RT_temp - VisionSOM_RT_FreeRTOS_emWIN_HandsOn/source/main.c - MCUXpresso IDE

Functional Group: BOARD_InitPins

Package [Pins Bottom]

Overview | Code Preview | Registers

Registers table:

Reg. Name	Set Value	Reset Value	Value Description
CCM_CCGR2	0xfc3ffff	0xfc3ffff	
CCM_CCGR4	0xffffffff	0xffffffff	
CCM_CCGR5	0xffffffff	0xffffffff	
DMAMUX_CHCFG0	0x00000000	0x00000000	
DMAMUX_CHCFG1	0x00000000	0x00000000	
DMAMUX_CHCFG2	0x00000000	0x00000000	
DMAMUX_CHCFG3	0x00000000	0x00000000	
DMAMUX_CHCFG4	0x00000000	0x00000000	
DMAMUX_CHCFG5	0x00000000	0x00000000	
DMAMUX_CHCFG6	0x00000000	0x00000000	
DMAMUX_CHCFG7	0x00000000	0x00000000	
DMAMUX_CHCFG8	0x00000000	0x00000000	
DMAMUX_CHCFG9	0x00000000	0x00000000	
DMAMUX_CHCFG10	0x00000000	0x00000000	
DMAMUX_CHCFG11	0x00000000	0x00000000	
DMAMUX_CHCFG12	0x00000000	0x00000000	
DMAMUX_CHCFG13	0x00000000	0x00000000	
DMAMUX_CHCFG14	0x00000000	0x00000000	
DMAMUX_CHCFG15	0x00000000	0x00000000	

Problems table:

Level	Issue	Origin
Warning	Peripheral LCDIF is not initialized	Pins:BOARD_InitPins

Routed Pins table:

#	Peripheral	Signal	Route to	Label	Identifier	Power group	Direction	GPIO initia	GPIO inter	Software Ir	Hysteresis	Pull Up/Do	Pull/Keep	Pull/Keep	Open drain	Speed
L14	LPUART1	TX	GPIO_AD_B0_12		n/a	NVCC_GPIO1	Output	n/a	n/a	Disabled	Disable	100K Ohm Pi	Keeper	Enabled	Disabled	medium(100 R
F13	LPUART1	RX	GPIO_AD_B0_13		n/a	NVCC_GPIO1	Input	n/a	n/a	Disabled	Disable	100K Ohm Pi	Keeper	Enabled	Disabled	medium(100 R
G11	GPIO1	gpio_io, 08	GPIO_AD_B0_08	gpioLed	gpioLed	NVCC_GPIO1	Output	Logical 0	n/a	Disabled	Disable	100K Ohm Pi	Pull	Enabled	Disabled	medium(100 R
D13	GPIO1	gpio_io, 03	GPIO_AD_B0_03	gpioButton	gpioButton	NVCC_GPIO1	Input	n/a	Falling Edge	Disabled	Disable	100K Ohm Pi	Pull	Enabled	Disabled	medium(100 R
	FLEXIO2	IO, 28	GPIO_B1_12		n/a	NVCC_GPIO1	Not Specified	n/a	n/a	Disabled	Disable	100K Ohm Pi	Keeper	Enabled	Disabled	medium(100 R

MCUXpresso

MCUXpresso IDE interface showing the configuration of the Low Power Universal Asynchronous Receiver/Transmitter (LPUART).

The interface includes a menu bar (File, Edit, Source, Refactor, Navigate, Search, Project, ConfigTools, Peripherals, Run, Window, Help), a toolbar, and a main workspace divided into several panels.

Peripherals Panel: Lists various peripherals. LPUART1 is selected and highlighted in green.

Peripheral	Used in
☐ LCDIF	eLCDIF_1 [disabled]
☐ LPI2C1	
☐ LPI2C2	
☐ LPI2C3	
☐ LPI2C4	
☐ LPSPI1	
☐ LPSPI2	
☐ LPSPI3	
☐ LPSPI4	
☒ LPUART1	LPUART_1
☐ LPUART2	
☐ LPUART3	
☐ LPUART4	
☐ LPUART5	
☐ LPUART6	
☐ LPUART7	
☐ LPUART8	
☐ PIT	
☐ RTWDOG	
☐ SAI1	
☐ SAI2	
☐ SAI3	
☐ SNVS	
☐ SPDIF	
☐ TEMPMON	
☐ TMR1	
☐ TMR2	
☐ TMR3	
☐ TMR4	

Low Power Universal Asynchronous Receiver/Transmitter (LPUART) Configuration:

- Name: LPUART_1
- Mode: Transfer
- Peripheral: LPUART1
- Preset: Custom...

LPUART general configuration:

- Clock source: UART_CLK_ROOT - BOARD_BootClockRUN: 80 MHz
- Clock source frequency: 80 MHz (BOARD_BootClockRUN)
- LPUART baud rate: 115200
- Parity mode: Parity disabled
- Data size: Eight data bit
- Enable MSB data bits order: ☐
- Number of stop bits: One stop bit
- TX FIFO watermark: 0
- RX FIFO watermark: 1
- RX RTS enable: ☐
- TX CTS enable: ☐
- TX CTS source: LPUART CTS pin
- TX CTS configuration: Sampled at the start
- RX IDLE type: Start counting after a valid start bit
- RX IDLE configuration: 1 idle character
- Enable TX: ☒
- Enable RX: ☐

Code Preview: Shows the initialization code for LPUART1 in the file `peripherals.c`.

```
/* BE CAREFUL MODIFYING THIS COMMENT - IT IS YAML SETTINGS */
/* clang-format on */
/* FlexIO peripheral configuration */
FLEXIO_I2C_Type FlexIO_I2C_1_peripheralConfig = {
    .flexioBase = FLEXIO_I2C_1_PERIPHERAL,
    .SDAPinIndex = 29,
    .SCLPinIndex = 28,
    .shifterIndex = {0, 1},
    .timerIndex = {0, 1}
};

/* I2C master configuration */
flexio_i2c_master_config_t FlexIO_I2C_1_config = {
    .enableMaster = true,
    .enableInDoze = false,
    .enableInDebug = true,
    .enableFastAccess = false,
    .baudRate_Bps = 100000
};

void FlexIO_I2C_1_init(void) {
    /* Master initialization */
    FLEXIO_I2C_MasterInit(&FlexIO_I2C_1_peripheralConfig, &
}

/* Initialization functions
*****
void BOARD_InitPeripherals(void)
{
    /* Initialize components */
    LPUART_1_init();
    GPIO_1_init();
    FlexIO_I2C_1_init();
}
```

Problems Panel: Shows warnings related to GPIO interrupt configuration and LCDIF initialization.

Level	Issue	Origin
Warning	When GPIO1 interrupt 0-15 is disabled	Peripherals:BOARD_InitPeriph
Warning	When GPIO1 interrupt 16-31 is disabled	Peripherals:BOARD_InitPeriph
Warning	Peripheral LCDIF is not initialized	Pins:BOARD_InitPins

Status Bar: Displays warnings about GPIO interrupt settings.

WARNINGS: VisionSOM_RT_Freertos_emWIN_HandsOn: - See setting: (GPIO_1 / Interrupts / Enable combined interrupt 0-15 request) - See setting: (GPIO_1 / Interrupts / Enable combined interrupt 16-31 request)

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) -->  
<!--          -->
```

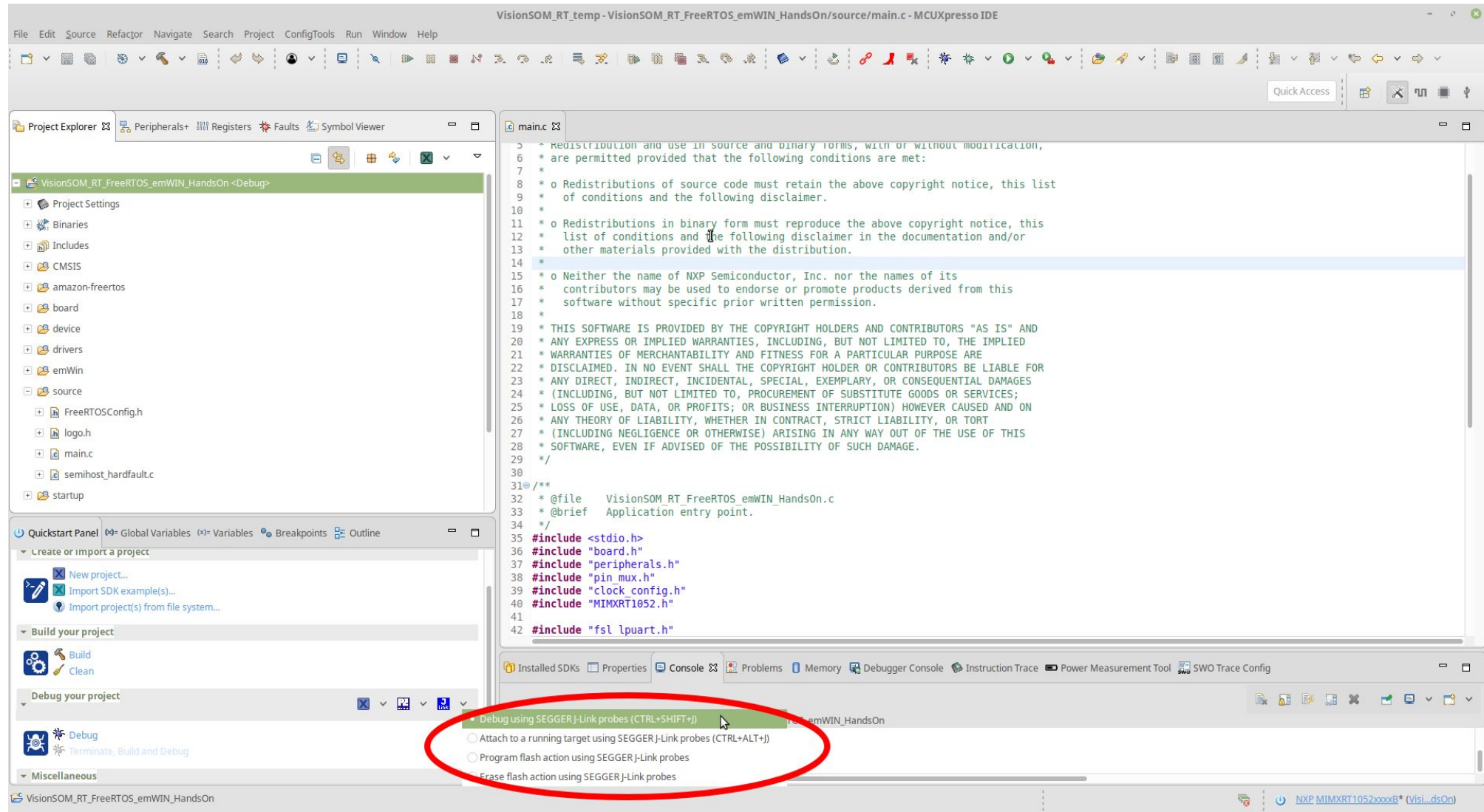
- Loader value

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

should be:

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

SEGGER J-Link



SEGGER J-Link

Probes discovered

Connect to target: MIMXRT1052xxxxB

1 probe found. Select the probe to use:

Available attached probes

	Name	Serial number/I	Type	Manuf	IDE Debug Mode
	J-Link	801009099	USB	SEGGER	All-Stop

Supported Probes (tick/untick to enable/disable)

☒ SEGGER J-Link probes

Probe search options

Search again

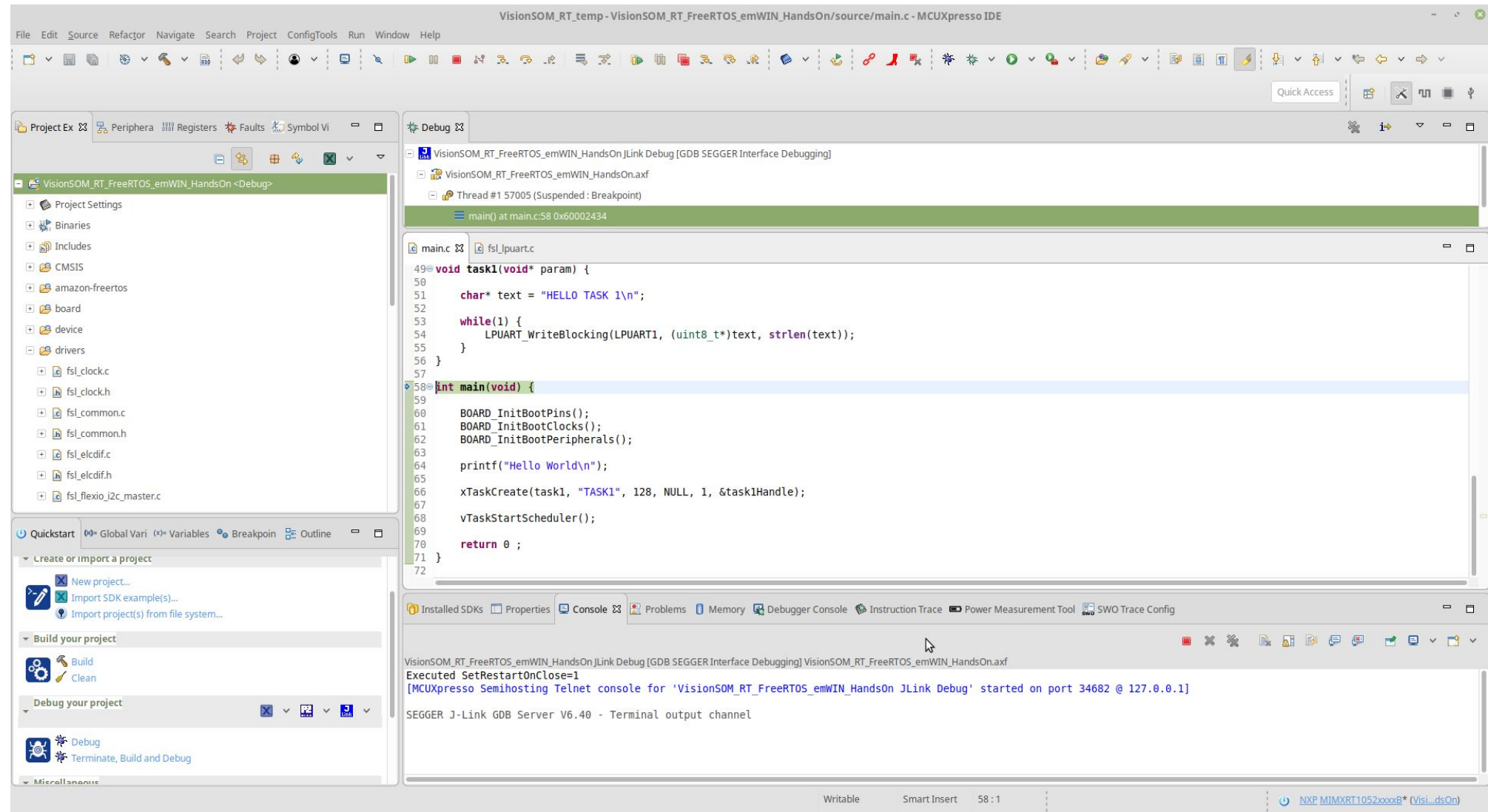
☒ Remember my selection (for this Launch configuration)



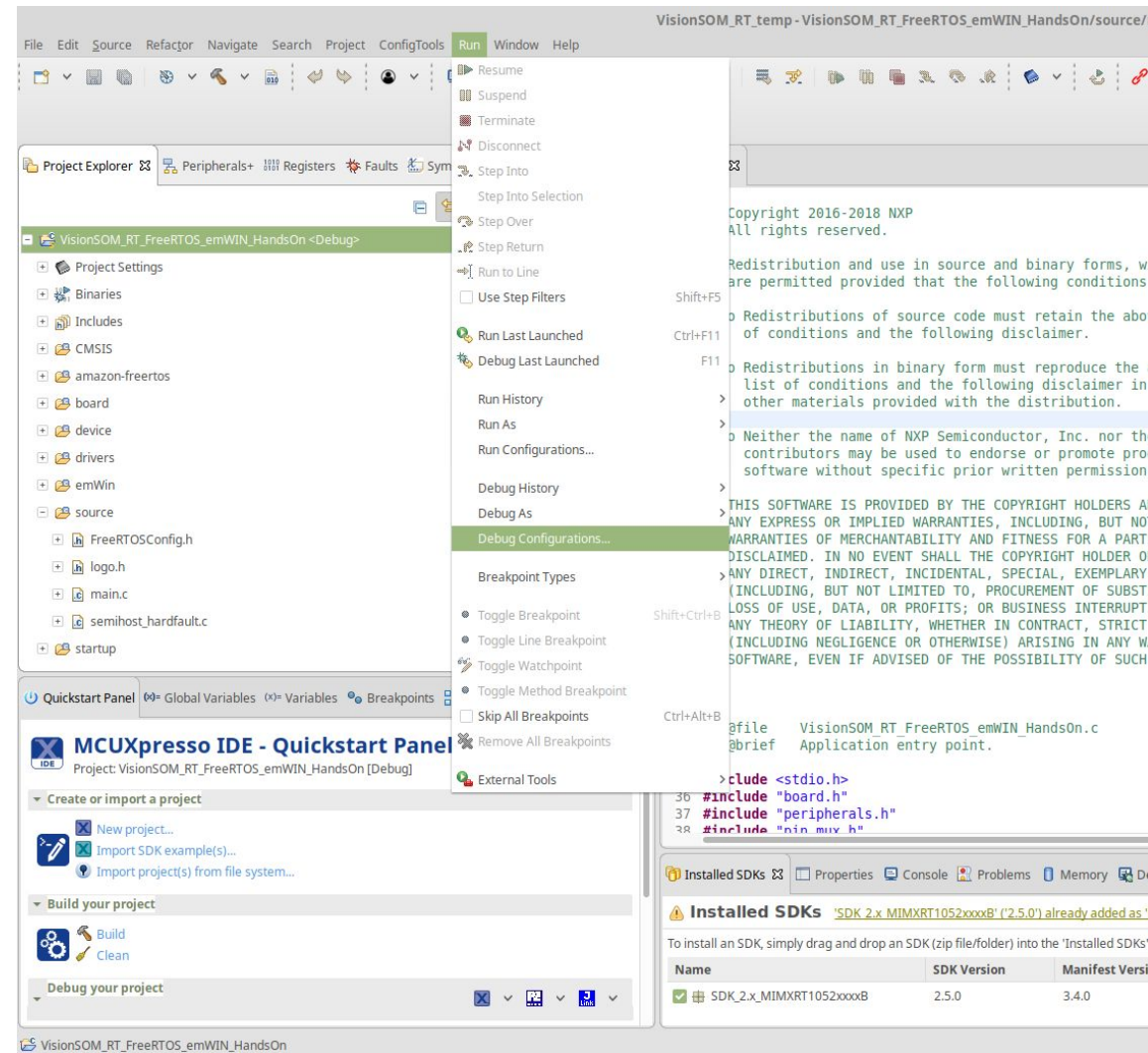
Cancel

OK

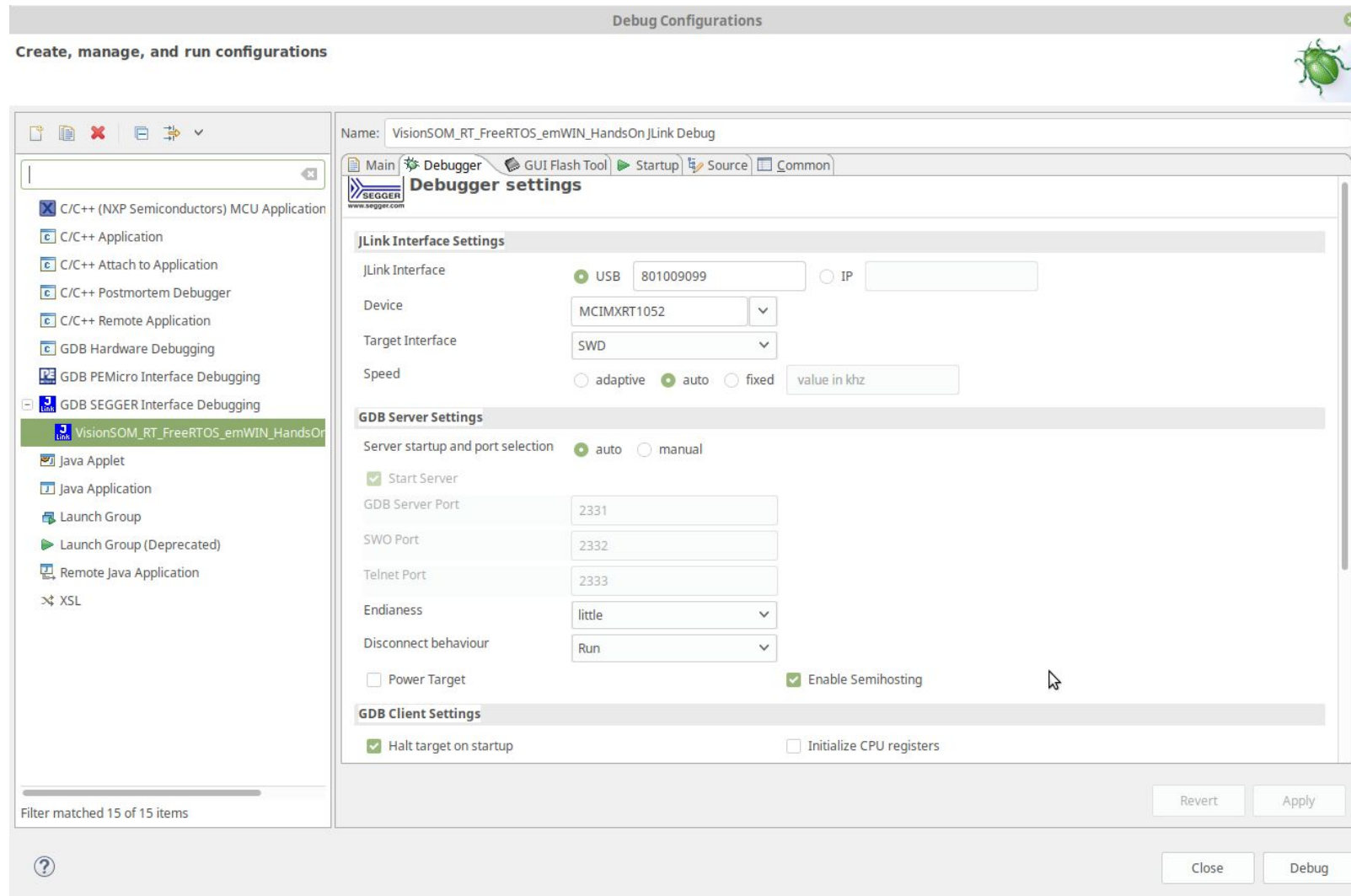
SEGGER J-Link



SEGGER J-Link



SEGGER J-Link



SEGGER J-Link

The screenshot displays the MCUXpresso IDE interface for a FreeRTOS project. The main window shows the source code for `main.c`, which includes a task named `TASK1`. The task is currently running, as indicated by the 'Running' state in the task list. The task list at the bottom shows three tasks: `TASK1` (Running), `IDLE` (Ready), and `Tmr Svc` (Suspended). The task list also shows the task handle, task state, priority, stack usage, event object, and runtime for each task.

TCB#	Task Name	Task Handle	Task State	Priority	Stack Usage	Event Object	Runtime
1	TASK1	0x200002b8	Running	1 (1)	0 B / 436 B		
2	IDLE	0x200004a8	Ready	0 (0)	0 B / 284 B		
3	Tmr Svc	0x200008f8	Suspended	4 (4)	0 B / 604 B	TmrQ (Rx)	

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

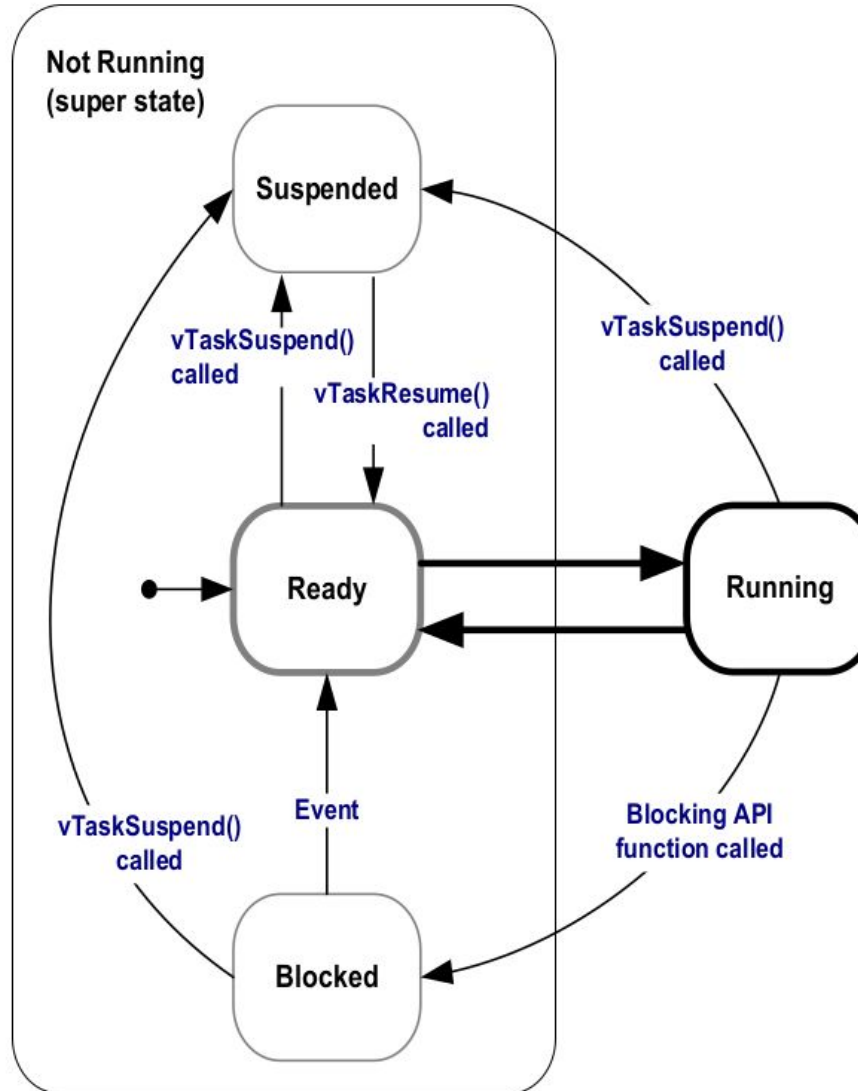
RTOS approach

```
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

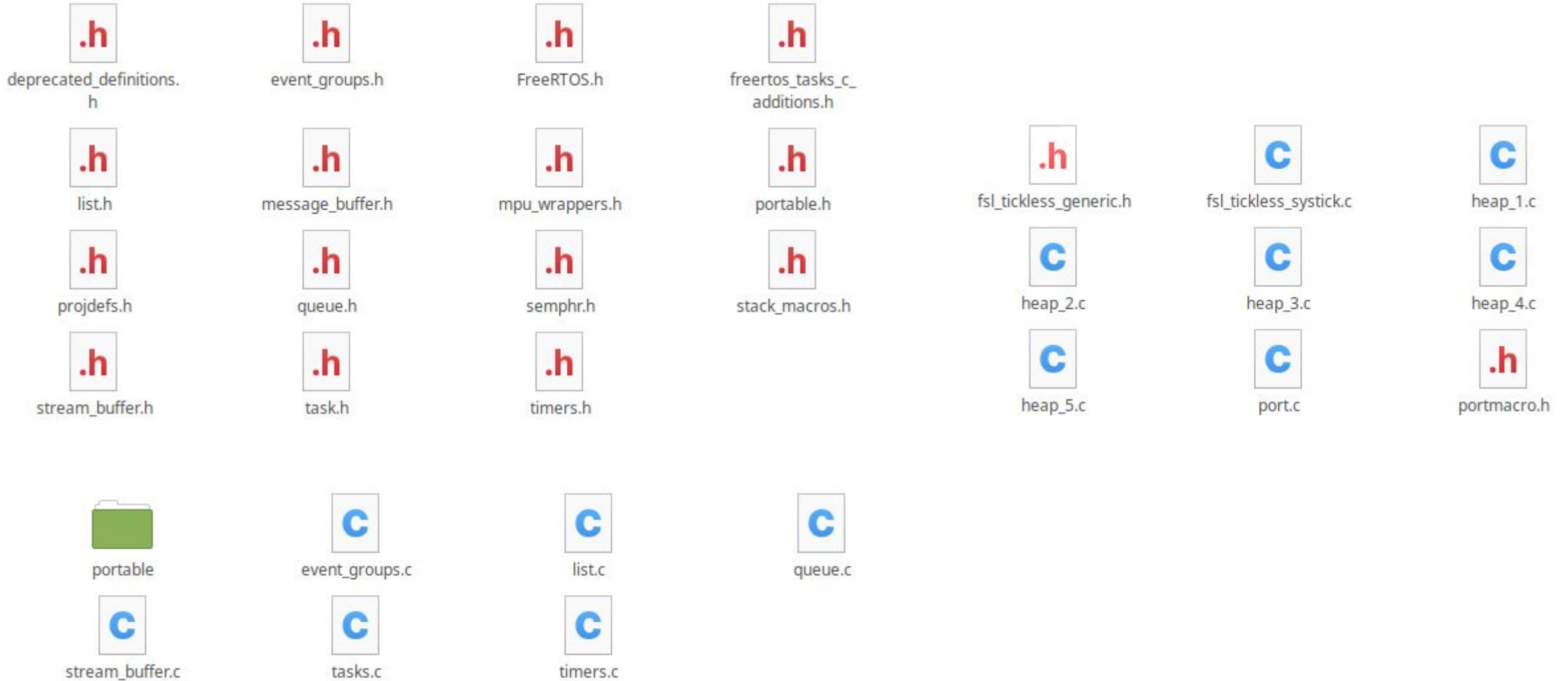
FreeRTOS - Idle task

- 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

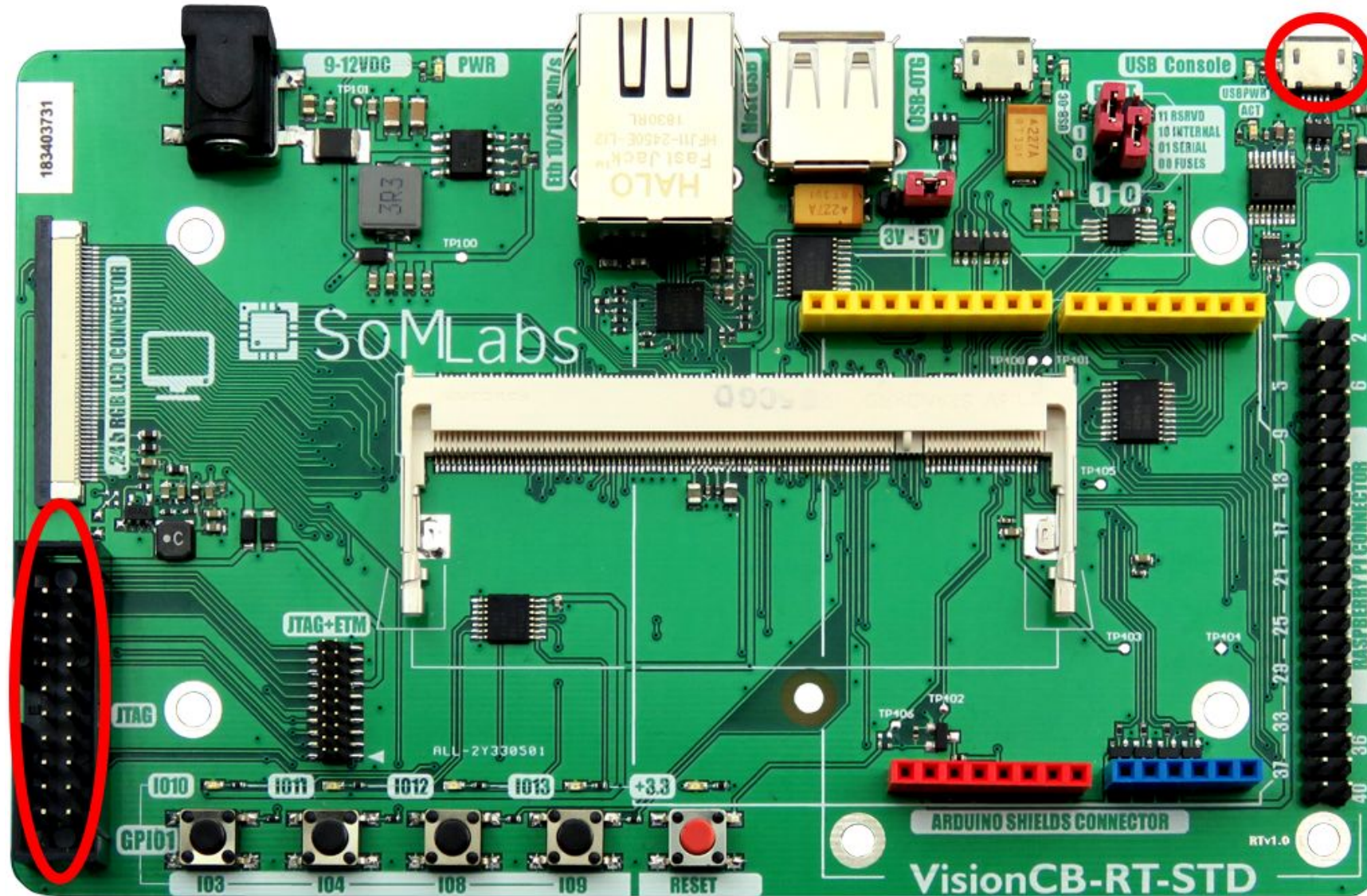
- FreeRTOS ARM Cortex-M3 port for the Keil compiler
- Stack overflow checking turned off
- Trace features turned off
- Run-time stats feature turned off
- Compiler set to optimise for speed
- <https://www.freertos.org/FAQMem.html>

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



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 < 0xFFFFFFFF; 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
```

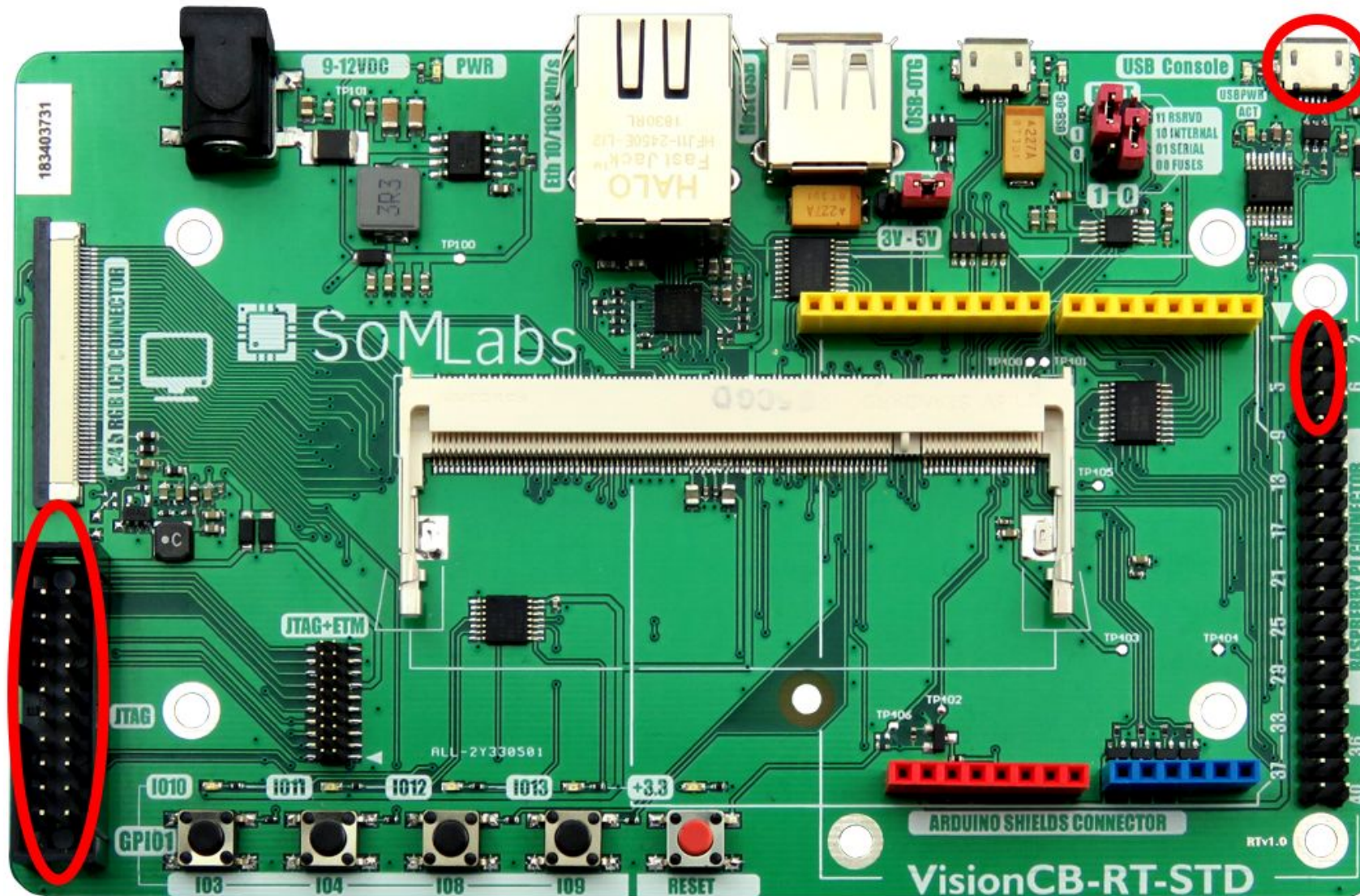
FreeRTOS - queues

- 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

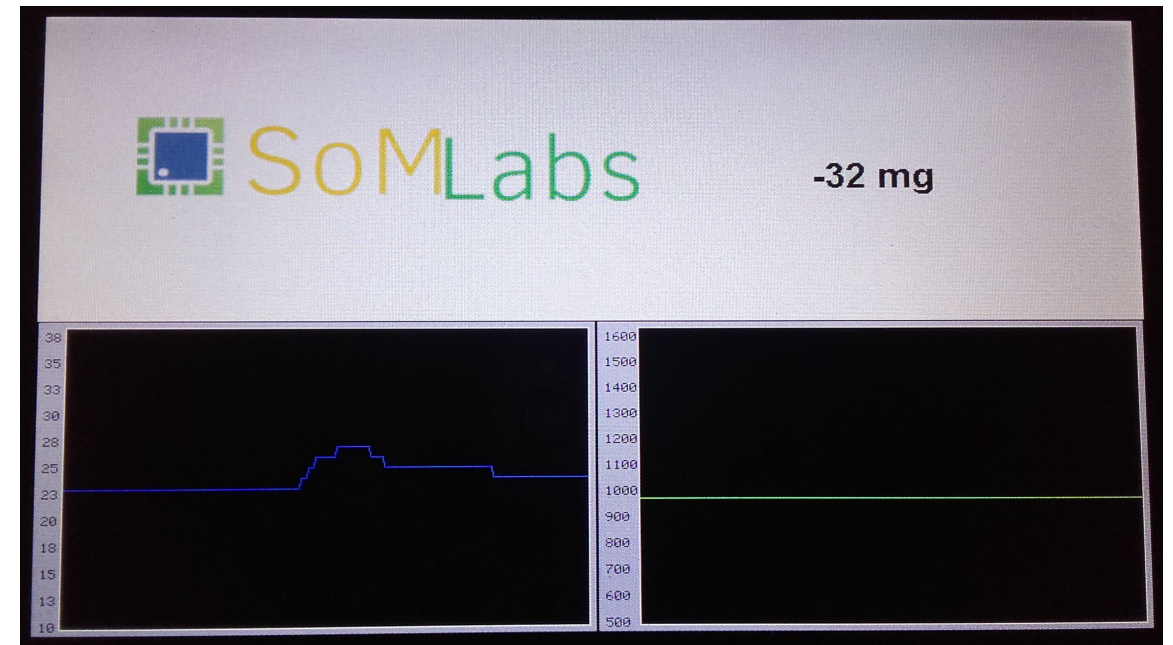
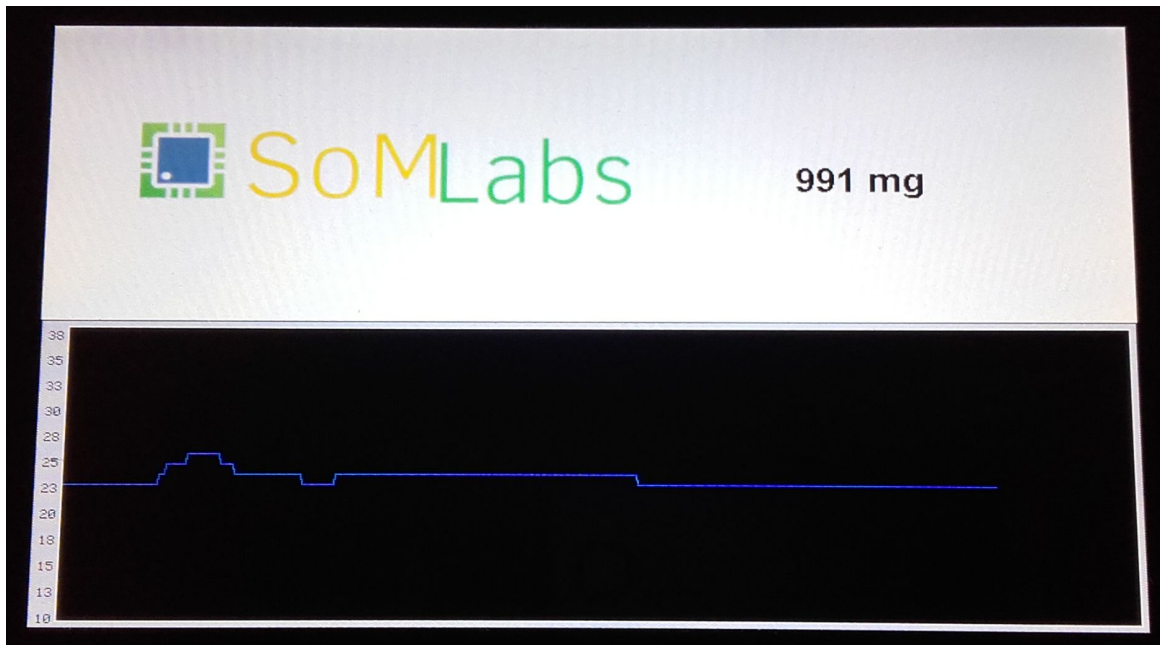
Configuration files:

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

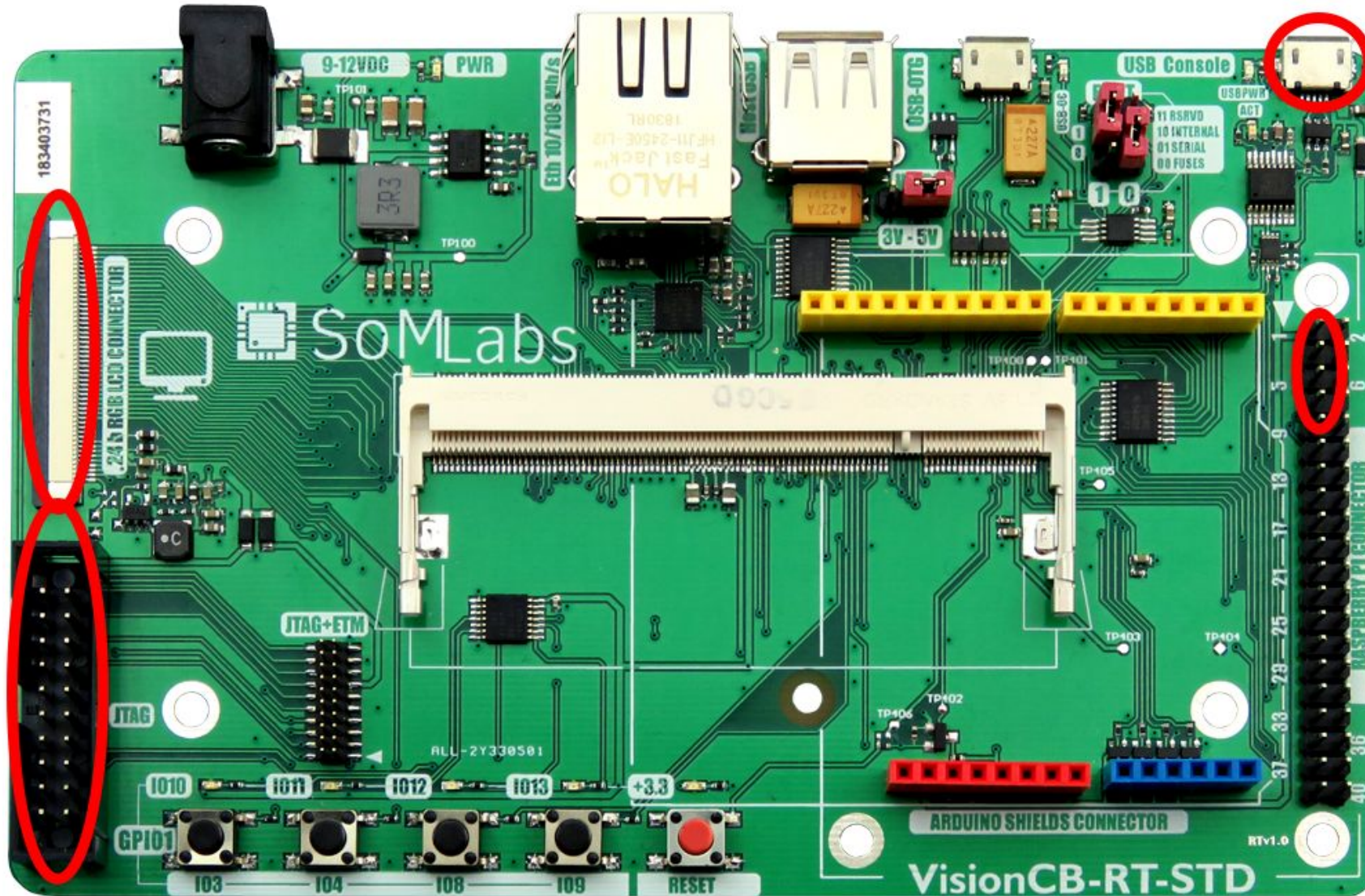
```
#define GUI_MEMORY_ADDR ((uint32_t)s_gui_memory)
#define VRAM_ADDR ((uint32_t)s_vram_buffer)
```

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

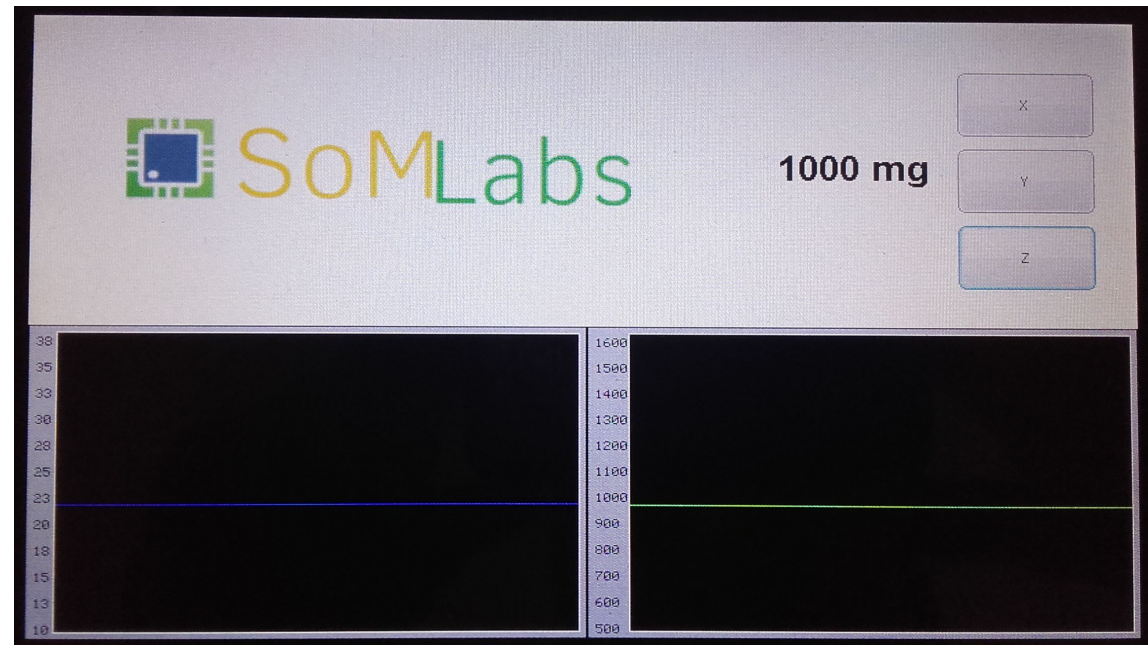
```
GRAPH_Handle GRAPH_CreateEx( int x0, int y0,  
                             int xSize, int ySize,  
                             WM_HWIN hParent,  
                             int WinFlags,  
                             int ExFlags, int Id);
```

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