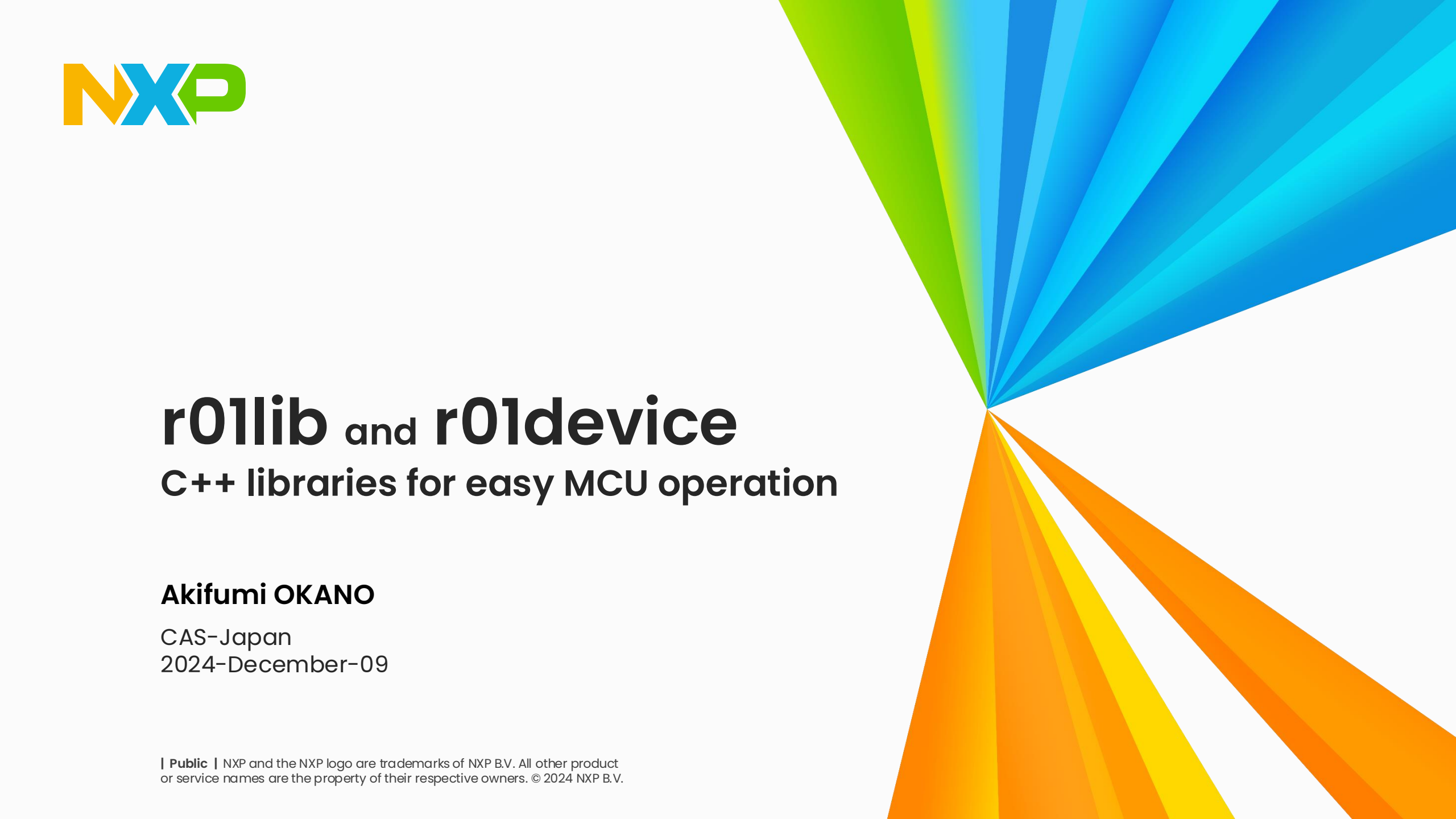


r01lib and r01device

C++ libraries for easy MCU operation

Akifumi OKANO

CAS-Japan
2024-December-09

Executive summary

- **Libraries** are available which enables to write **I3C/I2C/SPI applications easy** on **MCX** mocrocontrollers!

- Same application code works on any supported MCX boards

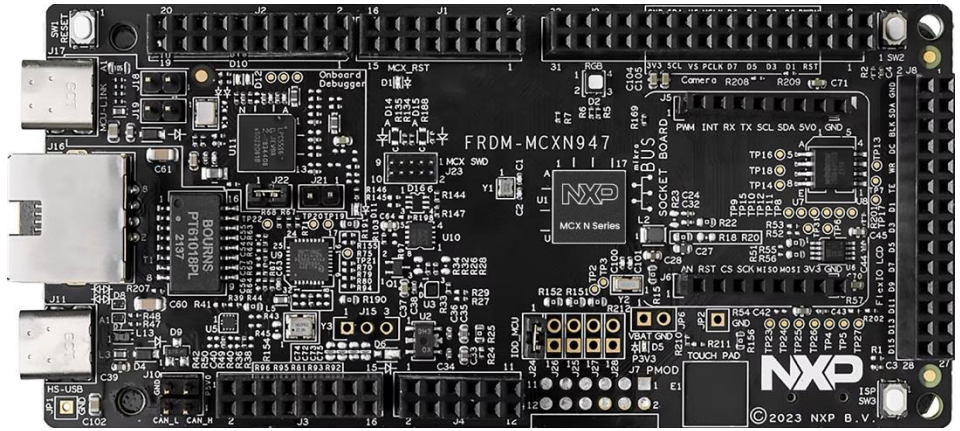
- **‘r01lib’** and **’r01device’** are the libraries provides **simple API** to operate MCU features/peripherals

- The libraries are **suitable** for **writing demo code** and **hands-on** for introducing devices

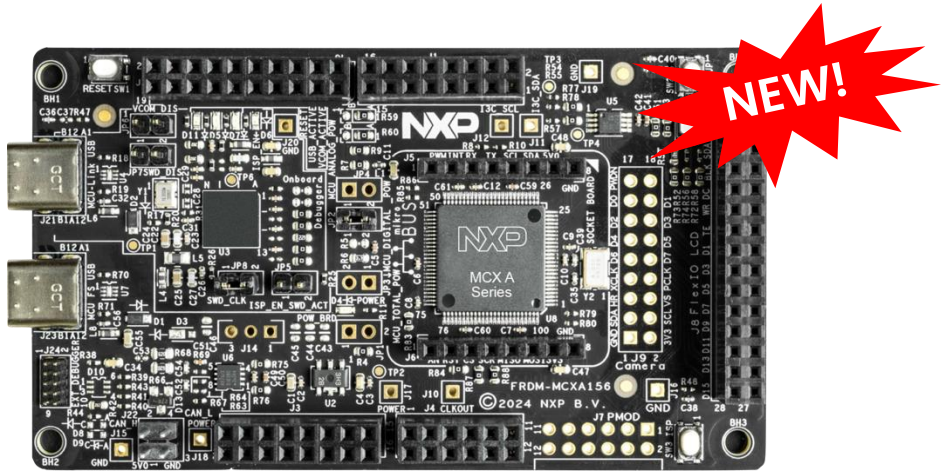
- Any comment, request, bug-report and pull-request are welcome ☺

  → akifumi.okano@nxp.com

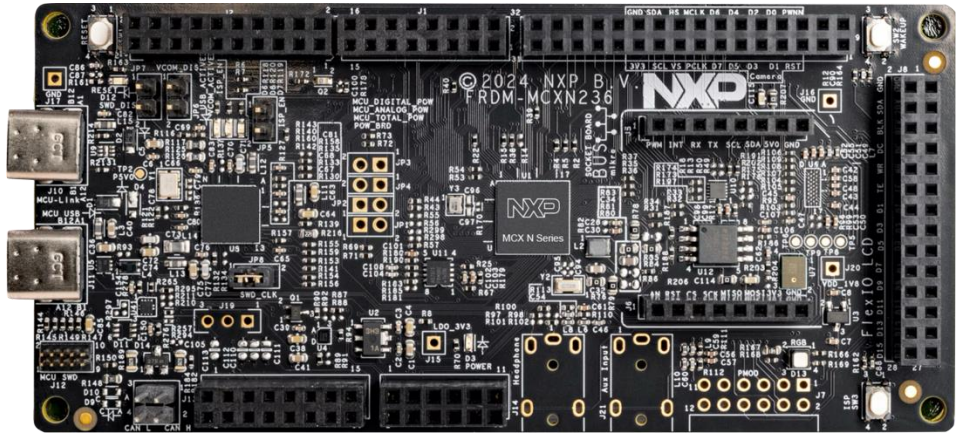
Supported MCU boards



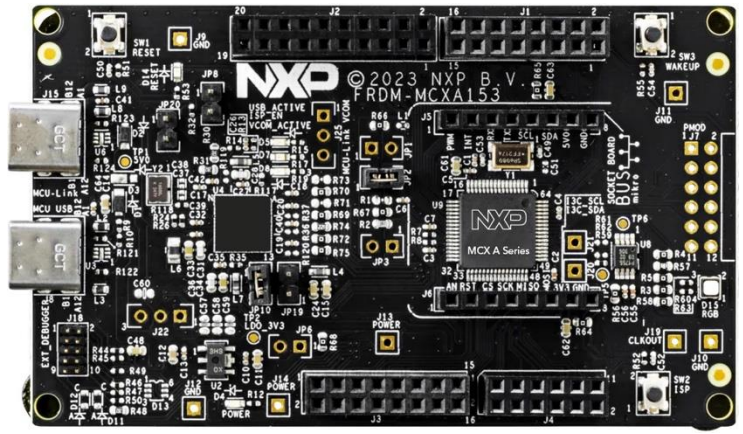
FRDM-MCXN947



FRDM-MCXA156



FRDM-MCXN236



FRDM-MCXA153

For instance.. : Evaluation board sample code can run any MCX with r01lib

RTC (I²C/SPI)

GPIO (I²C/SPI)

LCD (I²C)

Temp sens. (I3C)

LED (I²C/SPI)

R01 Arduino shield (ARD) boards

App | App | App | App | App

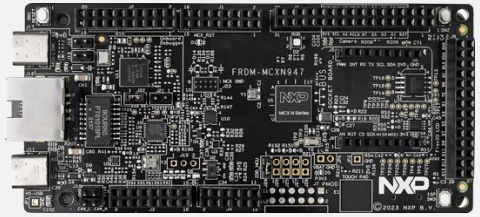
Application for each ARD boards
Written on "r01lib". **Same code runs on any MCX boards**

r01lib (+ r01device)

"r01lib" is an abstraction layer for MCXs
Abstracting I3C, I²C, SPI, GPIO, Timer, Interrupt with simple C++ API

SDK | SDK | SDK | SDK

SDK for each MCX boards
Drivers for each MCU and peripherals



FRDM-MCXN947

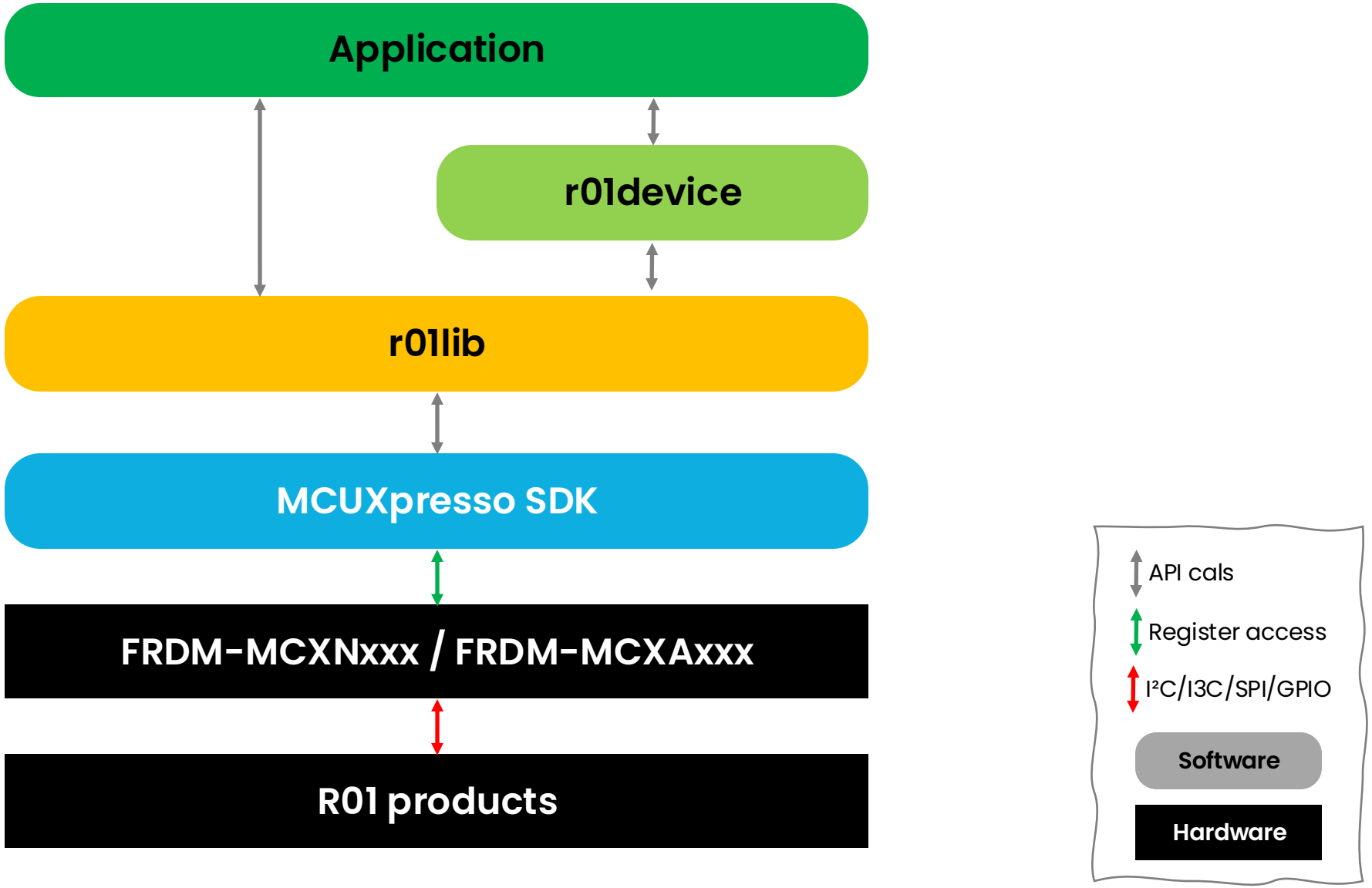
FRDM-MCXN236

FRDM-MCXA156

FRDM-MCXA153

FRDM-MCX boards

Hardware/Software structure

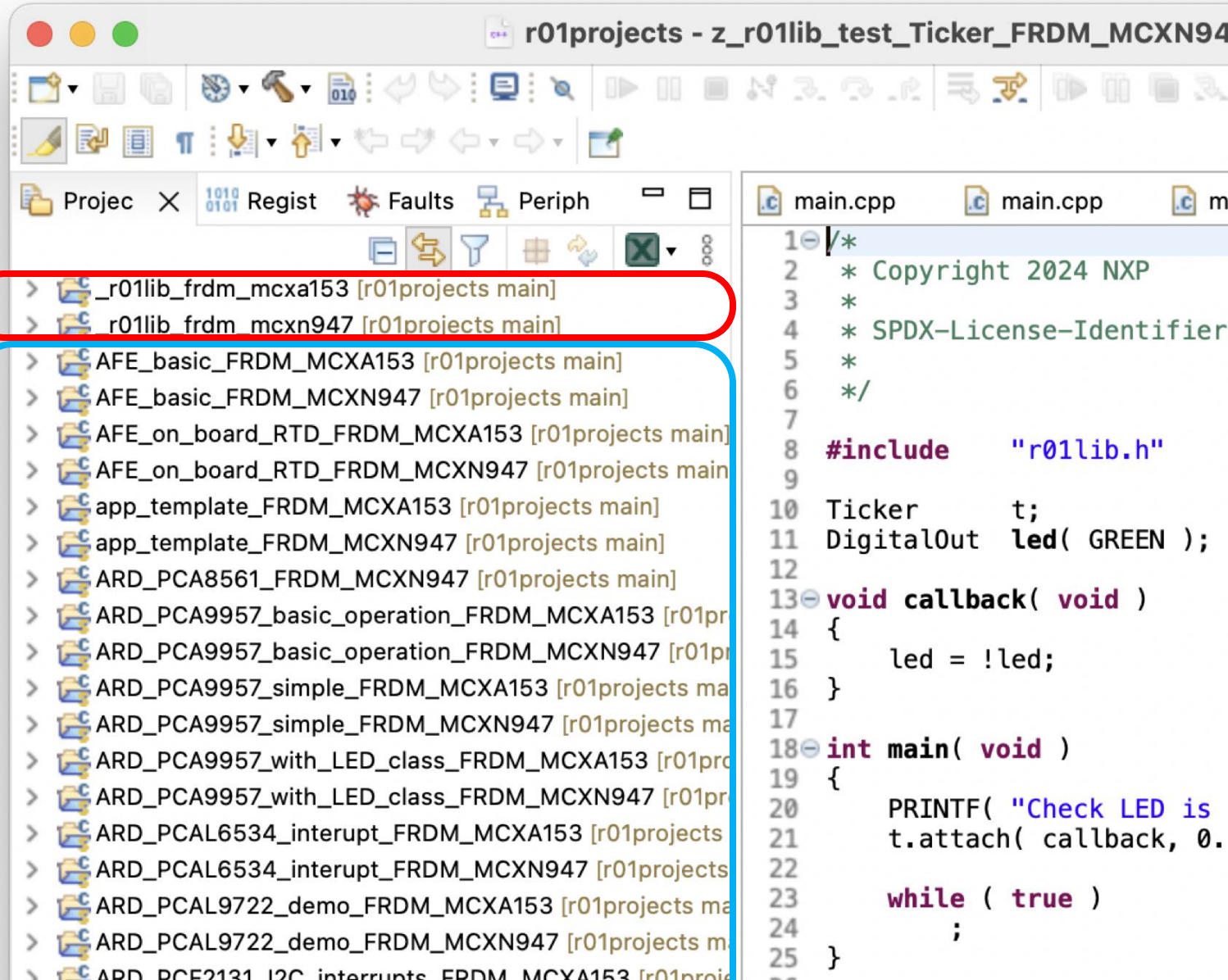


How will it be looked like?

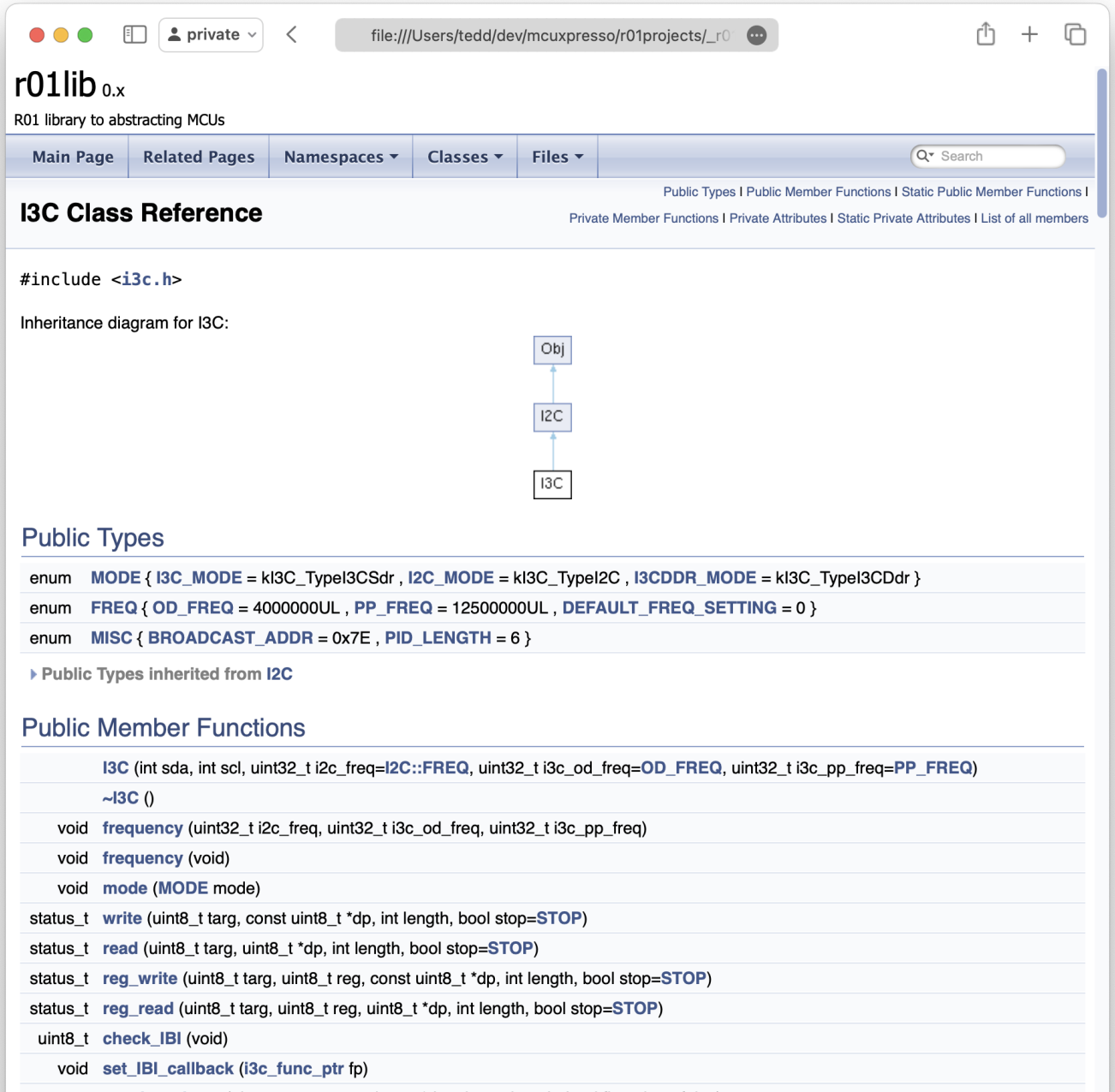
- The **r01lib** is delivered as a part of sample code

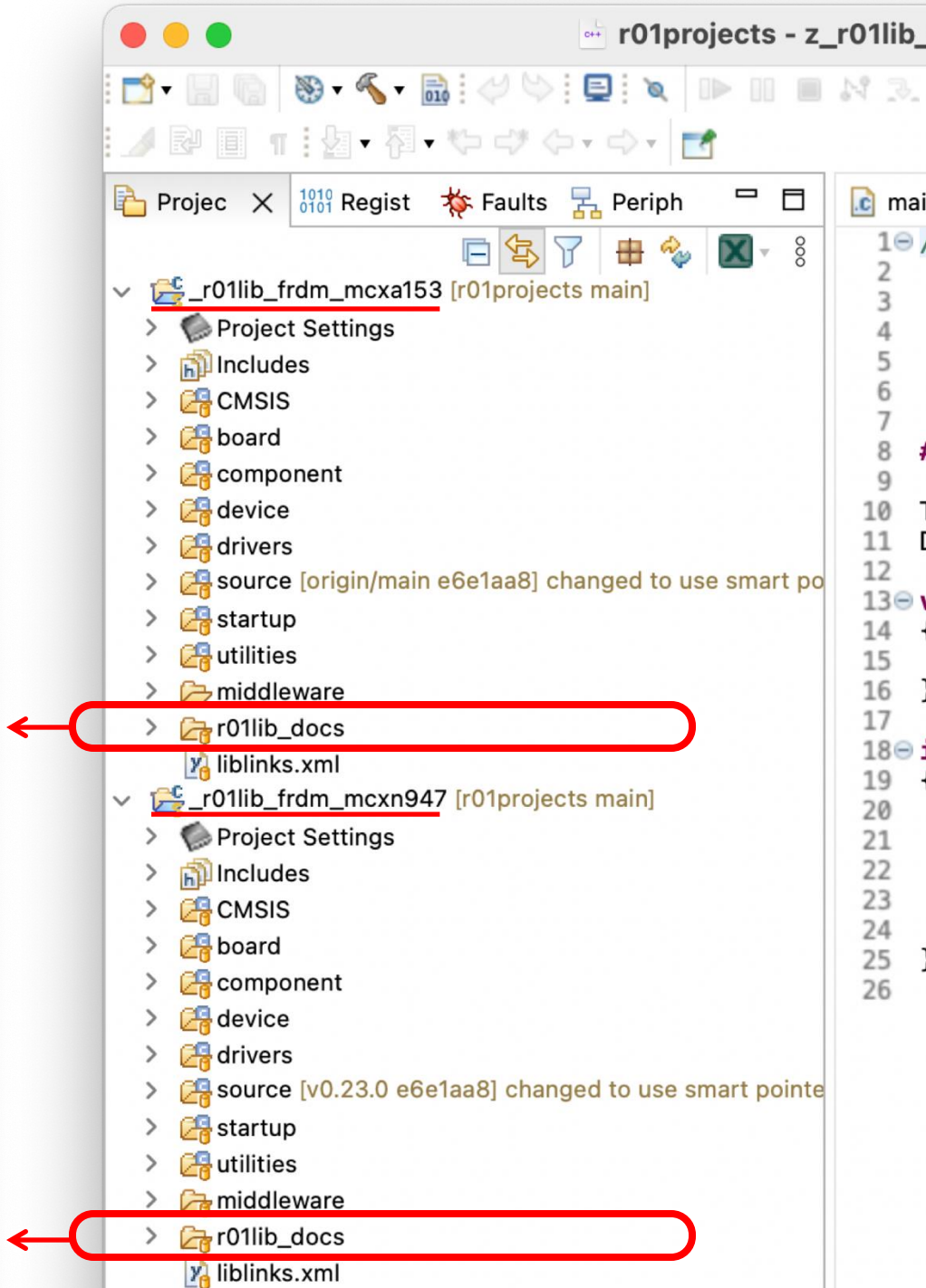
The **r01lib** is made as library project for each MCU boards

Various sample code projects those will be linked with **r01lib** library



API documents are available





00

r01lib

MCU abstraction layer

Library for abstracting MCU on MCUXpresso
Simple interface to write sample code

- **r01lib** = A driver collection for MCU's bus/pin operations

- Problem so far
  - The **MCUXpresso SDK drivers** are available in **low level**. It is good to software engineer who needs to derive maximum MCU performance but not easy to use for everyone
  - Assigning MCU pin requires **low level port and clock settings** or **use of "Config Tool"**
  - MCX desn't have user friendly development environment like other MCUs yet. For instance, "Mbed" and "Arduino" are not adopted for MCX those are popular SDKs providing simple API with abstracting MCUs.

- What has been done
  - Made a small set of library which controls **I3C, I2C, SPI, GPIO, pin-interrupt** and **timer**
  - The user-interface is made like Mbed SDK: Features and pins can be operated with high level simple interface
  - **wait()** function to control timing
  - The library can be bundled with demo code as a MCUXpresso project
  - The library is ported to FRDM-MCXN947, FRDM-MCXN236, FRDM-MCXA156 and FRDM-MCXA153. The library make possible to run same code on both evaluation boards

Available APIs

- DigitalInOut / DigitalIn / DigitalOut class
- BusInOut / BusIn / BusOut classes
- InterruptIn class
- Ticker class
- I2C class
- I3C class
- SPI class
- Wait function

01

r01lib operation sample

Simple API

Sample 0 : DigitalOut
LED blink

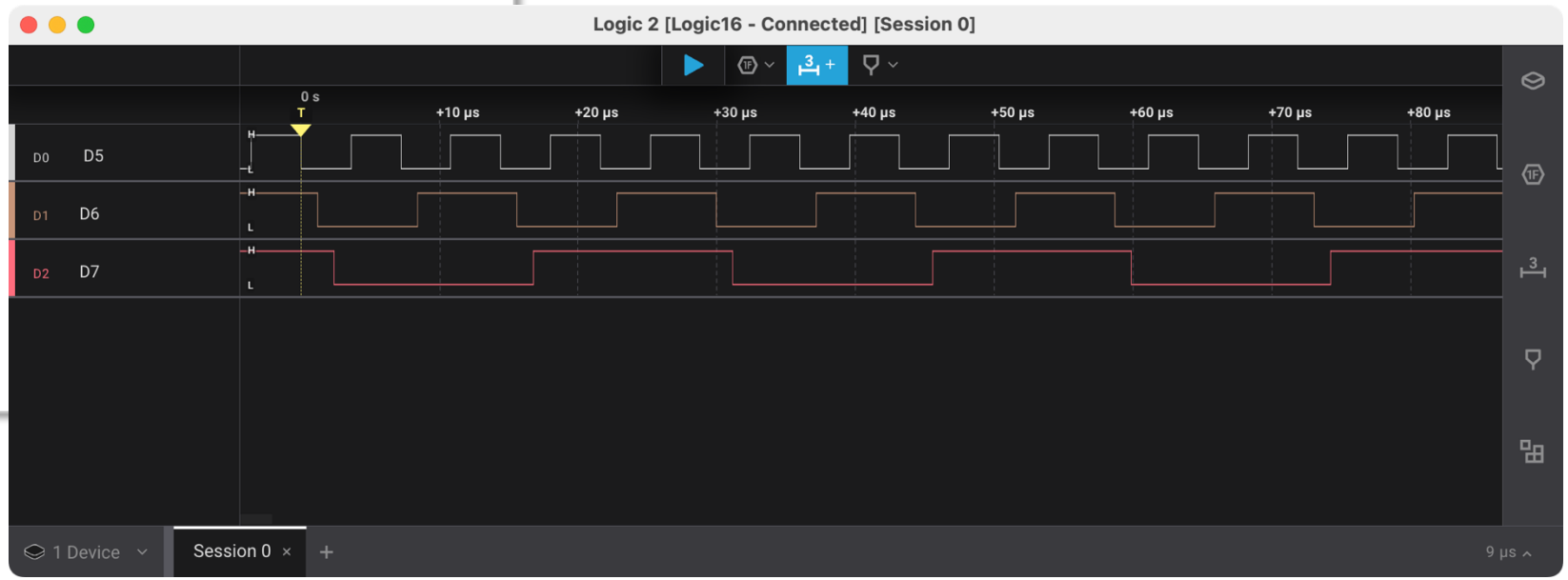
```
#include "r01lib.h"

DigitalOut led( GREEN );

int main(void)
{
    while ( 1 )
    {
        led = !led;
        wait( 0.5 );
    }
}
```

- GPIO can be controlled by simple interface
- No need to care about GPIO registers
- Pin names are defined in “component/r01lib/io.h”
- **Wait()** takes float number (seconds). Down to microsecond resolution

Sample 2 : DigitalOut
Flips GPIO pins

```
#include "r01lib.h"

DigitalOut d5( D5 );
DigitalOut d6( D6 );
DigitalOut d7( D7 );

int main(void)
{
    int count = 0;
    while ( 1 )
    {
        d5 = (count >> 0) & 0x01;
        d6 = (count >> 1) & 0x01;
        d7 = (count >> 2) & 0x01;
        count++;
    }
}
```

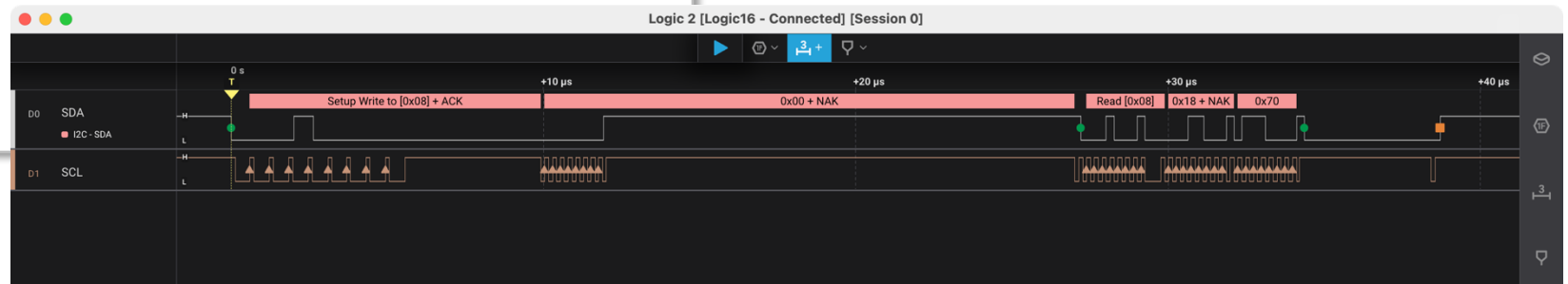
- It can handle multiple GPIOs
- Input can be done by **DigitalIn** and **DigitalInOut** classes



Sample 3 : I2C
Low level P3T1755 access

```
#include "r01lib.h"

I2C i2c( I2C_SDA, I2C_SCL );

int main( void )
{
    constexpr uint8_t address = 0x4C;
    uint8_t w_data[] = { 0 };
    uint8_t r_data[ 2 ];

    while ( true )
    {
        i2c.write( address, w_data, sizeof( w_data ), I2C::NO_STOP );
        i2c.read(  address, r_data, sizeof( r_data ) );

        printf( "0x %02X %02X\r\n", r_data[ 0 ], r_data[ 1 ] );
        wait( 1 );
    }
}
```

- I2C class provides a simple interface to perform transfer



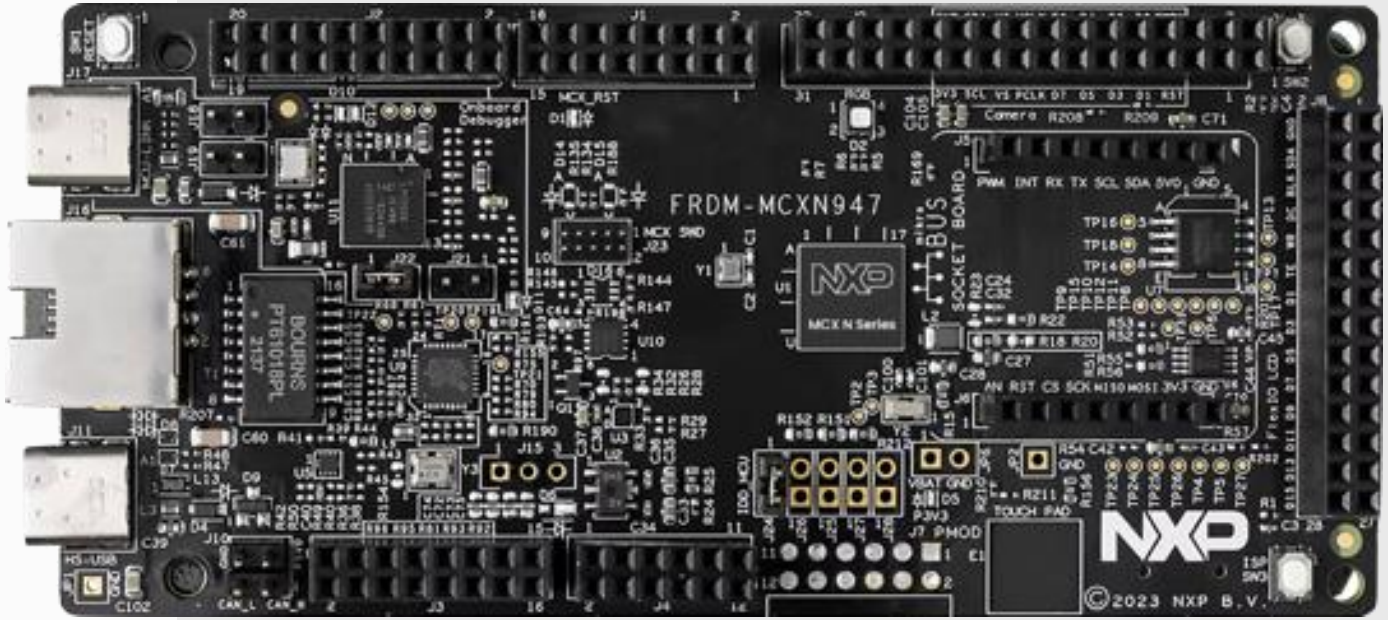
02

Available pins

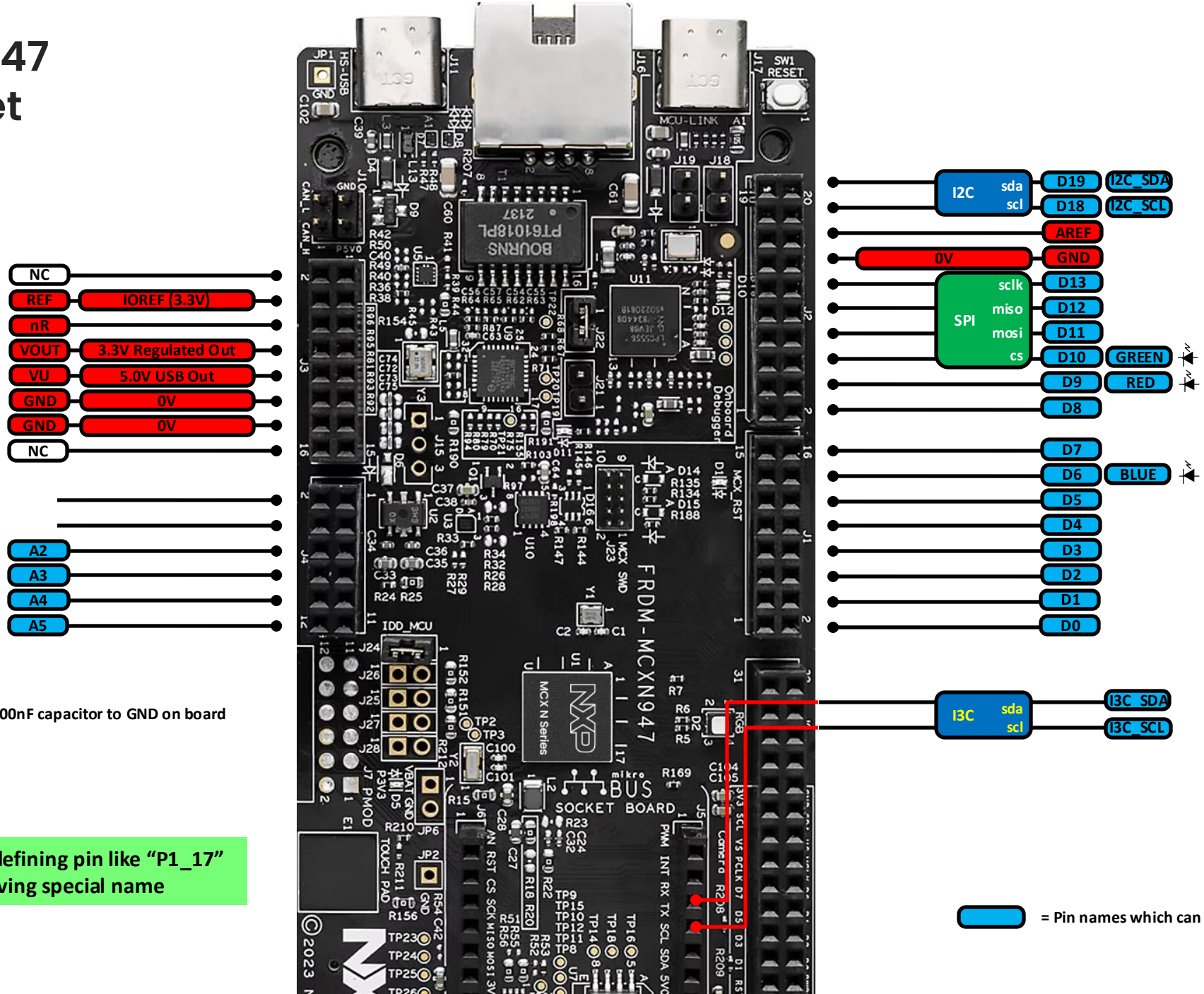
FRDM-MCXN947
FRDM-MCXN236
FRDM-MCXA156
FRDM-MCXA153

Pins

FRDM-MCXN947



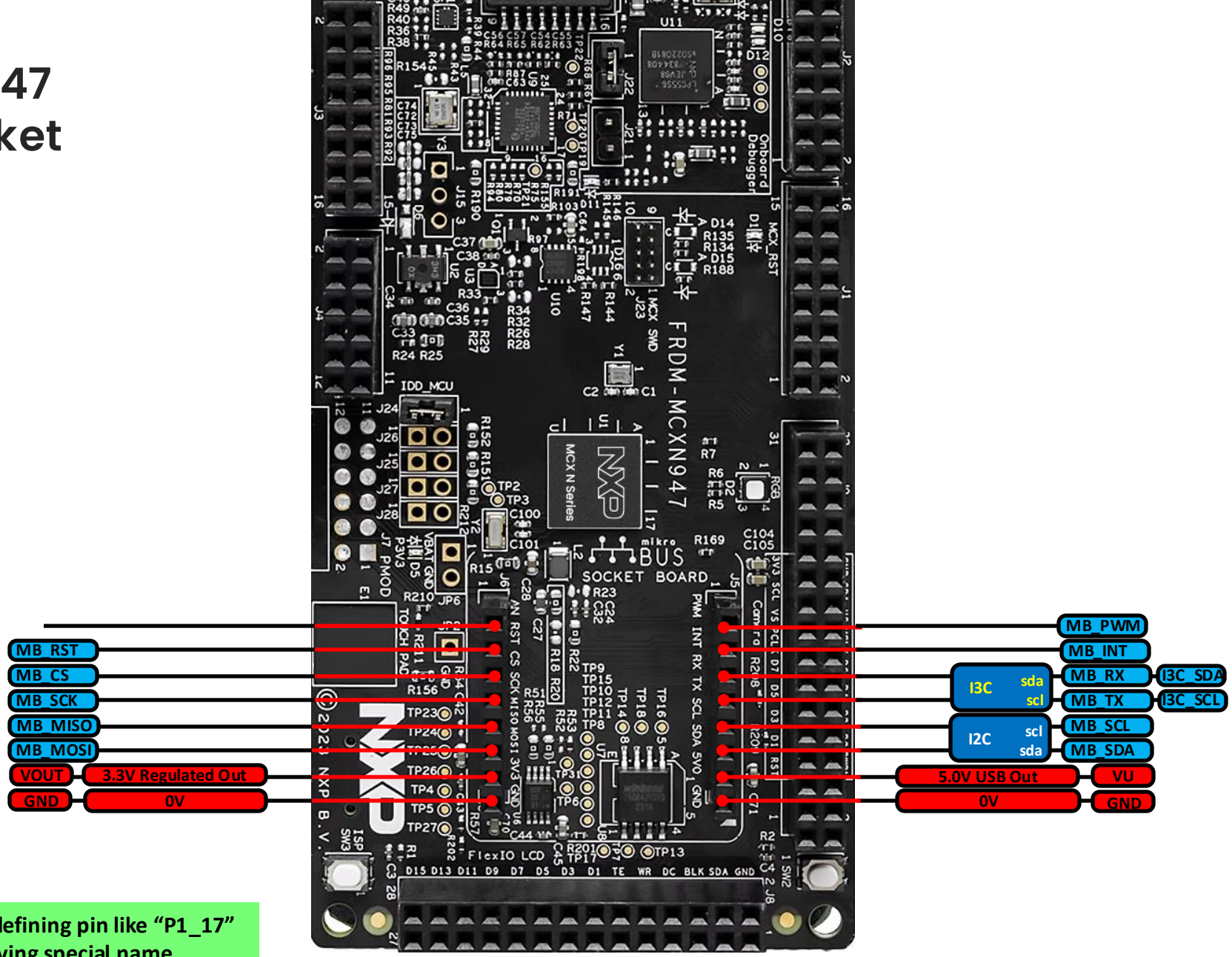
FRDM-MCXN947
Arduino socket



* Note: A5 pin is tied to 10kΩ pull-up and 100nF capacitor to GND on board

All pin sockets can be used by defining pin like “P1_17”
This page shows pin sockets having special name

FRDM-MCXN947
mikroBUS socket



All pin sockets can be used by defining pin like “P1_17”
This page shows pin sockets having special name

= Pin names which can be used in the code

Available pins
N947

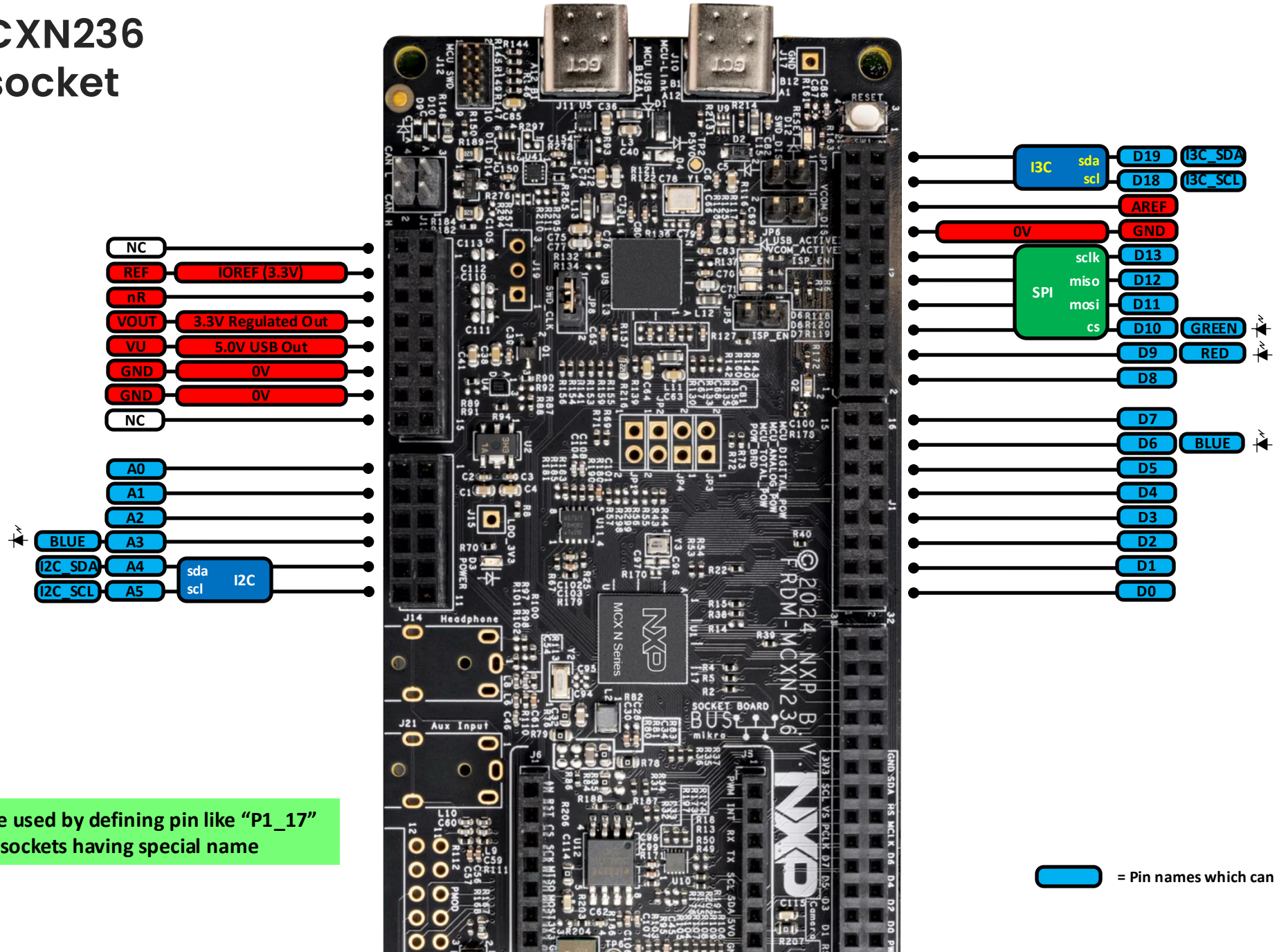
r01lib pin names

pin definition				class				
pin name	other pin name (1)	other pin name (2)	other pin name (3)	Digital&Bus InOut	InterruptIn	I2C	I3C	SPI
D0			P4_3	✅	✅			
D1			P4_2	✅	✅			
D2			P0_29	✅	✅			
D3			P1_23	✅	✅			
D4			P0_30	✅	✅			
D5			P1_21	✅	✅			
D6	BLUE		P1_2	✅	✅			
D7			P0_31	✅	✅			
D8			P0_28	✅	✅			
D9	RED		P0_10	✅	✅			
D10	GREEN	SPI_CS	P0_27	✅	✅			
D11		SPI_MOSI	P0_24	✅	✅			✅
D12		SPI_MISO	P0_26	✅	✅			
D13		SPI_SCLK	P0_25	✅	✅			
D18		I2C_SDA	P4_0	✅	✅	✅		
D19		I2C_SCL	P4_1	✅	✅			
A0								
A1								
A2			P0_14	✅	✅			
A3			P0_22	✅	✅			
A4			P0_15	✅	✅			
A5	SW2		P0_23	✅	✅			
MB_AN								
MB_RST			P1_3	✅	✅			
MB_CS			P3_23	✅	✅			
MB_SCK			P3_21	✅	✅			
MB_MISO			P3_22	✅	✅			
MB_MOSI			P3_20	✅	✅			
MB_PWM			P3_19	✅	✅			
MB_INT			P5_7	✅	✅			
MB_RX		I3C_SDA	P1_16	✅	✅		✅	
MB_TX		I3C_SCL	P1_17	✅	✅			
MB_SCL			P1_1	✅	✅			
MB_SDA			P1_0	✅	✅			
	SW3		P0_6	✅	✅			

Pins

FRDM-MCXN236

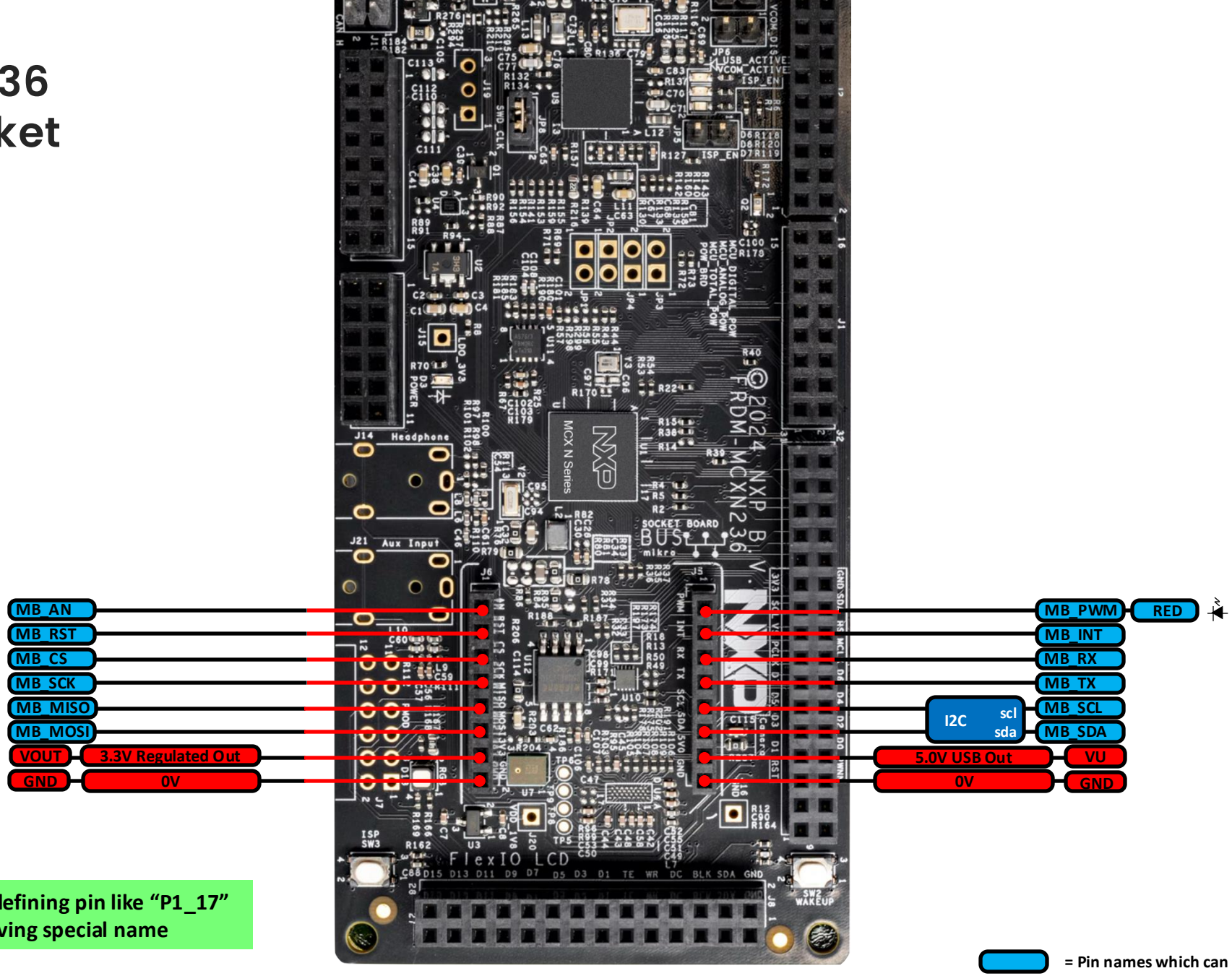
FRDM-MCXN236
Arduino socket



All pin sockets can be used by defining pin like “P1_17”
This page shows pin sockets having special name

= Pin names which can be used in the code

FRDM-MCXN236 mikroBUS socket



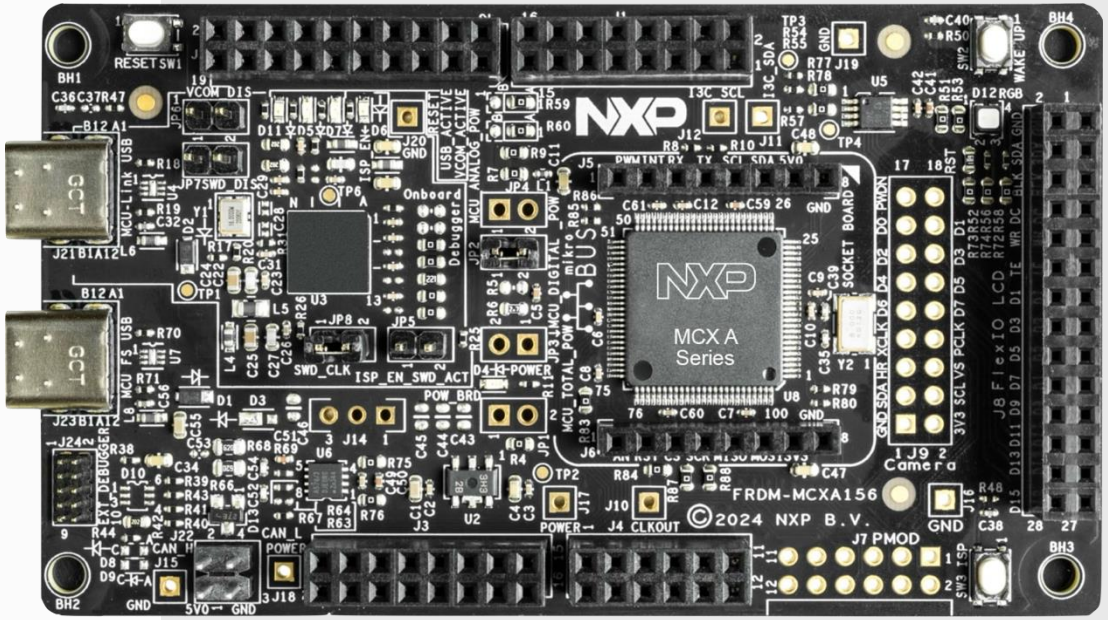
All pin sockets can be used by defining pin like “P1_17”
This page shows pin sockets having special name

Available pins
N236

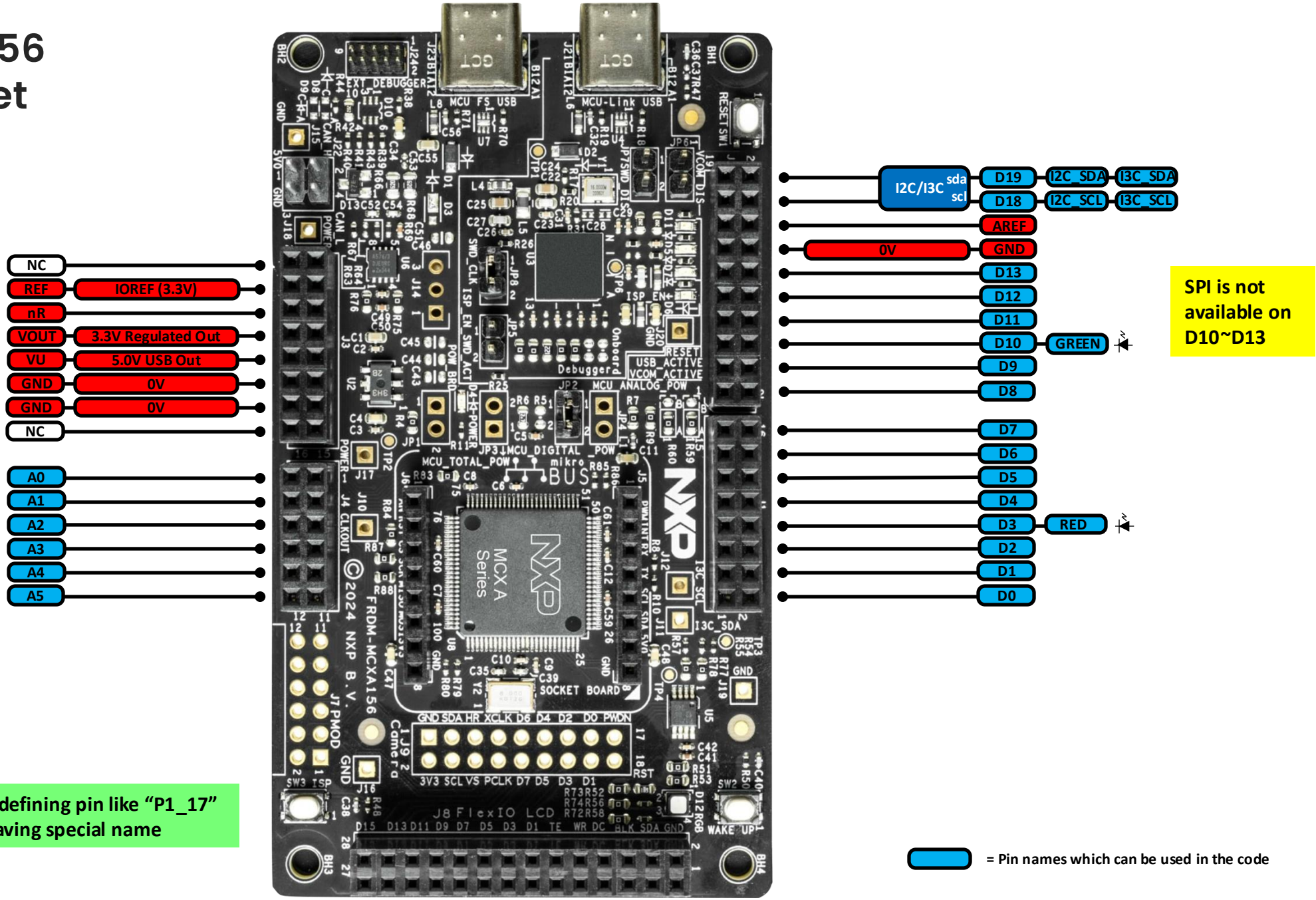
r01lib pin names								
pin definition				class				
pin name	other pin name (1)	other pin name (2)	other pin name (3)	Digital&Bus InOut	InterruptIn	I2C	I3C	SPI
D0			P4_3	✅	✅			
D1			P4_2	✅	✅			
D2			P2_0	✅	✅			
D3			P3_12	✅	✅			
D4			P0_21	✅	✅			
D5			P2_7	✅	✅			
D6			P3_17	✅	✅			
D7			P0_22	✅	✅			
D8			P0_23	✅	✅			
D9			P3_14	✅	✅			
D10	SPI_CS		P1_3	✅	✅			
D11	SPI_MOSI		P1_0	✅	✅			✅
D12	SPI_MISO		P1_2	✅	✅			
D13	SPI_SCLK		P1_1	✅	✅			
D18	MB_CS	I3C_SDA	P1_16	✅	✅		✅	
D19		I3C_SCL	P1_17	✅	✅			
A0			P4_6	✅	✅			
A1			P4_15	✅	✅			
A2			P4_16	✅	✅			
A3	BLUE		P4_17	✅	✅			
A4		I2C_SDA	P4_12	✅	✅	✅		
A5		I2C_SCL	P4_13	✅	✅			
MB_AN			P5_3	✅	✅			
MB_RST			P5_2	✅	✅			
MB_CS	D18		P1_16	✅	✅			
MB_SCK			P1_1	✅	✅			
MB_MISO			P1_2	✅	✅			
MB_MOSI			P1_0	✅	✅			
MB_PWM	RED		P4_18	✅	✅			
MB_INT			P5_6	✅	✅			
MB_RX			P4_3	✅	✅			
MB_TX			P4_2	✅	✅			
MB_SCL			P4_1	✅	✅	✅		
MB_SDA			P4_0	✅	✅			
	GREEN		P4_19	✅	✅			
	SW2		P0_20	✅	✅			
	SW3		P0_6	✅	✅			

Pins

FRDM-MCXA156



FRDM-MCXA156
Arduino socket



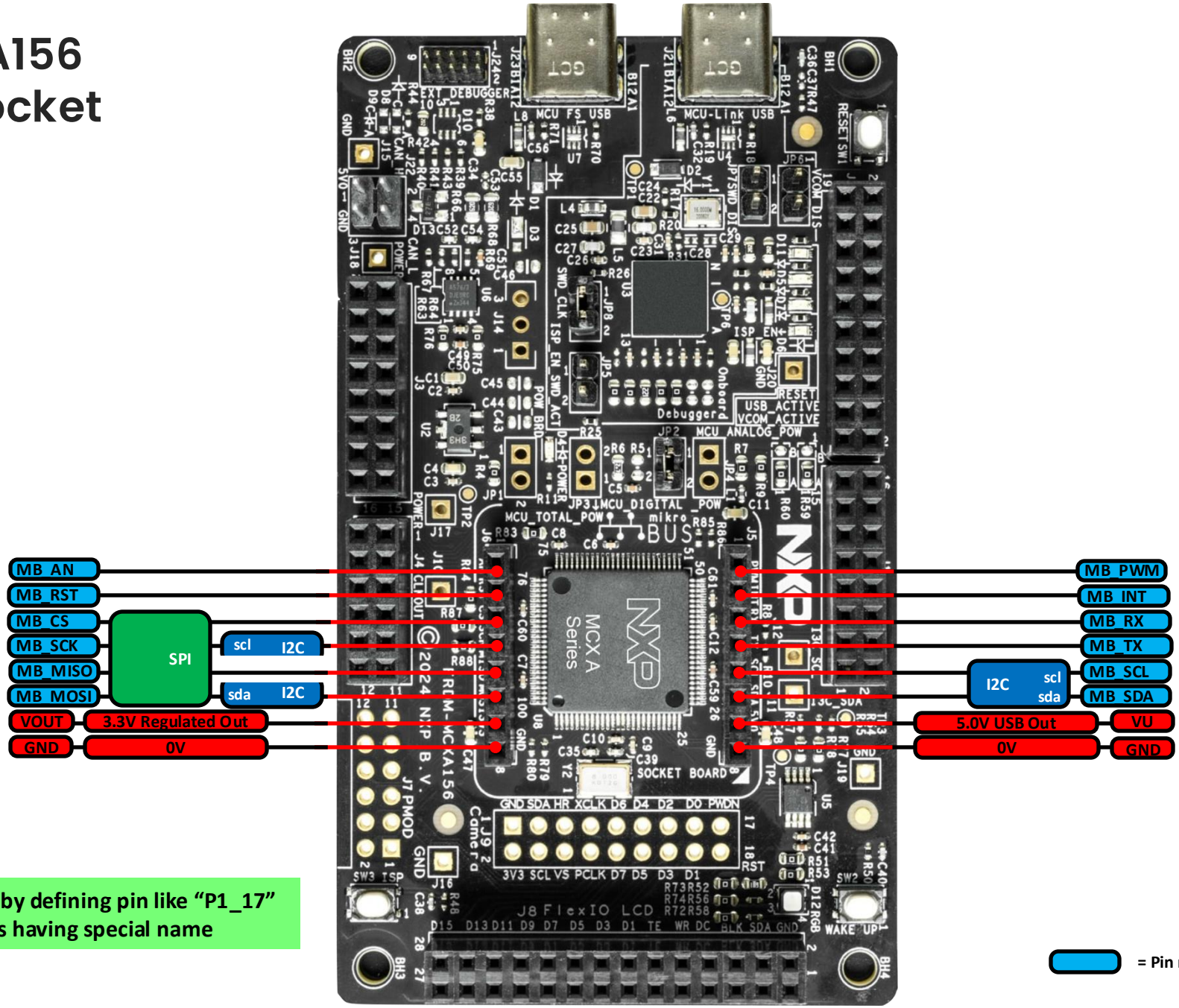
To use I²C on A4/A5, remove R75 and R76

SPI is not available on D10~D13

All pin sockets can be used by defining pin like “P1_17”
This page shows pin sockets having special name

= Pin names which can be used in the code

FRDM-MCXA156 mikroBUS socket



All pin sockets can be used by defining pin like “P1_17”
This page shows pin sockets having special name

= Pin names which can be used in the code

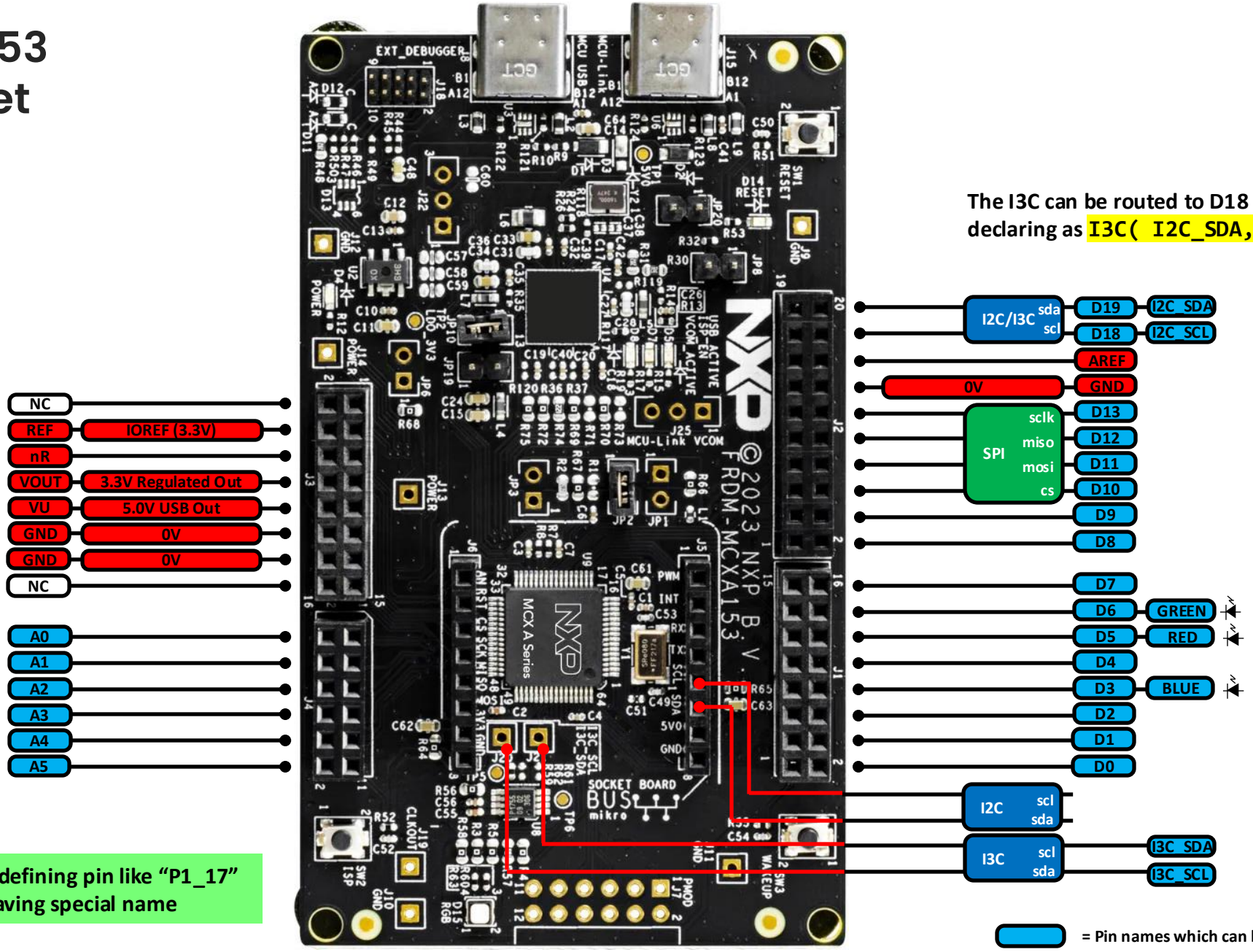
Available pins
A156

r01lib pin names								
pin definition				class				
pin name	other pin name (1)	other pin name (2)	other pin name (3)	Digital&Bus InOut	InterruptIn	I2C	I3C	SPI
D0			P2_11	✅	✅			
D1			P2_10	✅	✅			
D2			P3_1	✅	✅			
D3	RED		P3_12	✅	✅			
D4			P3_31	✅	✅			
D5			P3_14	✅	✅			
D6			P3_16	✅	✅			
D7			P1_14	✅	✅			
D8			P1_15	✅	✅			
D9			P3_17	✅	✅			
D10	GREEN		P3_13	✅	✅			
D11			P3_15	✅	✅			
D12			P2_16	✅	✅			
D13			P2_12	✅	✅			
D18	I3C_SDA	I2C_SDA	P0_16	✅	✅	✅	✅	
D19	I3C_SCL	I2C_SCL	P0_17	✅	✅			
A0			P1_10	✅	✅			
A1			P2_5	✅	✅			
A2			P2_3	✅	✅			
A3			P2_4	✅	✅			
A4			P1_12	✅	✅			
A5			P1_13	✅	✅			
MB_AN			P3_30	✅	✅			
MB_RST			P3_29	✅	✅			
MB_CS	SPI_CS		P1_3	✅	✅			
MB_SCK	SPI_SCLK		P1_1	✅	✅	✅ with MK_MOSI		✅
MB_MISO	SPI_MISO		P1_2	✅	✅			
MB_MOSI	SPI_MOSI		P1_0	✅	✅	✅ with MK_SCK		
MB_PWM			P3_18	✅	✅			
MB_INT			P3_19	✅	✅			
MB_RX			P3_20	✅	✅			
MB_TX			P3_21	✅	✅			
MB_SCL			P3_27	✅	✅	✅		
MB_SDA			P3_28	✅	✅			
	SW2		P1_7	✅	✅			
	SW3		P0_6	✅	✅			
	BLUE		P3_29	✅	✅			

Pins

FRDM-MCXA153

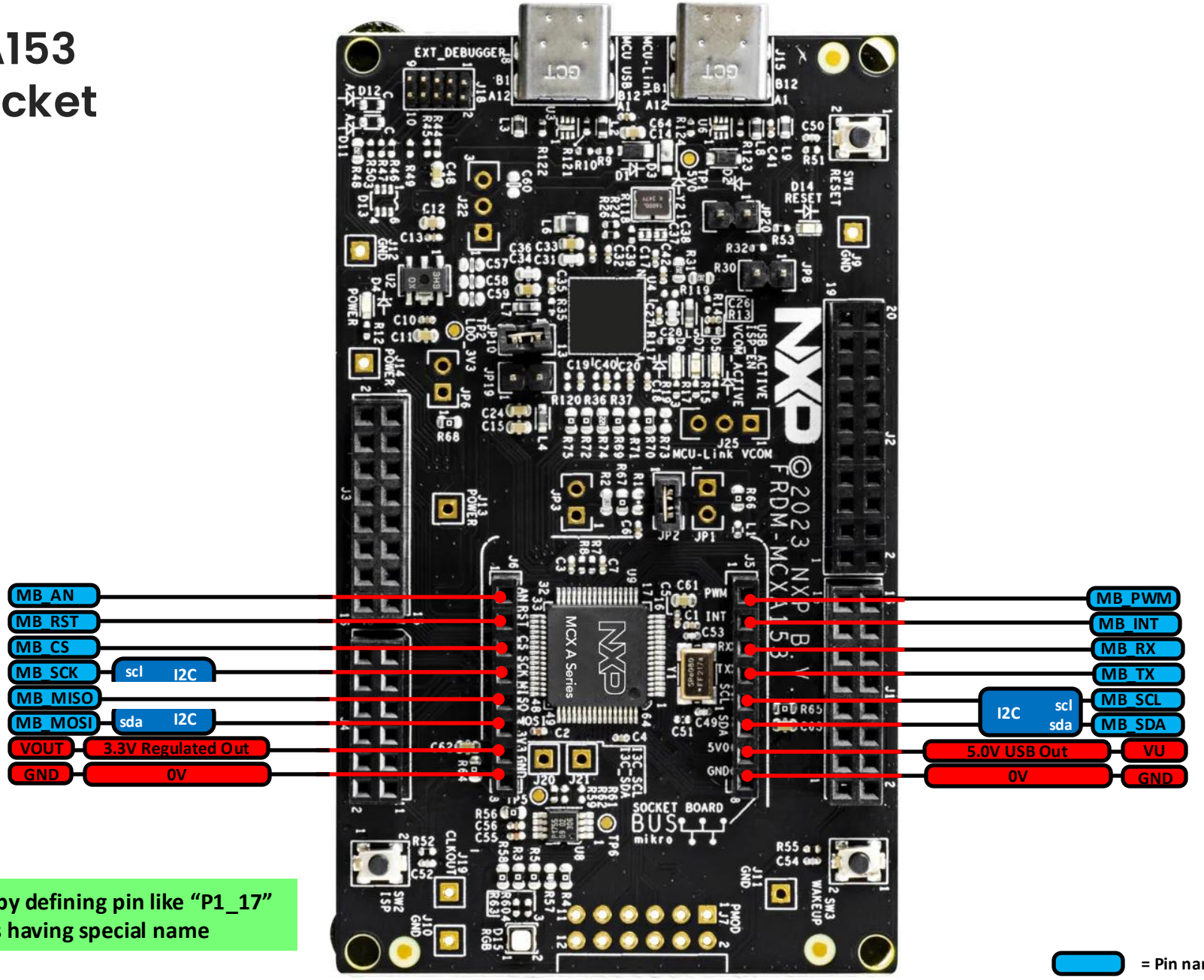
FRDM-MCXA153
Arduino socket



The I3C can be routed to D18 and D19 pins by declaring as `I3C( I2C_SDA, I2C_SCL )`

All pin sockets can be used by defining pin like “P1_17”
This page shows pin sockets having special name

FRDM-MCXA153 mikroBUS socket



All pin sockets can be used by defining pin like “P1_17”
This page shows pin sockets having special name

= Pin names which can be used in the code

Available pins
A153

r01lib pin names								
pin definition				class				
pin name	other pin name (1)	other pin name (2)	other pin name (3)	Digital&Bus InOut	InterruptIn	I2C	I3C	SPI
D0			P1_4	✅	✅			
D1			P1_5	✅	✅			
D2			P2_4	✅	✅			
D3	BLUE		P3_0	✅	✅			
D4			P2_5	✅	✅			
D5	RED		P3_12	✅	✅			
D6	GREEN		P3_13	✅	✅			
D7			P3_1	✅	✅			
D8			P3_15	✅	✅			
D9			P3_14	✅	✅			
D10		SPI_CS	P2_6	✅	✅			
D11		SPI_MOSI	P2_13	✅	✅			✅
D12		SPI_MISO	P2_16	✅	✅			
D13		SPI_SCLK	P2_12	✅	✅			
D18		I2C_SDA	P1_8	✅	✅	✅	✅	
D19		I2C_SCL	P1_9	✅	✅			
A0			P1_10	✅	✅			
A1			P1_12	✅	✅			
A2			P1_13	✅	✅			
A3			P2_0	✅	✅			
A4			P3_31	✅	✅			
A5			P3_30	✅	✅			
MB_AN			P3_30	✅	✅			
MB_RST			P3_1	✅	✅			
MB_CS			P1_3	✅	✅			
MB_SCK			P1_1	✅	✅	✅ with MK_MOSI		
MB_MISO			P1_2	✅	✅			
MB_MOSI			P1_0	✅	✅	✅ with MK_SCK		
MB_PWM			P3_12	✅	✅			
MB_INT			P2_5	✅	✅			
MB_RX			P3_14	✅	✅			
MB_TX			P3_15	✅	✅			
MB_SCL			P3_27	✅	✅	✅		
MB_SDA			P3_28	✅	✅			
		I3C_SDA	P0_16	✅	✅		✅	
		I3C_SCL	P0_17	✅	✅			
	SW2		P3_29	✅	✅			
	SW3		P1_7	✅	✅			

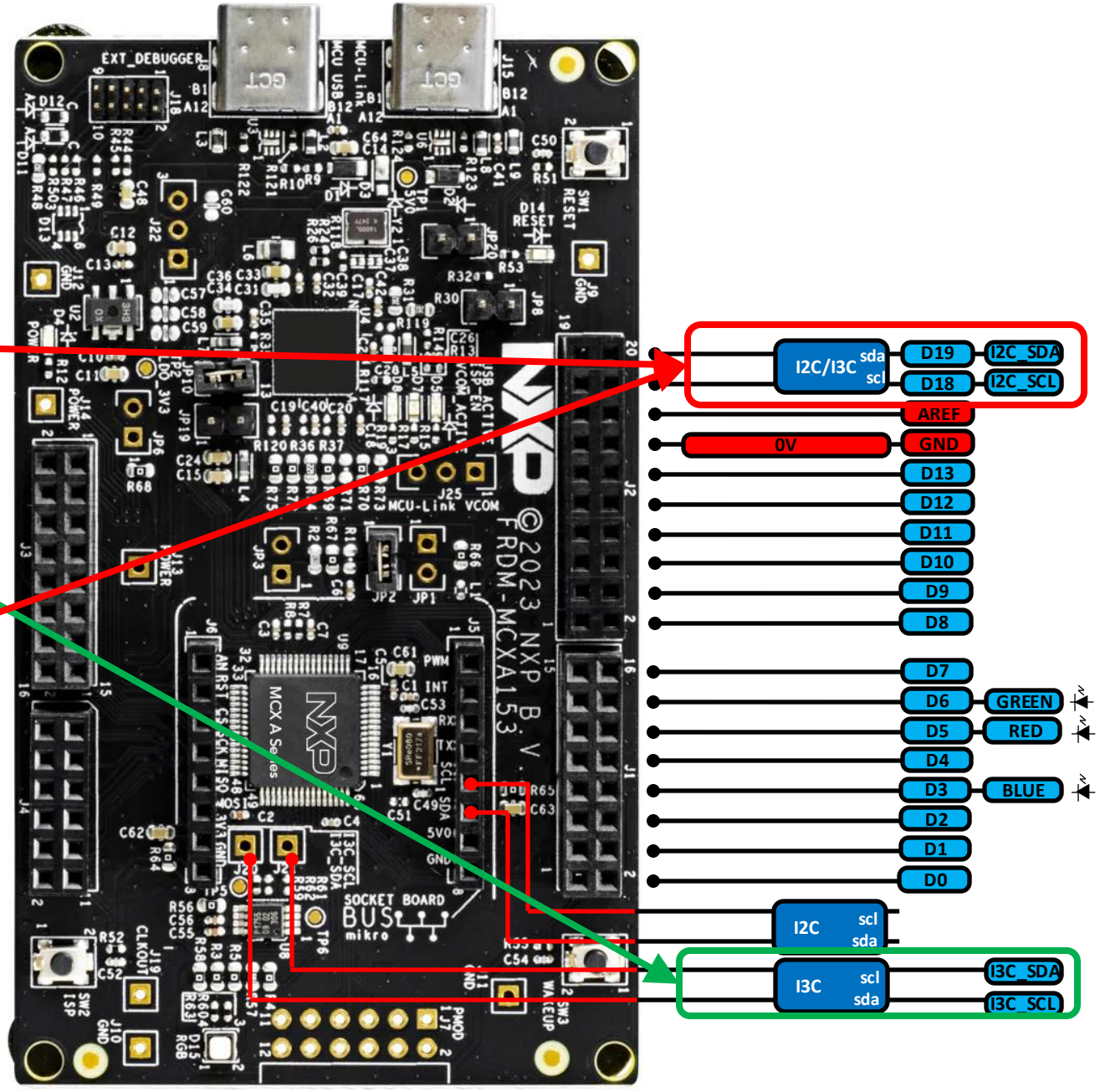
Example of defining I3C pins from code on FRDM-MCXA153

```
#include "r01lib.h"

I2C i2c( I2C_SDA, I2C_SCL );
```

```
#include "r01lib.h"

I3C i3c( I3C_SDA, I3C_SCL );
```

```
#include "r01lib.h"

I3C i3c( I2C_SDA, I2C_SCL );
```

For instance, in I3C case, the signals can be changed to re-route the signals to D18 and D19 by giving arguments for constructor.



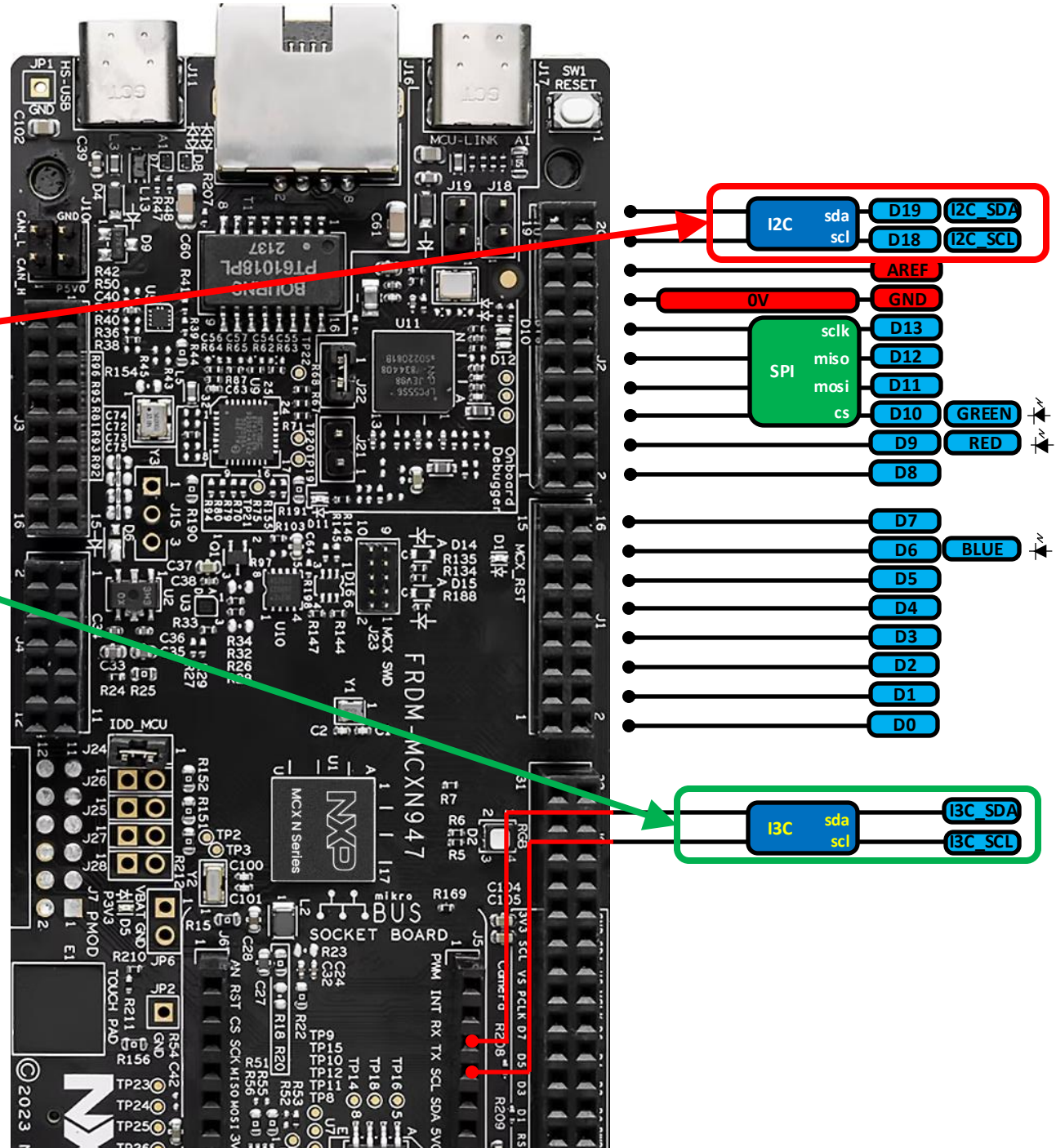
Example of defining I3C pins from code on FRDM-MCXN947

```
#include "r01lib.h"

I2C i2c( I2C_SDA, I2C_SCL );
```

```
#include "r01lib.h"

I3C i3c( I3C_SDA, I3C_SCL );
```

```
#include "r01lib.h"

I3C i3c( I2C_SDA, I2C_SCL );
```

On FRDM-MCXN947, the I3C signals cannot be assigned to D18 and D19 because hardware doesn't allow this configuration



03

r01device

R01 product operation sample library

Library for R01 products operation on MCUXpresso

Simple interface to write sample code

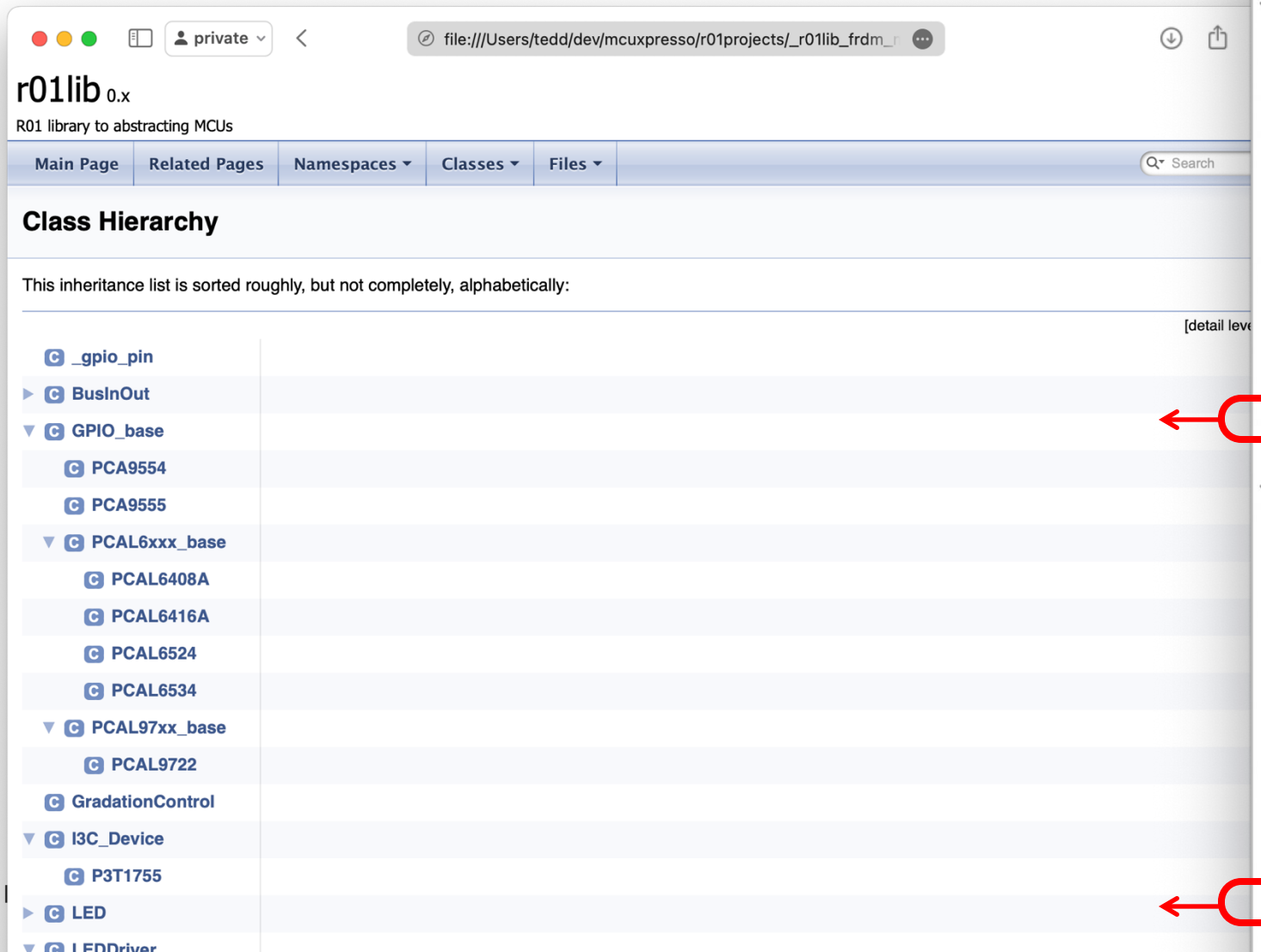
- **r01device** = A class driver correction for **r01 device** operations

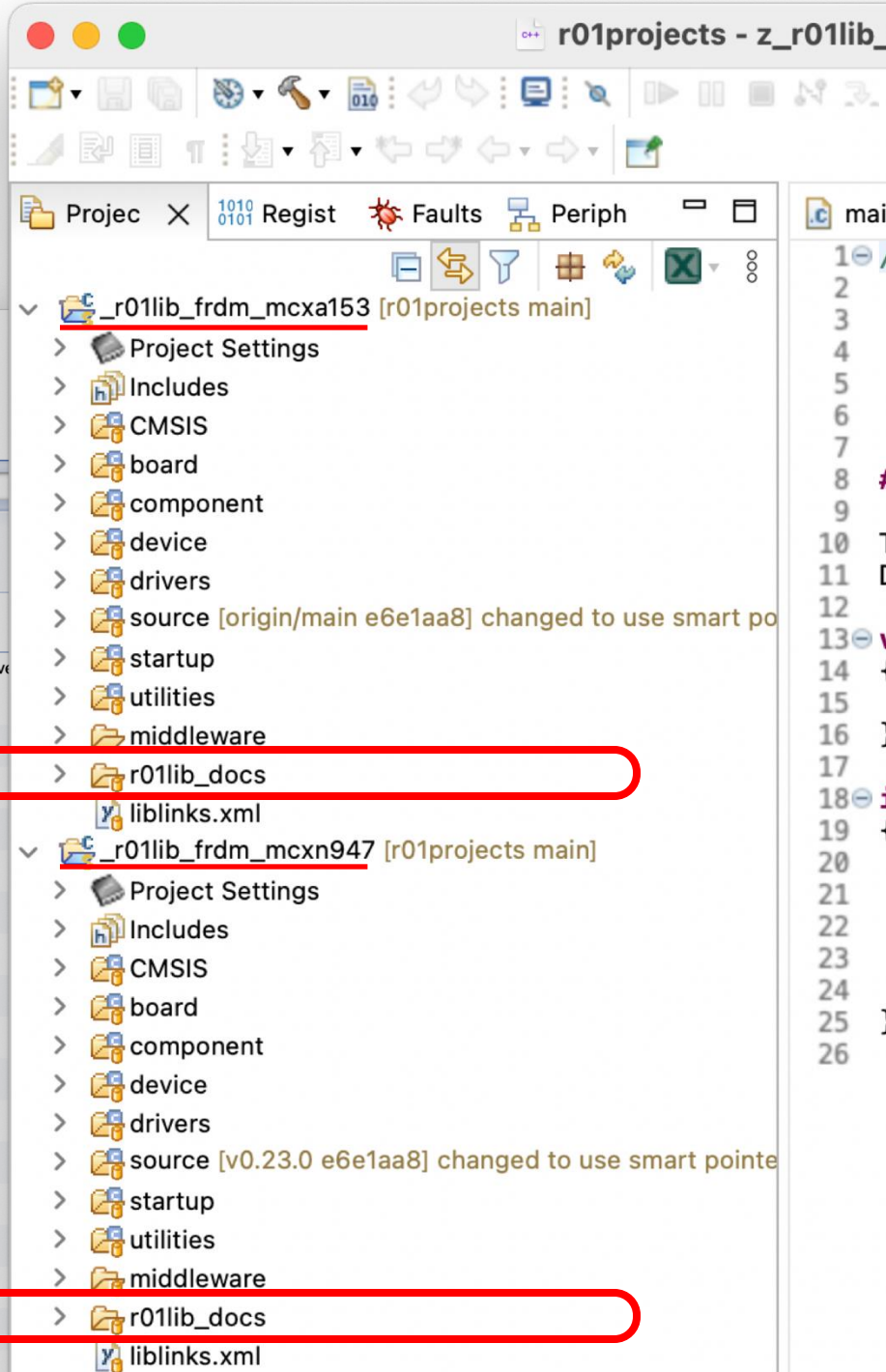
- Delivered as a part of **r01lib**

- Abstracting device operation
  - To present devices' main features with simple interfaces
  - User don't need to care register level operation
  - If user need to access register, it can be done through register access APIs

Available device classes

All available classes can be found on online document





04

r01device operation sample

Simple API

Sample 1 : P3T1755
Interface to temp sensor

```
#include "r01lib.h"

I3C     i3c( I3C_SDA, I3C_SCL );
P3T1755 p3t1755( i3c );

int main(void)
{
    float temp;

    i3c.ccc_broadcast( CCC::BROADCAST_RSTDAA, NULL, 0 );
    i3c.ccc_set( CCC::DIRECT_SETDASA, 0x48, 0x08 << 1 );
    p3t1755.address_overwrite( 0x08 );

    while ( true )
    {
        temp = p3t1755;
        printf( "Temperature: %8.4f°C\r\n", temp );
        wait( 1 );
    }
}
```

- P3T1755 class makes an interface for the I3C device
- Register level access is abstracted

- This sample is using CCC interface in I3C class

Sample 2 : PCA9955B
Interface to LED driver

```
#include "r01lib.h"

#include "led/PCA9955B.h"

I2C      i2c( I2C_SDA, I2C_SCL );
PCA9955B ledd( i2c );

int main(void)
{
    ledd.begin( 1.0, PCA9955B::ARDUINO_SHIELD );

    while ( true )
    {
        ledd.pwm( 0, 1.0 );
        wait( 0.1 );
        ledd.pwm( 0, 0.0 );
        wait( 0.1 );
    }
}
```

- PCA9955B class makes an interface for the LED driver
- PWM can be controlled through instance method

Sample 3 : PCF2131
Interface to RTC

```
#include "r01lib.h"
#include <time.h>
#include "rtc/PCF2131_I2C.h”

I2C         i2c( I2C_SDA, I2C_SCL );
PCF2131_I2C rtc( i2c );

int main(void)
{
    while ( true )
    {
        time_t current_time = rtc.time( NULL );

        printf( "time: %s", ctime(&current_time) );
        wait( 1 );
    }
}
```

- PCF2131 class makes an interface for the RTC
- The interface data format is compatible with “time.h” in C standard lib.

05

Repository

Access to the code

Repo

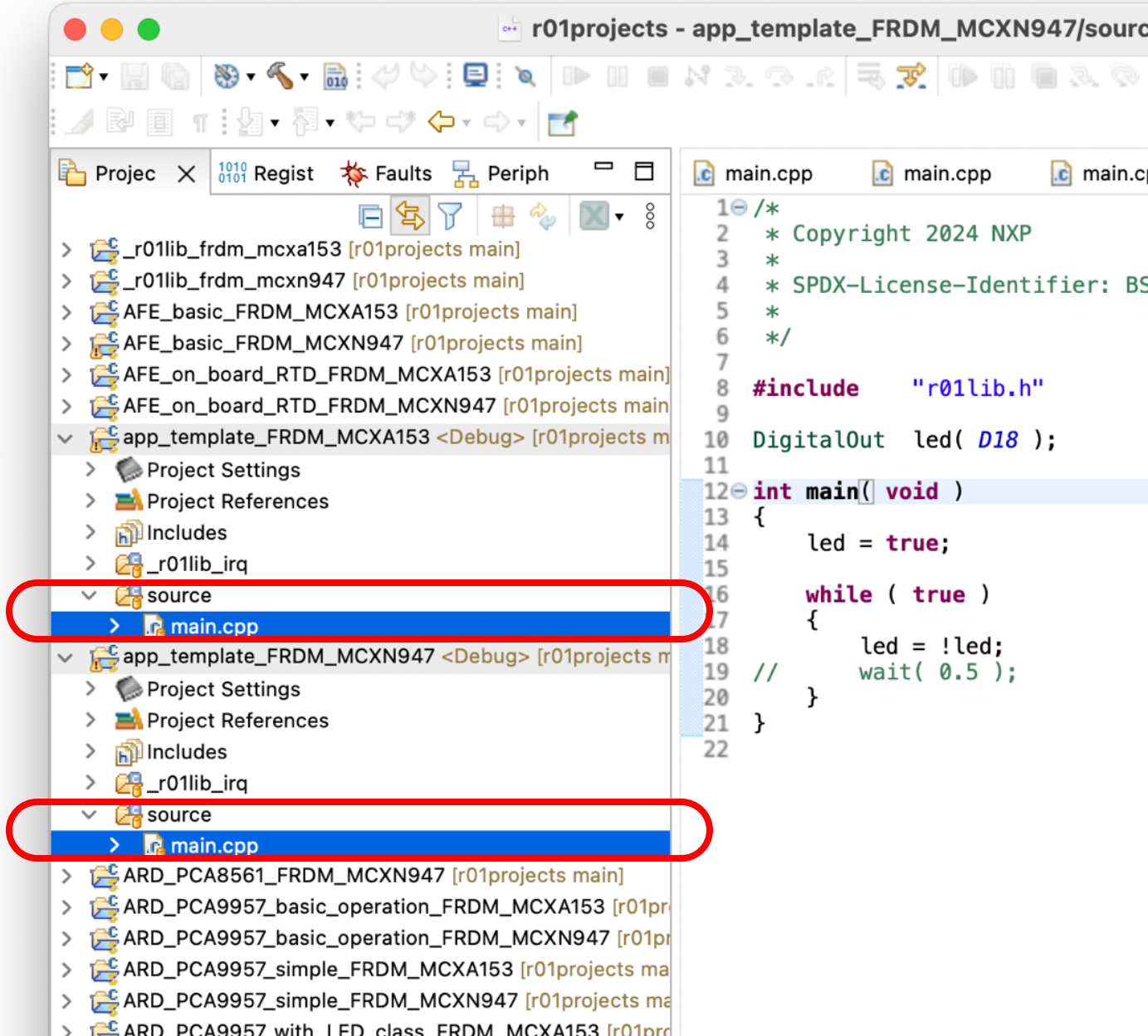
- https://github.com/teddokano/r01lib_MCUXpresso
  - Including both r01lib and r01device source code
  - This repo including sample/test code for available features
    - DigitalInOut, DigirtalIn, DigitalOut, BusInOut, BusIn, BusOut, InterruptIn, Ticker, I3C, I2C, SPI classes

- The device usage samples with ‘r01device’ are available in https://github.com/teddokano/r01projects
  - This repo is hidden by some reasons. But all you are welcome to join to see this. Please let OKANO know your GitHub to invite!
  - Sample code to operate ..
    - NAFE13388, PCA8561, PCA9957, PCAL6534, PCAL9722, PCF2131(I²C&SPI), P3T1755

How to start writing your own code
Write your code

Choose template project and overwrite **`source/main.cpp`** .

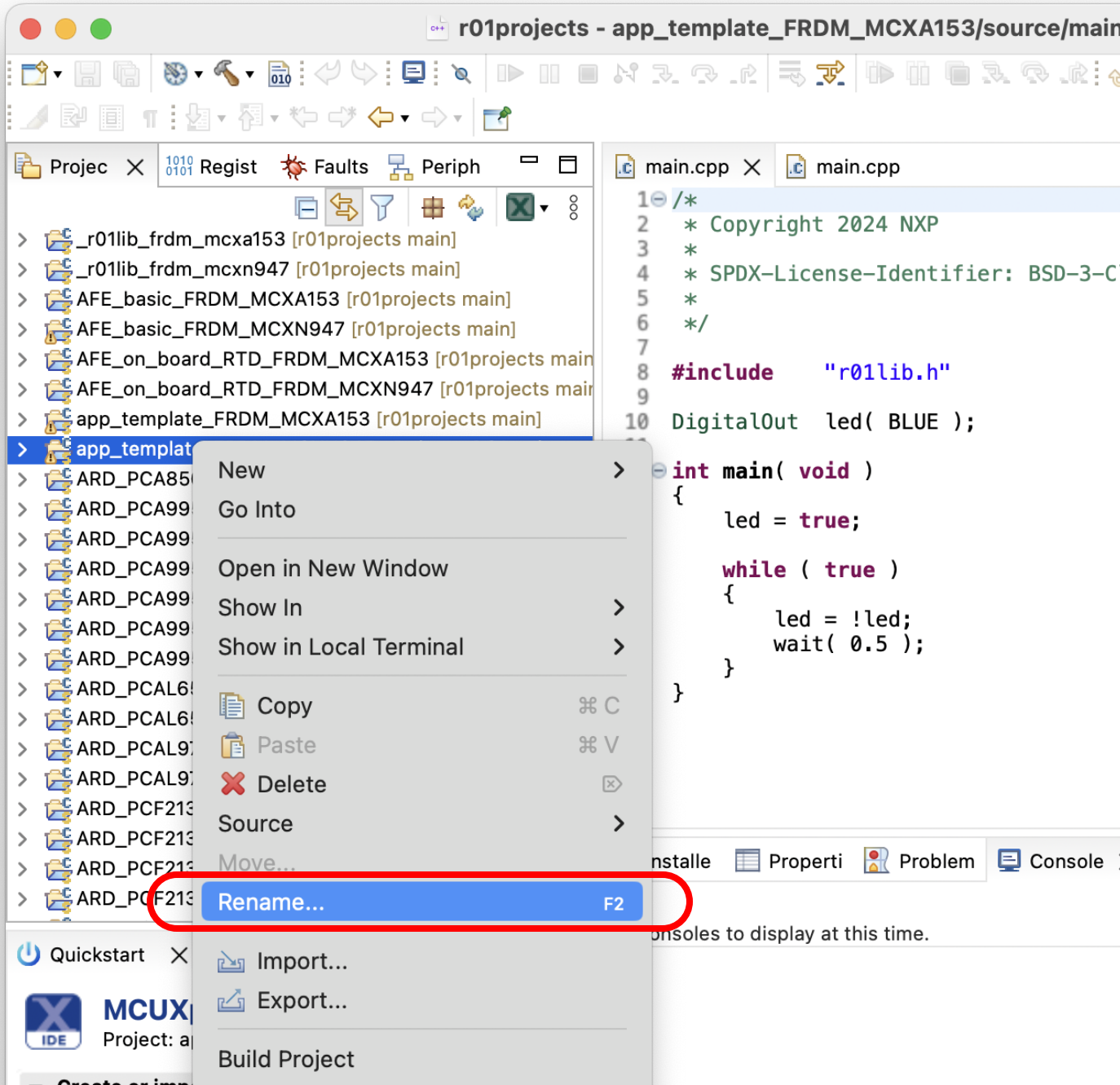
If needed, new files (.c/.cpp/.h) can be added in **`source/`** folder.



How to start writing your own code
Code exporting: step1

Rename the project

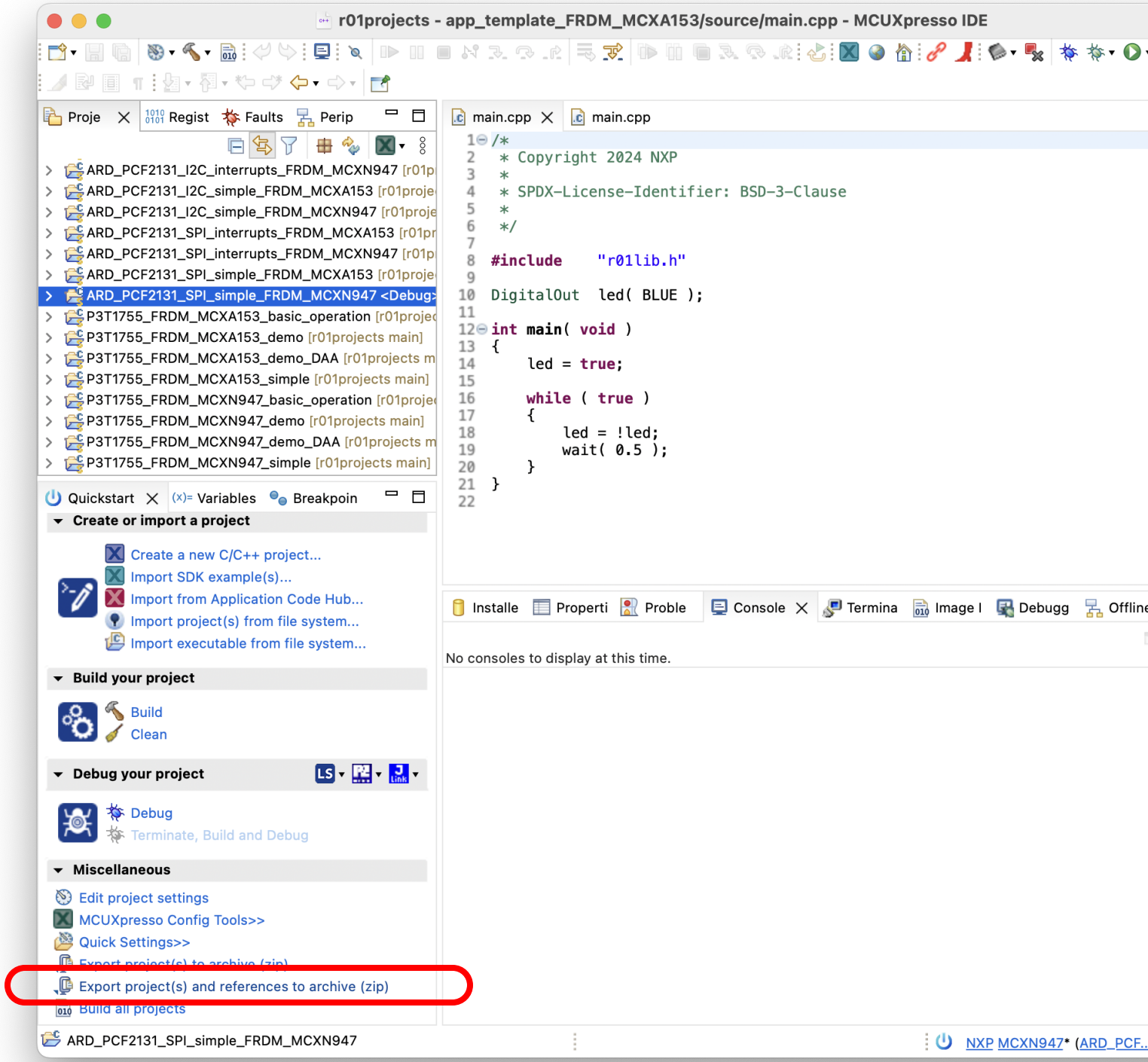
Right click on the project and choose “Rename…” from pop-up menu



Code distribution: step2

Choose the project(s)

Click on “Export project(s) and reference to archive (zip)”



06

History

Document revision history

Date	Updates
2024 Mar 21	Initial version
2024 May 29	Pin name information added
2024 Jul 17	FRDM-MCXN236 support added
2024 Dec 09	FRDM-MCXA156 support added

nxp.com