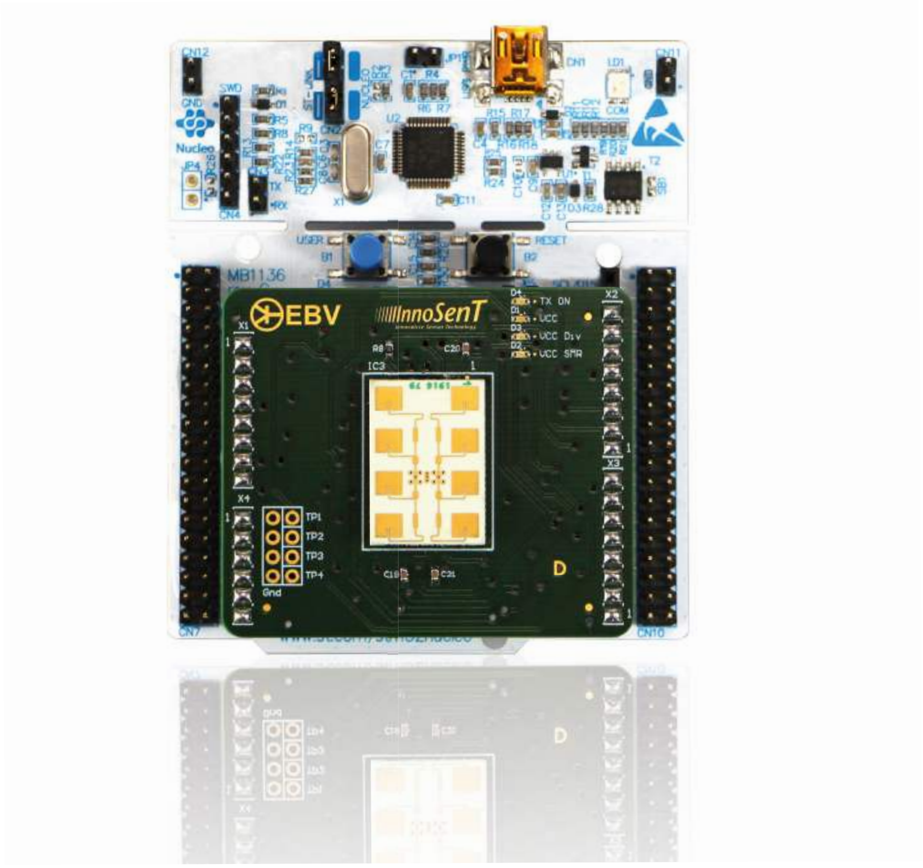
SMR Evaluation Kit User Manual



Content

1. Introduction

Overview

The Evaluation Kit “EVALKIT SMR-334” is a programmable compact radar system with a focus on creating a friendly and easy-to-use FMCW radar platform. It comes with a custom and open firmware that provides beginners with an easy way to experience radar sensors in their application and take a look into the signal processing with our detailed examples.

The SMR-Eval Kit is built around the SMR sensor family and comes equipped with our flagship module SMR-334. The underlying STM32F401RE Nucleo Board from ST provides the necessary processing power and interface to the IDE and GUI. The kit comes with all necessary hardware, software and documentation to quickly start evaluating innovative radar technology.

Features

- Supports Doppler and FMCW radar principle
- VCO controlled with 16-bit DAC on the SMR interface module
- Stereo receiver
- 12bit data acquisition
- Configurable Doppler frequency and FMCW bandwidth
- Frequency auto-calibration for compliance with government regulations
- Simple GUI to visualize receive signals
- ADC raw data and FFT display
- Configurable radar parameters for advanced users
    - Bandwidth
    - ADC sampling rate
    - Frequency auto-calibration enable
    - Etc.
- Full SMR antenna module control
- System power and communication via a single USB interface
- Source code with detailed comments giving users full control to the Evaluation Kit

2. Installation

Hardware Installation

The SMR-EvalKit packet consists of:

- SMR antenna module
- STM32F401RE-Nucleoboard
- Mini USB cable
- Download link on the InnoSenT Homepage: Project files with Source code, GUI, User Manual, Quickstart Software, SMR Data Sheet)

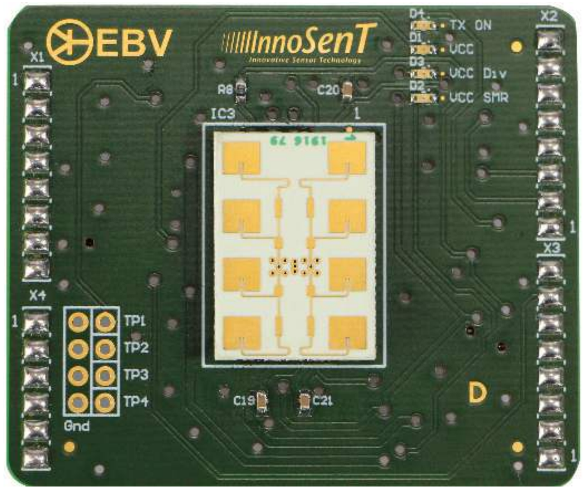


Figure 1: SMR Antenna module

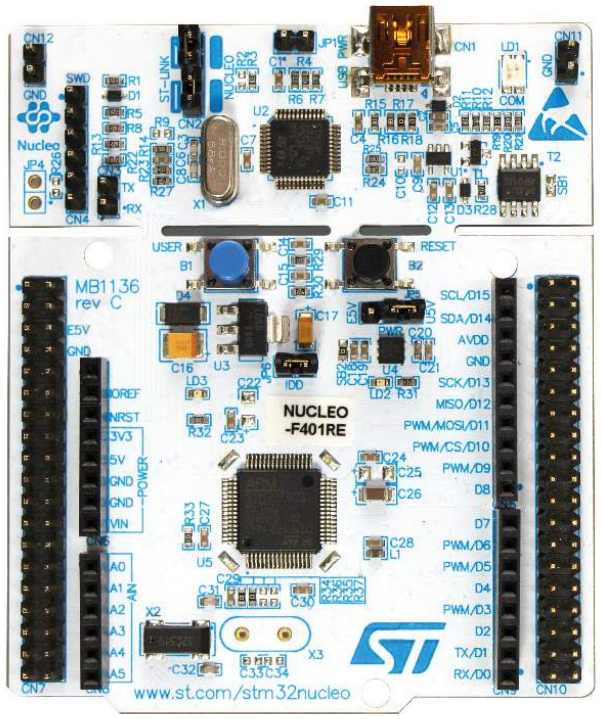


Figure 2: STM32F401RE-Nucleoboard

In order to have a proper hardware installation, please follow the instructions below:

- The SMR Evaluation Kit comes preassembled, however it is also possible to disassemble the two boards.
- When the two boards are separated and need to be assembled, connect them as in the following picture. Note: before connecting the two boards please make sure the power supply to the STM32F401RE board (via USB cable) is disconnected in order to prevent any damage to the system due to wrong connection and the connection should be checked again before applying the power supply.
- Jumpers on the microcontroller board should be left unchanged.

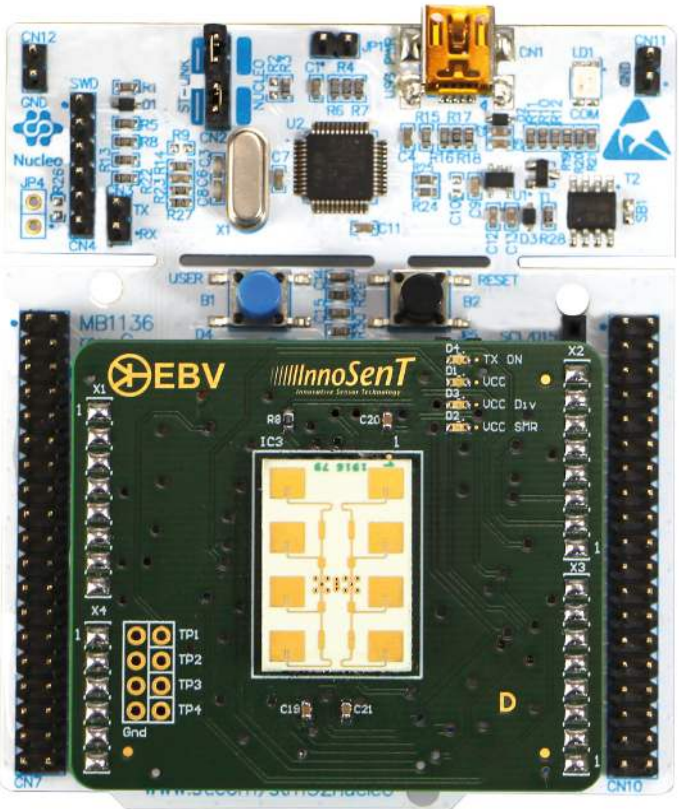


Figure 3: SMR-board and STM32 board connection

Software Installation

Please refer to the SMR Evaluation Kit quick start guide for instructions about the software installation.

3. Brief Introduction to Radar Principles

Doppler Principle

Basis elements

The CW-Doppler radar is the simplest kind of Radar, but most effective for detecting motion and measuring speed. It utilizes the Doppler Effect, an effect which applies to all sorts of wave generators and says the following:

Wave fronts, transmitted by a wave generator (sound, microwaves, light etc.) hit a moving target. Depending on the direction of the motion of this object, the wave fronts are either "compressed" or "stretched", which finally results in a shift in frequency. The received signal is mixed with the unchanged transmit signal in the receiver (called "homodyne" mixing), which results in a sinusoidal intermediate frequency (IF). It doesn't matter whether the sensor moves relatively to the object or the object moves relatively to the sensor.

As a matter of fact, only the radial component of the velocity vector can be detected. The mathematical formula looks as follows:

$$f_{\mathrm{Doppler}} = 2f_0 * \frac{v}{C_0} * \cos\alpha$$

f_D	Doppler- or differential frequency
f_0	Transmit frequency of the radar
v	velocity of the moving object
C_0	Speed of light
α	Angle between the actual direction of motion and the connecting line sensor-object

Selecting 24GHz as transmit frequency, the following rule of thumb applies:

$$f_{\mathrm{Doppler}} = 44\frac{\mathrm{Hz}}{\mathrm{km/h}} * v * \cos\alpha$$

With this simple equation the expected Doppler frequency can easily be calculated and the parameters of the IF-filter and amplifier can be defined.

For instance it is not practical to design the upper frequency limit of the signal conditioning part of a unit detecting human beings much higher than 300Hz, since this corresponds to a speed of 6.8 km/h of a (pretty fast) pedestrian. On the other hand, when using radar sensors to check the speed of cars for example on a German motorway, the amplifier needs to have an upper frequency limit of at least 10 kHz corresponding to 220 km/h.

As a summary the speed of an object can be evaluated by measuring the Doppler frequency (in an analog system by counting the zero crossings or in digital system by frequency analysis with FFT), while considering the angle of the motion vector.

Please note:
In the very rare case of a perfect circular motion of the object around the sensor, the angle α would be 90o, which causes the cosine and therefore the Doppler frequency to drop to zero. This specific motion won't be detected by this type of radar. However this object would have to move along this circle with absolute perfection, which is highly unlikely for a real life object.

Identification of direction of motion

Radar sensors with dual IF output can provide information about the direction of motion (leaving or approaching) simply by utilizing two mixer circuits, which are spaced by a quarter wavelength, called an I(n phase)/Q(uadrature phase) mixer.

This information is useful for example in door opener applications, where the door should open only when a person approaches it.

For detailed information on the radar basics, see Application_Note I-IV on the InnoSenT homepage.

Example

The calculation of the parameters of the implemented CW-Doppler example is shown below:

speed of light	C0	299792458	m/s
CW-Frequency	F	depending on FREQ_START	Hz
		Example: 24.000	GHz
sampling period	Ta	0.0004	s
		400	µs
buffer length (number of samples)	N	128	
sampling time	Tsample	0.0512	s
		51.2	ms
velocity resolution	Vmin	0.122	m/s
		0.44	km/h
low pass filter (theoretical)	LP	1250	Hz
low pass filter (current hardware)		915	Hz
high pass filter (theoretical)	HP	19.53	Hz
high pass filter (current hardware)		20	Hz

$$\mathbf{Tsample} = \mathrm{Ta} * \mathrm{N} = 400\mu\mathrm{s} * 128 = \mathbf{51.2ms}$$

$$\mathbf{Vmin} = \frac{C_0}{2 * \mathrm{Tsample} * \mathrm{F}} = \frac{299792458\mathrm{m/s}}{2 * 51.2\mathrm{ms} * 24.000\mathrm{GHz}} = \mathbf{0.122}\boldsymbol{ms}$$

$$\mathbf{Vmax} = \mathrm{Vmin} * \left(\frac{\mathrm{N}}{2} - 1\right) = 0.122ms * \left(\frac{128}{2} - 1\right) = \mathbf{7.68}\boldsymbol{ms}$$

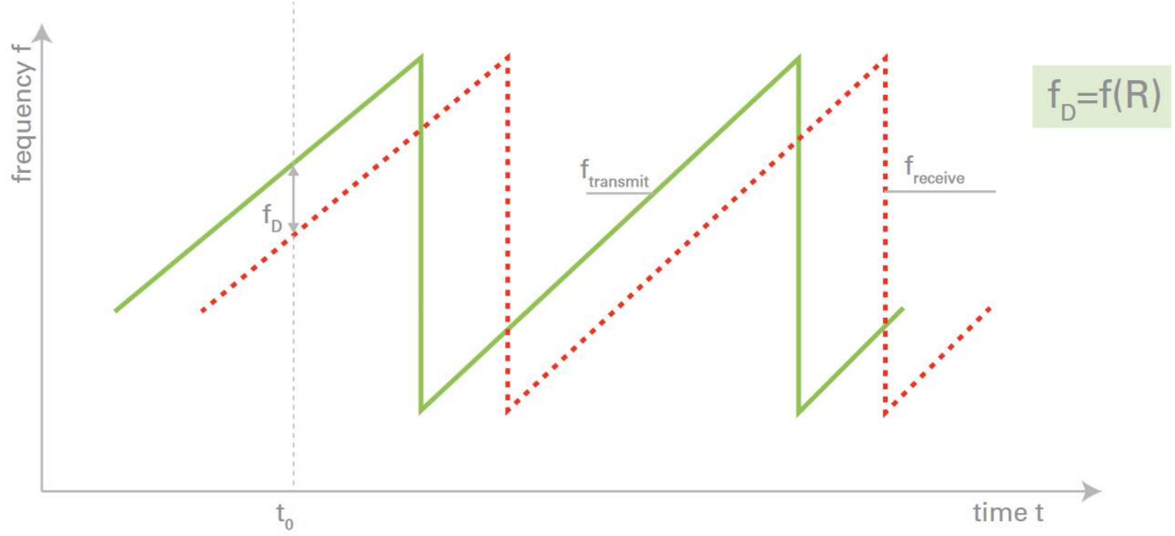
$$\mathbf{LP} = \frac{1}{2 * Ta} = \frac{1}{2 * 400\mu\mathrm{s}} = \mathbf{1250}\boldsymbol{Hz}$$

$$\mathbf{HP} = \frac{1}{Tsample} = \frac{1}{51.2} = \mathbf{19.53}\boldsymbol{Hz}$$

4. FMCW Principle

Basis elements

The FMCW-(Frequency-Modulated-Continuous-Wave) radar is a common approach to detect stationary objects.



Time-dependent shape of transmit and receive signal of a FMCW radar with sawtooth modulation scheme

Unlike a Pulse Radar the FMCW sensor emits a continuous wave changing the frequency linearly over time. Due to the propagation delay the received signal shows a slightly different frequency compared to the presently generated signal leading to a beat frequency in the receiver, which is proportional to the travelling time.

The following equation describes the relation in case of a sawtooth-modulation:

$$\mathrm{R} = \frac{\mathrm{C_0}}{2} * \mathrm{T} * \frac{\mathrm{f_D}}{\Delta\mathrm{f}}$$

f_D	differential frequency
Δf	frequency deviation (bandwidth)
T	sawtooth prepetition time period
R	distance of a reflecting object
C_0	speed of light

For the 24GHz-ISM-Band the modulation bandwidth is limited by regulation to 250MHz at maximum. In consequence the range resolution is limited to 60cm as a theoretical value. In practice ranging for distances greater than 2m is possible with simple data processing. Closer ranges are possible using more complex algorithms.

For more detailed information on the radar basics, see Application_Note I-IV on the InnoSenT homepage.

Crosstalk

A typical problem of FMCW radar is the crosstalk between transmitter and receiver due to non-ideal isolation between them. The crosstalk effect causes blind zone approximately 3m from sensor at 24GHz. This blind zone can be reduced by applying Highpass filtering, but it cannot be eliminated entirely.

Example

The calculation of the parameters of the implemented FMCW-modulation example is shown below:

speed of light	C0	299792458	m/s
bandwidth	B	depending on FREQ_ START and FREQ_STOP	Hz
		Example: 250	MHz
sampling period	Ta	0.0004	s
		400	µs
puffer length (number of samples)	N	128	
sampling time (modulation time)	Tsample	0.0512	s
		51.2	ms
range resolution	Rmin	0.60	m
maximum range (theoretical)	Rmax	37.8	m
low pass filter (theoretical)	LP	1250	Hz
low pass filter (current hardware)		915	Hz
high pass filter (theoretical)	HP	19.53	Hz
high pass filter (current hardware)		20	Hz

FREQ_START = 24.000GHz

FREQ_STOP = 24.250GHz

$$\mathbf{B} = \mathrm{FREQ}_{\mathrm{STOP}} - \mathrm{FREQ}_{\mathrm{START}} = 24.000\mathrm{GHz} - 24.250\mathrm{GHz} = \mathbf{250MHz}$$

$$\mathbf{Tsample} = \mathrm{Ta} * \mathrm{N} = 400\mu\mathrm{s} * 128 = \mathbf{51.2ms}$$

$$\mathbf{Rmin} = \frac{\mathrm{C}_0}{2 * \mathrm{B}} = \frac{299792458\mathrm{m/s}}{2 * 250\mathrm{MHz}} = \boldsymbol{0.60m}$$

$$\mathbf{Rmax} = \mathbf{Rmin} * \left(\frac{\mathrm{N}}{2} - 1\right) = 0.60m * \left(\frac{128}{2} - 1\right) = \boldsymbol{37.8m}$$

$$\mathbf{LP} = \frac{1}{2 * ta} = \frac{1}{2 * 400\mu s} = \boldsymbol{1250Hz}$$

$$\mathbf{HP} = \frac{1}{\mathrm{Tsample}} = \frac{1}{51.2} = \boldsymbol{19.53Hz}$$

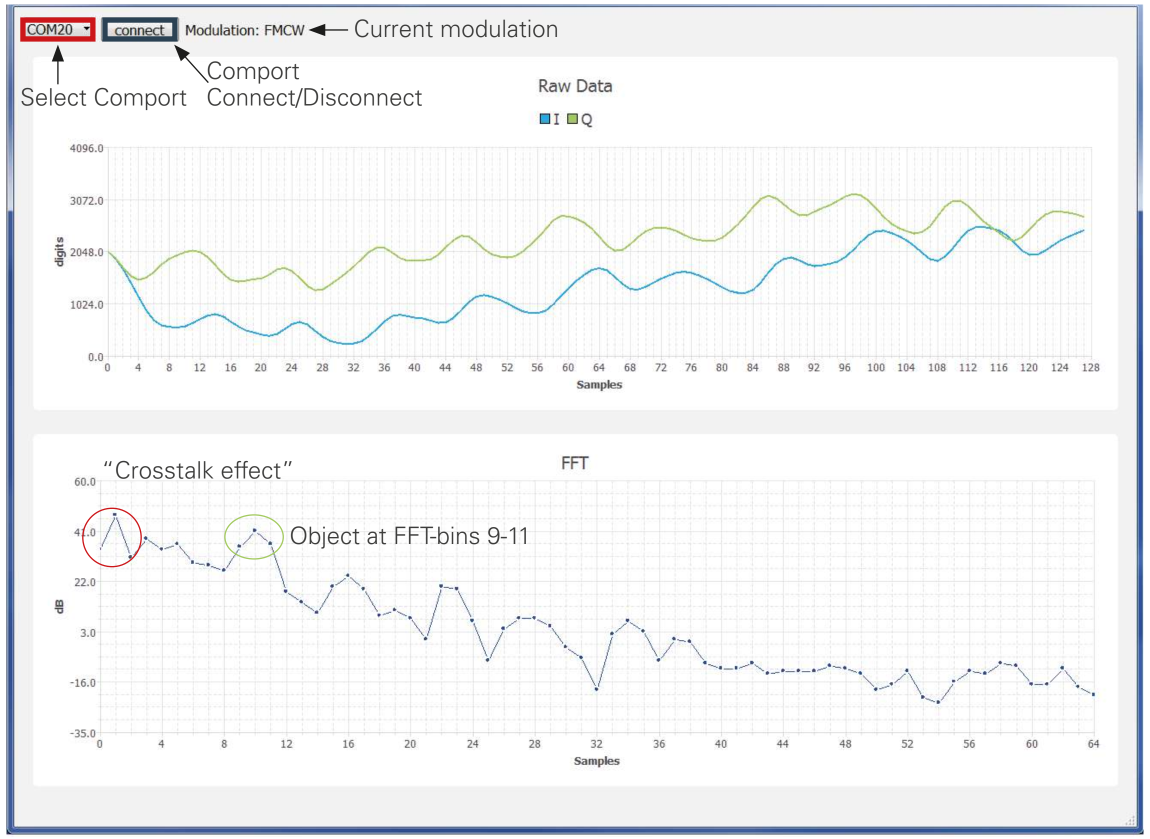
5. Getting Started

Receive Signals Visualization

The SMR-EvalKit is a plug-and-play device. The system will boot the previously uploaded firmware as soon as it receives power via its USB port and the IF signals can be visualized using the SMR EvalKit GUI.
To visualize receive signals:

- open the SMR EvalKit GUI
- select corresponding Comport of SMR EvalKit
- Click Connect/Disconnect button to **connect** or **disconnect** to SMR EvalKit Comport

In the GUI, the upper graph shows the raw receive signals on I- and Q-channel. The magnitudes are scaled in digit values. The bottom graph shows the corresponding FFT of raw receive signals and is displayed logarithmic (dB).



Note: When starting SMR-Evalkit, it may take a few seconds to perform initial frequency calibration and there is no data being transmitted to the PC during this time. It might take a few moments until a signal is displayed in the GUI.

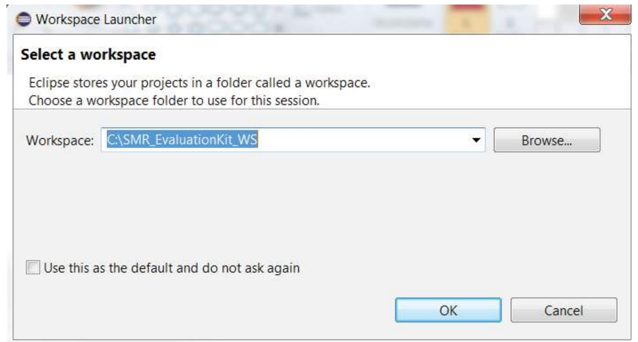
Radar Setting Modification

By modifying and uploading the source code onto the microcontroller board, the radar settings can be changed to different configurations. The following instructions show the procedure for doing this:

a) Create a workspace folder for the project e.g: "C:\SMR_EvaluationKit_WS"

b) Copy the SMR EvalKit firmware project folder into the workspace

c) Run SW4STM32 software and select the created workspace folder



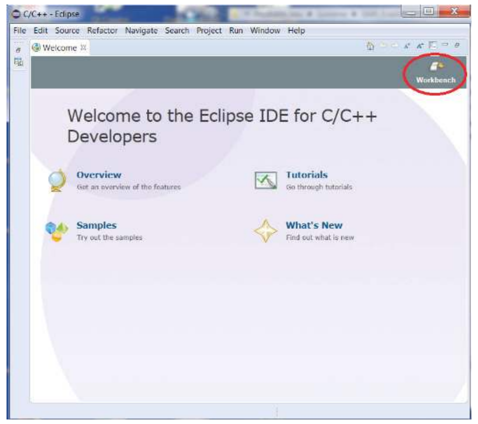
Select a workspace

Eclipse stores your projects in a folder called a workspace.
Choose a workspace folder to use for this session.

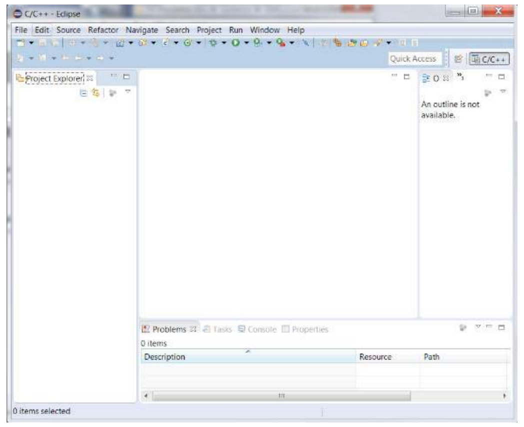
Workspace: C:\SMR_EvaluationKit_WS

□ Use this as the default and do not ask again

d) Go to workbench


C/C++ - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Welcome
Workbench
Overview
Tutorials
Samples
What's New

Welcome to the Eclipse IDE for C/C++ Developers

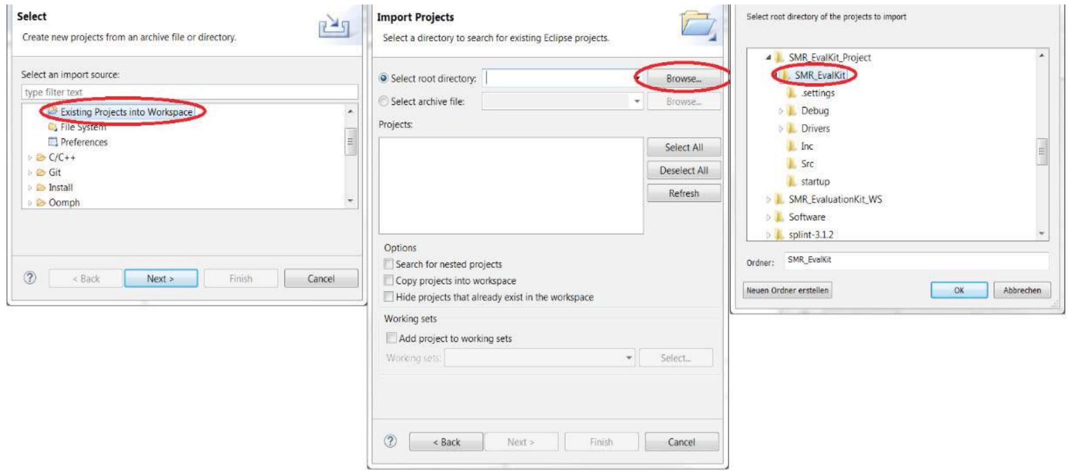


0 items

An outline is not available.

0 items selected

a) Import the project containing the source code into the workspace. In the Project Explorer tab -> Right Click -> Import -> General -> Existing Projects into Workspace -> Select root directory -> Brower -> "select directory where the source code project located" -> OK -> Finish



Select

Create new projects from an archive file or directory.

Select an import source:

type filter text

Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory:

○ Select archive file:

Projects:

Options

□ Search for nested projects

□ Copy projects into workspace

□ Hide projects that already exist in the workspace

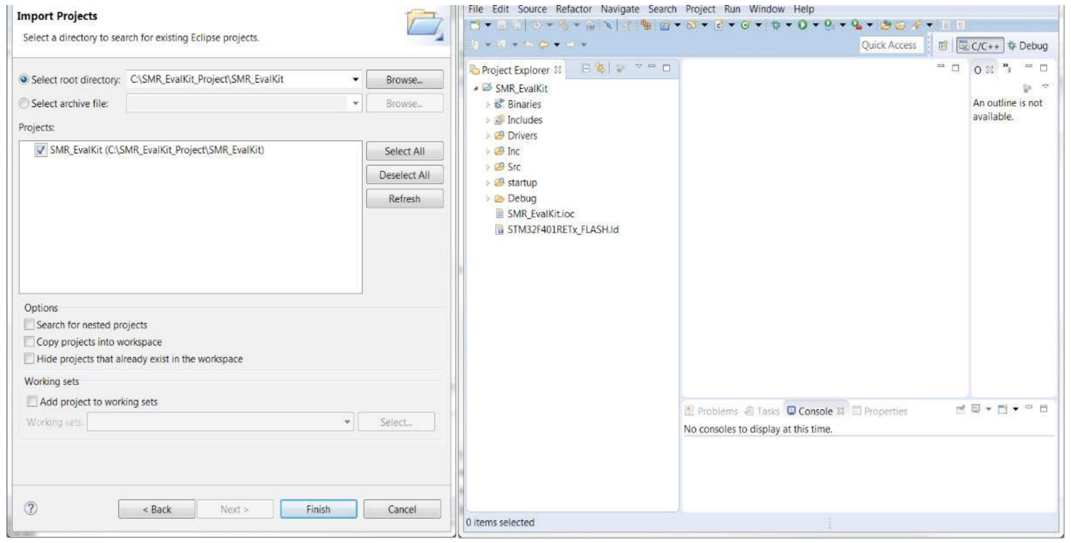
Working sets

□ Add project to working sets

Working sets:

Select root directory of the projects to import

Ordner: SMR_EvalKit



Import Projects

Select a directory to search for existing Eclipse projects.

● Select root directory: C:\SMR_EvalKit_Project\SMR_EvalKit

○ Select archive file:

Projects:

☑ SMR_EvalKit (C:\SMR_EvalKit_Project\SMR_EvalKit)

Options

□ Search for nested projects

□ Copy projects into workspace

□ Hide projects that already exist in the workspace

Working sets

□ Add project to working sets

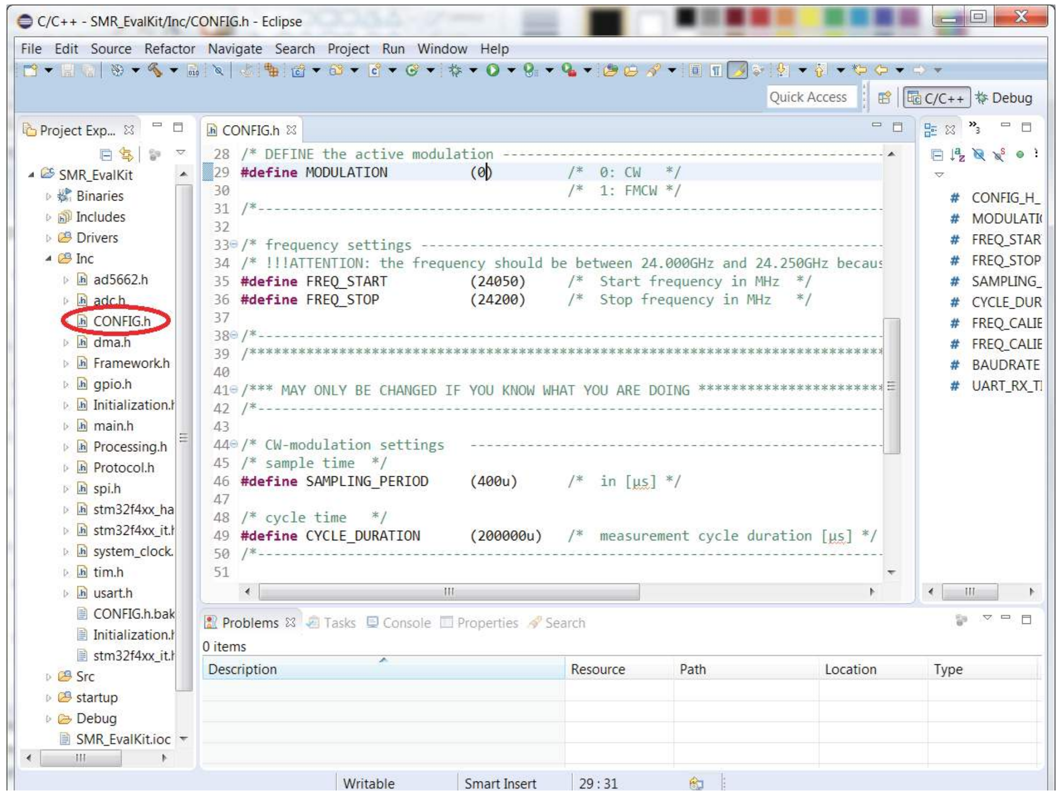
Working sets:

An outline is not available.

No consoles to display at this time.

0 items selected

b) Expand the project and open the file "CONFIG.h" in "Inc" folder

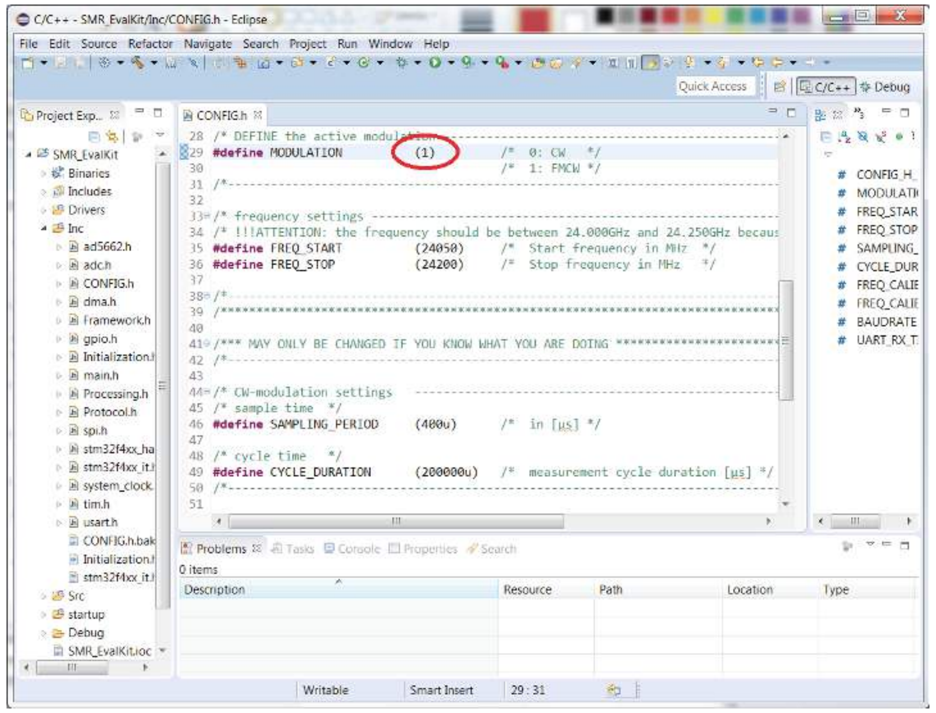


```
/* DEFINE the active modulation ---------------------------------------------
#define MODULATION          (0)         /*  0: CW   */
                                        /*  1: FMCW */
/*--------------------------------------------------------------------------

/* frequency settings -------------------------------------------------------
/* !!!ATTENTION: the frequency should be between 24.000GHz and 24.250GHz becaus
#define FREQ_START          (24050)     /*  Start frequency in MHz  */
#define FREQ_STOP           (24200)     /*  Stop frequency in MHz   */

/*--------------------------------------------------------------------------
/***************************************************************************

/*** MAY ONLY BE CHANGED IF YOU KNOW WHAT YOU ARE DOING *********************
/*--------------------------------------------------------------------------

/* CW-modulation settings   ------------------------------------------------
/* sample time  */
#define SAMPLING_PERIOD     (400u)      /*  in [µs] */

/* cycle time   */
#define CYCLE_DURATION      (200000u)   /*  measurement cycle duration [µs] */
/*--------------------------------------------------------------------------
```

0 items

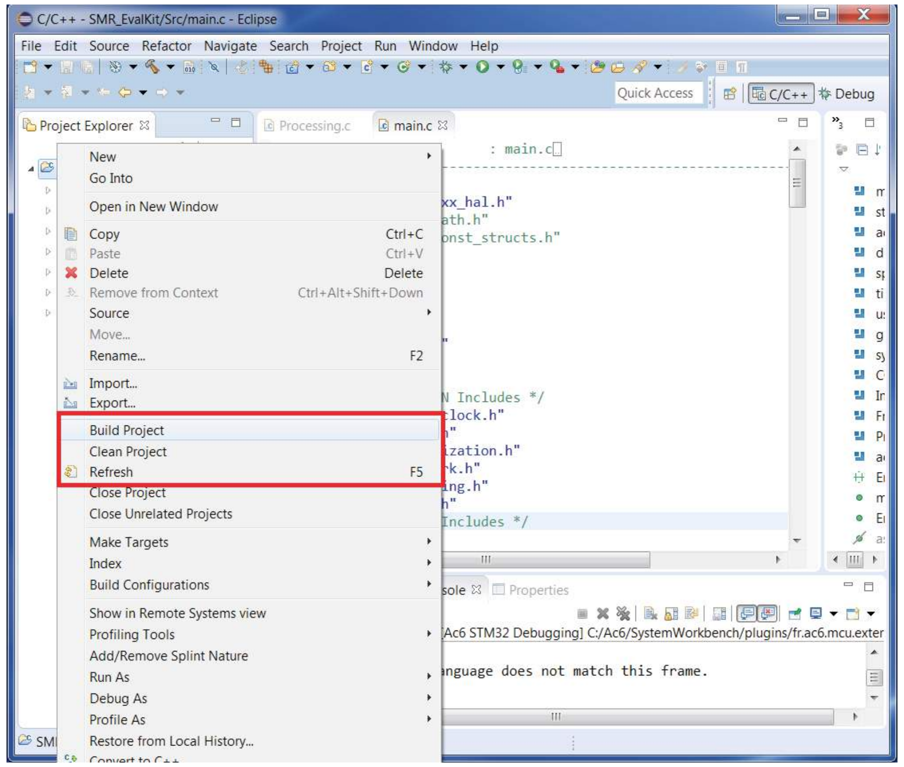
Writable   Smart Insert   29 : 31

c) The radar setting can be changed by modifying the parameters in this “CONFIG.h” file. For example changing radar configuration to FMCW mode by setting “#define MODULATION (1) ”

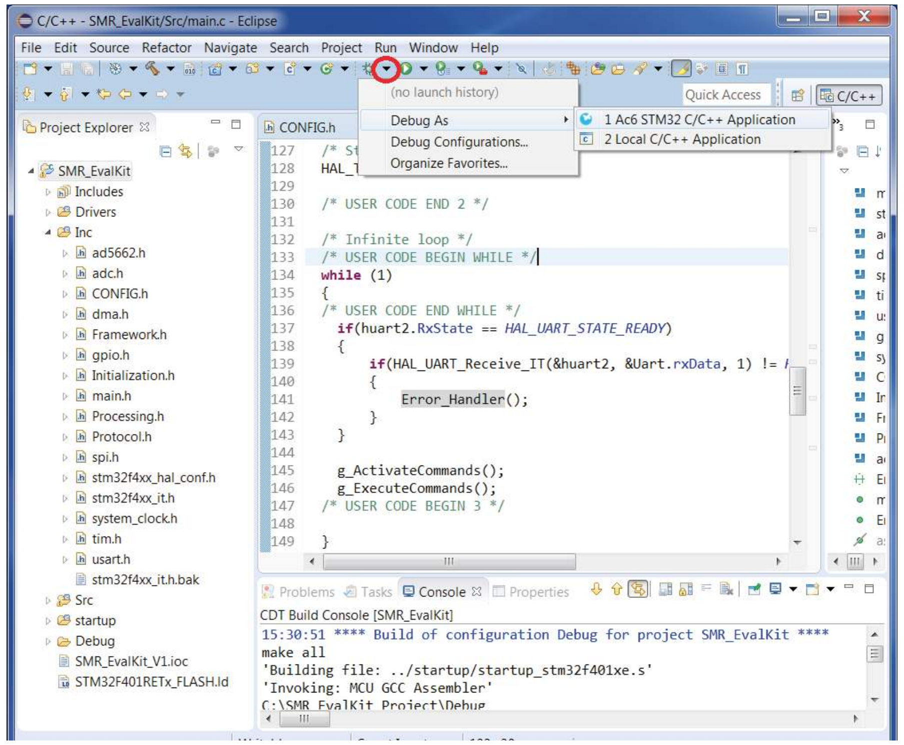


```
/* DEFINE the active modulation ---------------------------------------
#define MODULATION          (1)          /*  0: CW   */
                                         /*  1: FMCW */
/*--------------------------------------------------------------------

/* frequency settings ------------------------------------------------
/* !!!ATTENTION: the frequency should be between 24.000GHz and 24.250GHz becaus
#define FREQ_START          (24050)      /*  Start frequency in MHz  */
#define FREQ_STOP           (24200)      /*  Stop frequency in MHz   */

/*--------------------------------------------------------------------
/*********************************************************************

/*** MAY ONLY BE CHANGED IF YOU KNOW WHAT YOU ARE DOING ****************
/*--------------------------------------------------------------------

/* CW-modulation settings  ---------------------------------------------
/* sample time  */
#define SAMPLING_PERIOD     (400u)       /*  in [µs] */

/* cycle time   */
#define CYCLE_DURATION      (200000u)    /*  measurement cycle duration [µs] */
/*--------------------------------------------------------------------
```

d) Build the project after changing radar setting in “CONFIG.h” file. Note: It may be necessary to clean and refresh the project folder before building. Right click on project folder -> Clean Project. Right click on project folder -> Refresh. Right click on project folder -> Build Project.



e) Upload the modified source code onto microcontroller board. Right click on project folder (or left click on debug arrow button) -> Debug As -> Ac6 STM32 C/C++ Application

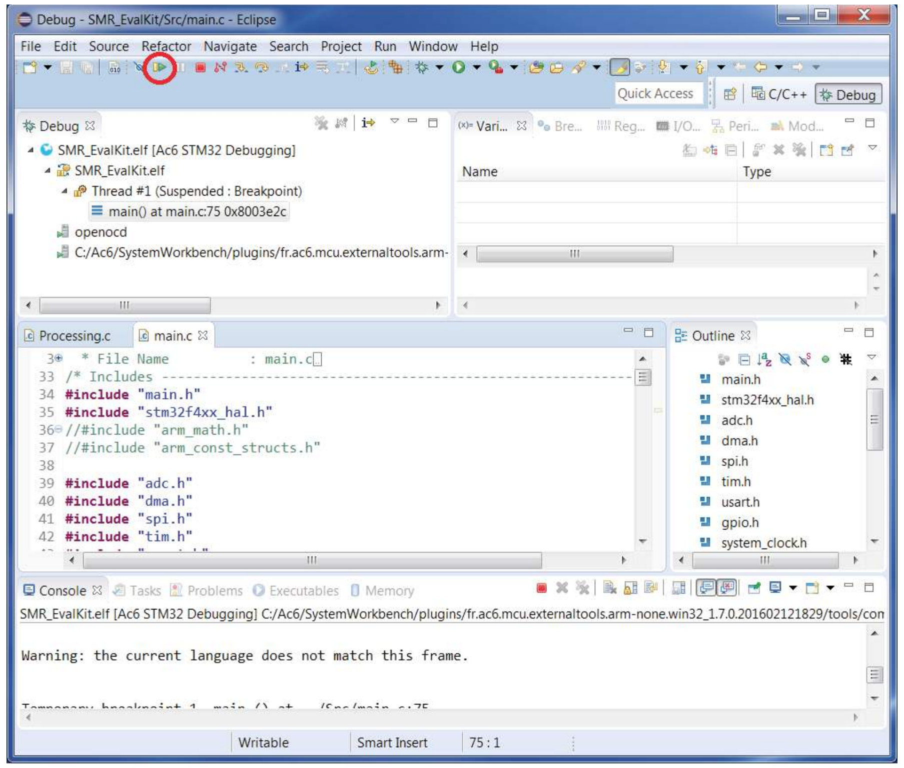


```
  /* USER CODE END 2 */

  /* Infinite loop */
  /* USER CODE BEGIN WHILE */
  while (1)
  {
  /* USER CODE END WHILE */
    if(huart2.RxState == HAL_UART_STATE_READY)
    {
        if(HAL_UART_Receive_IT(&huart2, &Uart.rxData, 1) != H
        {
            Error_Handler();
        }
    }

    g_ActivateCommands();
    g_ExecuteCommands();
  /* USER CODE BEGIN 3 */

  }
```

CDT Build Console [SMR_EvalKit]

```
15:30:51 **** Build of configuration Debug for project SMR_EvalKit ****
make all
'Building file: ../startup/startup_stm32f401xe.s'
'Invoking: MCU GCC Assembler'
C:\SMR_EvalKit_Project\Debug
```

f) To run the application. Click on “Run” button.



```
 * File Name          : main.c
/* Includes ------------------------------------------------------------------
#include "main.h"
#include "stm32f4xx_hal.h"
//#include "arm_math.h"
//#include "arm_const_structs.h"

#include "adc.h"
#include "dma.h"
#include "spi.h"
#include "tim.h"
```

SMR_EvalKit.elf [Ac6 STM32 Debugging] C:/Ac6/SystemWorkbench/plugins/fr.ac6.mcu.externaltools.arm-none.win32_1.7.0.201602121829/tools/con

```
Warning: the current language does not match this frame.
```

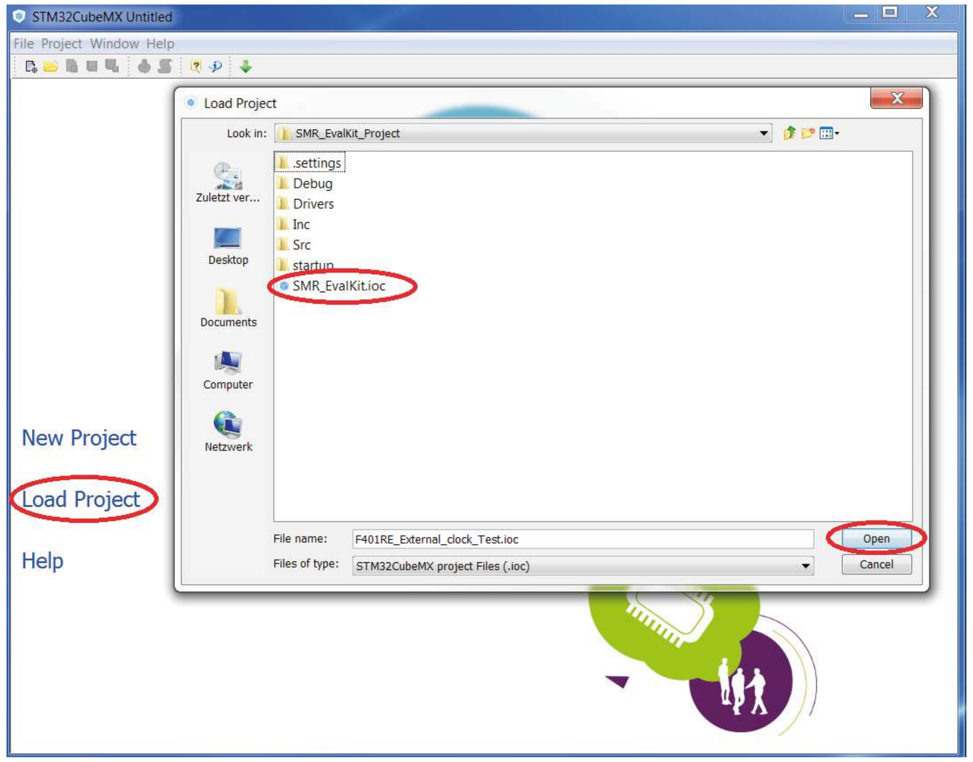
Debug - SMR_EvalKit/Src/main.c - Eclipse
File Edit Source Refactor Navigate Search Project Run Window Help
Quick Access
C/C++
Debug
SMR_EvalKit.elf [Ac6 STM32 Debugging]
SMR_EvalKit.elf
Thread #1 (Suspended : Breakpoint)
main() at main.c:75 0x8003e2c
openocd
C:/Ac6/SystemWorkbench/plugins/fr.ac6.mcu.externaltools.arm-
Vari...
Bre...
Reg...
I/O...
Peri...
Mod...
Name
Type
Processing.c
main.c
Outline
main.h
stm32f4xx_hal.h
adc.h
dma.h
spi.h
tim.h
usart.h
gpio.h
system_clock.h
Console
Tasks
Problems
Executables
Memory
Writable
Smart Insert
75 : 1

g) Once the firmware has even uploaded to the target and has been run it will be stored in the controllers flash memory and run even without connecting to the system workbench on power up.

SMR-EvalKit Pin layout and software components

a) STM32Cubemx

The SMR-EvalKit pin layout as well as software components configurations can be viewed using STM32Cubemx. To do this, open STM32Cubemx software and load the file “SMR_EvalKit.ioc” in the SMR_EvalKit_Project folder.



STM32CubeMX Untitled

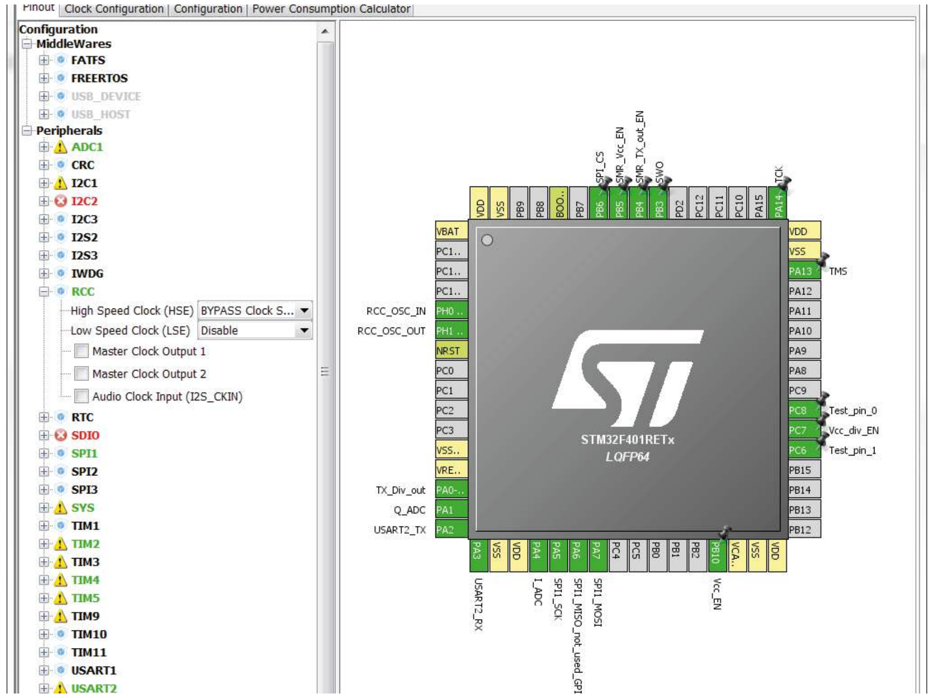
Load Project

Look in: SMR_EvalKit_Project

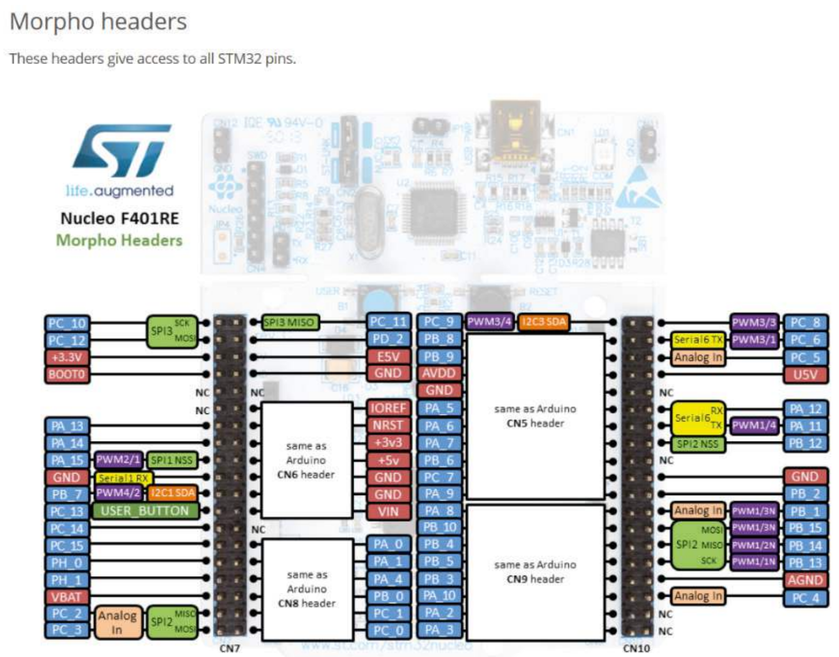
.settings
Debug
Drivers
Inc
Src
startup
SMR_EvalKit.ioc

File name: F401RE_External_clock_Test.ioc

Files of type: STM32CubeMX project Files (.ioc)



Configuration
MiddleWares
FATFS
FREERTOS
USB_DEVICE
USB_HOST
Peripherals
ADC1
CRC
I2C1
I2C2
I2C3
I2S2
I2S3
IWDG
RCC
High Speed Clock (HSE): BYPASS Clock S...
Low Speed Clock (LSE): Disable
□ Master Clock Output 1
□ Master Clock Output 2
□ Audio Clock Input (I2S_CKIN)
RTC
SDIO
SPI1
SPI2
SPI3
SYS
TIM1
TIM2
TIM3
TIM4
TIM5
TIM9
TIM10
TIM11
USART1
USART2



Morpho headers

These headers give access to all STM32 pins.

STM32F401RE Pin map (https://developer.mbed.org/platforms/ST-Nucleo-F401RE/)

b) SMR-EvalKit Pin Configuration

Name	Pin map	GPIO	Description	Note
Vcc_EN	PB10	Output	Power supply of SMR board	High-active
SMR_Vcc_EN	PB5	Output	Power supply SMR radar module	High-active
SMR_Tx_out_EN	PB4	Output	Transmit signal amplifier	High-active
Vcc_div_EN	PC7	Output	Transmit signal down-conversion	High-active
Tx_Div_out	PA0	Input	Down-converted transmit signal	Square wave signal
Test_pin_0	PC8	Output	For testing purpose	high-active
Test_pin_1	PC6	Output	For testing purpose	high-active
RCC_OSC_IN	PH0	n/a	External clock input	n/a
RCC_OSC_OUT	PH1	n/a	External clock output	n/a
I_ADC	PA4	Input	Receive I-Channel signal	Analog
Q_ADC	PA1	Input	Receive Q-Channel signal	Analog
SPI1_CS	PB6	Output	SPI Chip select	Low-active
SPI1_CLK	PA5	Output	SPI Communication clock	high-active
SPI1_MOSI	PA7	Output	SPI Transmission of data to SMR board	high-active
SPI1_MISO	PA6	Input	SPI Reception of data from SMR board	Unused
USART2_TX	PA2	Output	Transmission of receive signal to PC	High-active
USART2_RX	PA3	Input	Reception commands from PC	Unused
TCK	PA13	n/a	Debug serial wire clock	n/a
TMS	PA14	n/a	Debug serial wire I/O	n/a
SWO	PB3	n/a	Debug serial wire output trace port	n/a

SMR-EvalKit pin configuration with respect to STM32F401RE microcontroller board

Note: when toggling the following SMR I/O-pins: Vcc_EN, SMR_Vcc_EN, SMR_Tx_EN, Vcc_Div_EN, it is absolutely important to follow the scheme below with a 1ms delay between each pin.

Switching on: Vcc_EN -> SMR_Vcc_EN -> SMR_Tx_EN and Vcc_div_EN
Switching off: SMR_Tx_EN and Vcc_Div_EN -> SMR_Vcc_EN -> Vcc_EN

c) Software components brief description

Component	Usage
HSE	• High speed external clock for clocking MCU system
Timer_2	• Measure SMR down-converted transmit signal frequency
Timer_4	• Control ADC sampling period • Generate FMCW ramp • Monitor measurement cycle duration
Timer_5	• Monitor communication message timeout via USART
SPI	• Set SMR transmit frequency
ADC	• Sample analog receive signals of SMR module
DMA	• Read sampled signals from ADC registers • Write data to SPI registers
USART	• Communicate to PC
GPIO	• Interface between SMR board and MCU board

6. Radar configurable parameters

The “CONFIG.h” file in the source code provides a list of configurable parameters.

Parameter	Description	Note	Unit
MODULATION	selecting of radar principle	0: Doppler principle 1: FMCW principle	-
FREQ_START*)	Start frequency in FMCW mode	Min: 24000 Max: 24250	MHz
FREQ_STOP	Stop frequency in FMCW mode	Min: 24000 Max: 24250	MHz
FREQ_CALIB_EN	Frequency calibration enable	0: Off 1: On	
SAMPLE_PERIOD	ADC sampling interval	1*)	µs
CYCLE_DURATION	Measurement cycle duration	2*)	µs
FREQ_CALIB_INTVL	Frequency auto-calibration interval		Number of cycles
BAUDRATE	Data transmission rate	Default: 115000	Bits/s

*): FREQ_START must always be smaller than FREQ_STOP: FREQ_START < FREQ_STOP

1*): A careful consideration should be taken when changing the parameter “SAMPLE_PERIOD” since an improper value of this parameter may cause application software not functioning correctly.

2*): Min. CYCLE_DURATION = SAMPLE_PERIOD*FFT_SIZE + FRAME_LENGTH*10/BAUDRATE
E.g.: SAMPLE_PERIOD = 400µs, FFT_SIZE = 128, FRAME_LENGTH = 1031 bytes, BAUDRATE = 115000bits/s
-> Min. CYCLE_DURATION = 400µs*128 + 1031*10/115200 ≈ 141ms

7. Data Transmission

At the end of each measurement, the raw data as well as its FFT magnitudes in logarithmic scale will be transmitted to the GUI. The data is transmitted in a frame with the following structure:

Example: FFT_SIZE = 128 (i.e. 128 samples)

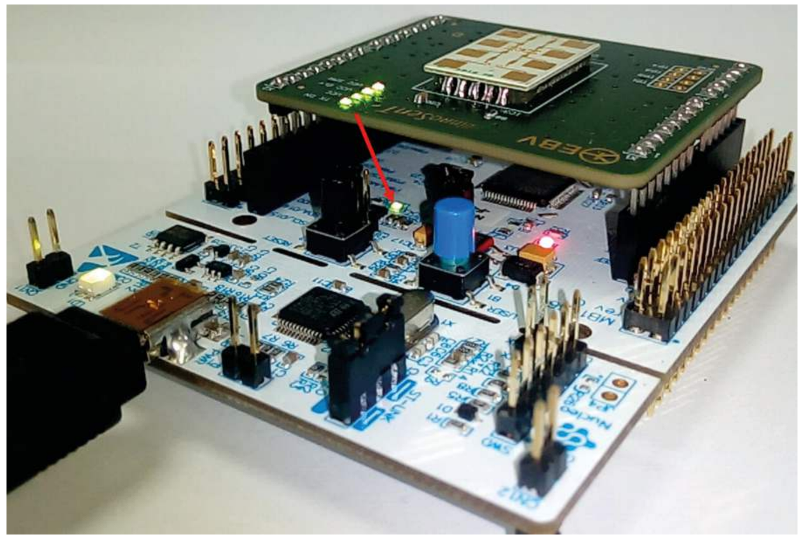
Byte No.	Byte 1	Byte 2	Byte 3	Byte 4,5	Byte 6,7		Byte 260,261 260,261	Byte 262,263		Byte 516,517	Byte 518,519 520,521		Byte 1026,1027 1028,1029	Byte 1030	Byte 1031
Data type	uint8	uint8	uint8	uint16	int16		int16	int16		int16	int32		int32	1 byte	1 byte
Content	SD	FC	Modulation	Number sample	Raw I1		Raw I128	Raw Q1		Raw Q128	FFT-mag 1		FFT-mag 128	CS	ED

SD: start delimiter = 162 (0xA1)	uint8
FC: function code = 224 (0xE0)	uint8
Modulation: 0-Doppler, 1-FMCW	uint8
Number sample: number of captured samples for single measurement	uint16
Raw I1: sample 1st of raw I signal	int16
Raw I128: sample 128th of raw I signal	int16
Raw Q1: sample 1st of raw Q signal	int16
Raw Q128: sample 128th of raw Q signal	int16
FFT-mag 1: magnitude of FFT sample 1st in logarithmic scale	int32
FFT-mag 128: magnitude of FFT sample 128th in logarithmic scale	int32
CS: checksum = (sum of Byte2 to Byte1029) & 0x000000FF	uint8
ED: end delimiter = 22 (0x16)	uint8

Note: Depending on FFT_SIZE the frame length will change and byte numbers may need to be adjusted, however the frame's structure remains.

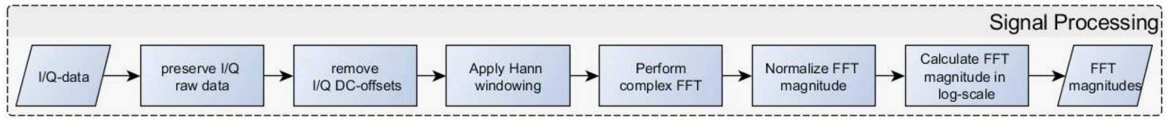
Transmission State

During the time when data is being transmitted to the PC, "LED2" on the microcontroller board will turn on and it will turn off again when the transmission is completed. During normal operation the LED will flash due to the continuous data transmission.



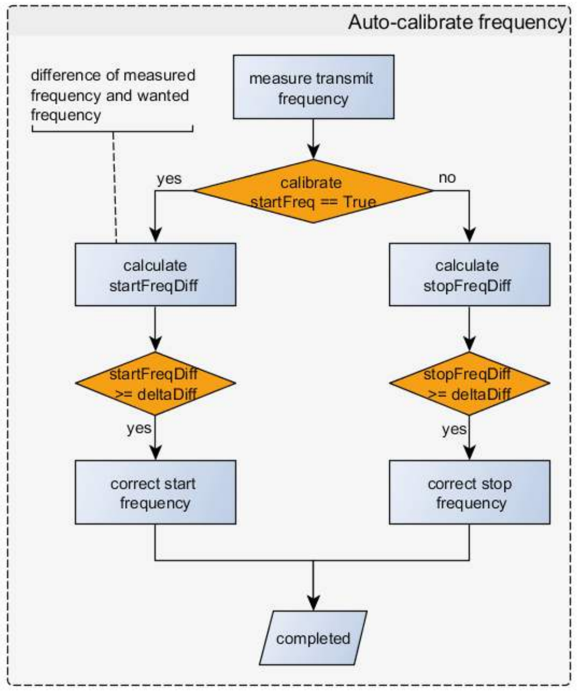
Signal Processing

This is an illustration of the signal processing for both Doppler and FMCW mode.



8. Frequency Auto-Calibration

One highlighted feature of the SMR-Evalkit is the ability to minimize transmit frequency drift over temperature. The following diagram describes the principle behind the frequency calibration.


Auto-calibrate frequency
difference of measured frequency and wanted frequency
measure transmit frequency
calibrate startFreq == True
yes
no
calculate startFreqDiff
calculate stopFreqDiff
startFreqDiff >= deltaDiff
stopFreqDiff >= deltaDiff
yes
yes
correct start frequency
correct stop frequency
completed

a) In the first step the transmit frequency of SMR module is measured. Afterwards the difference between the target transmit frequency and the measured transmit frequency is calculated. Based on this difference the transmit frequency is changed accordingly and tested again.

Transmit frequency measurement

The SMR-module outputs a square wave prescaler signal with a frequency corresponding to 1:8192 of its current transmit frequency. The prescaler is enabled by connecting "Vcc_div_EN" to 3.3V.

Example: A transmit frequency of 24.125GHz will result in a square wave signal with a frequency of approx. 2.944946MHz.

This square wave signal is routed into the microcontroller board via the "TX_Div_out" pin. In order to measure the frequency of the square wave signal, an input-capture channel of the microcontroller is applied on the "TX_Div_out" pin to detect and count the number of rising edges of the square wave signal in a duration of 100ms. After the sampling process, the frequency of the square wave signal is calculated by dividing the number of detected rising edges by 100ms.

Example: 293584 rising edges detected -> 2.93584MHz square wave signal

b) Frequency shift calculation

After measuring the frequency of the square wave signal, the difference between the target frequency and the measured frequency is calculated and used for correcting the transmit frequency.

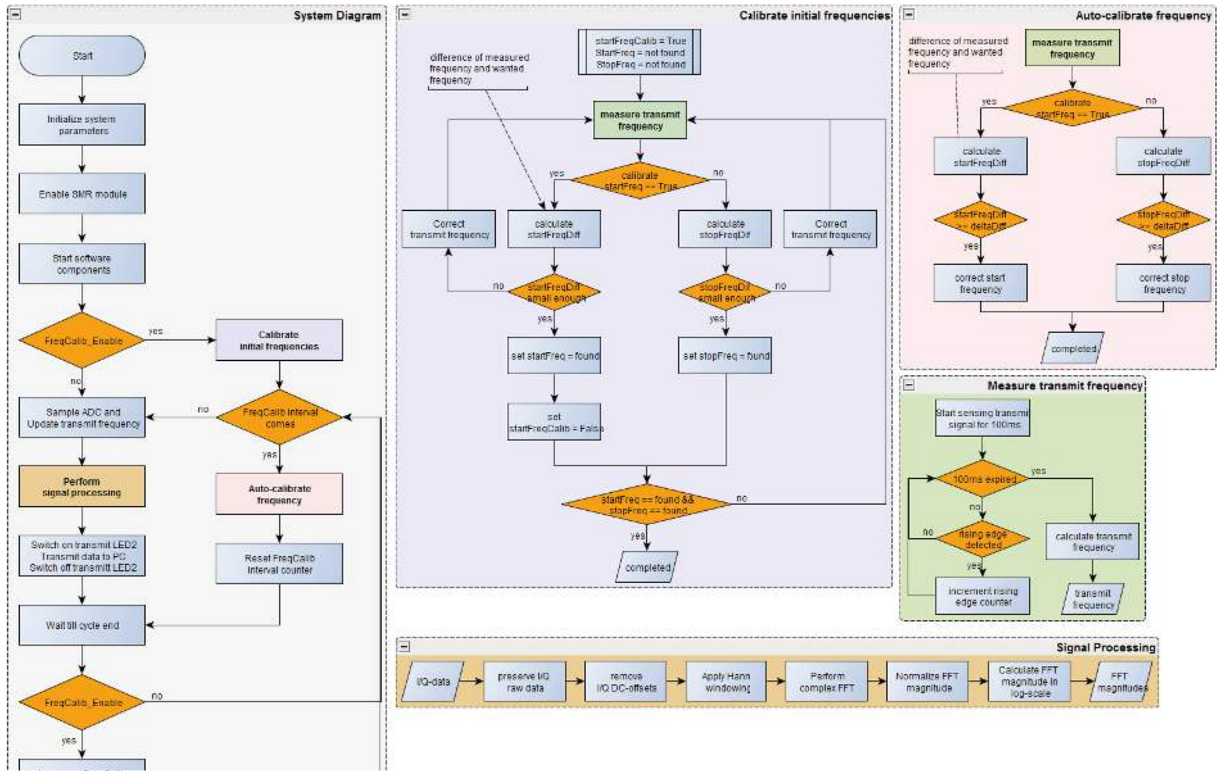
c) Transmit frequency correction

The calculated frequency difference is compared with a frequency tolerance. When the difference exceeds the tolerance, an adjustment for transmit frequency with a predefined value is performed to move the transmit frequency closer to desired frequency.

This frequency auto-calibration routine is repeated after a defined duration and this defined duration can be configured by the users. After calibration process, the application will wait for one measurement cycle before continuing with measurements.

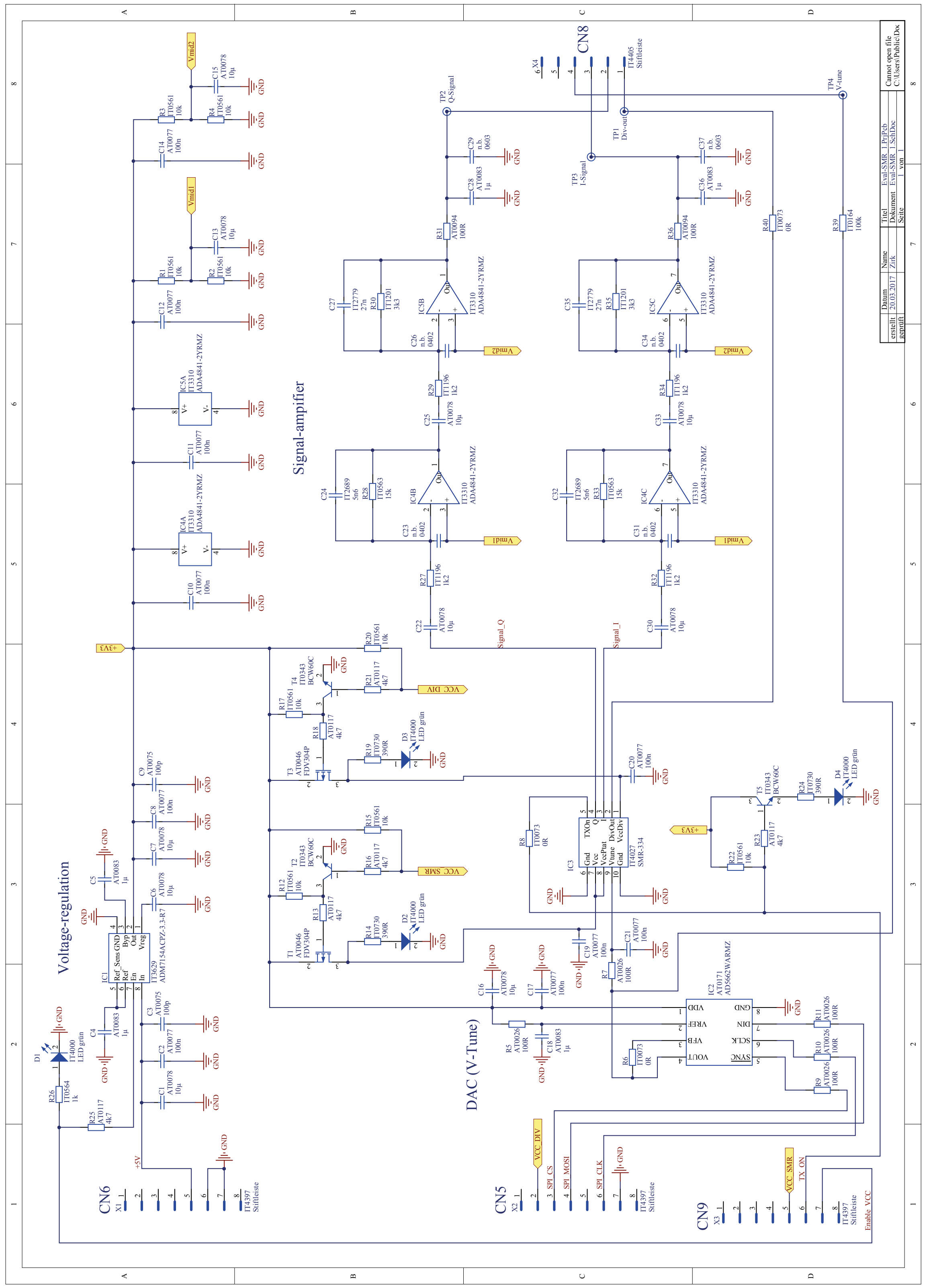
Note: In Doppler mode only start frequency is calibrated. In FMCW mode, start- and stop-frequency are calibrated alternately.

9. Complete Application Diagram


System Diagram
Start
Initialize system parameters
Enable SMR module
Start software components
FreqCalib_Enable
yes
Calibrate initial frequencies
no
Sample ADC and Update transmit frequency
no
FreqCalib interval comes
yes
Perform signal processing
Auto-calibrate frequency
Switch on transmit LED2 Transmit data to PC Switch off transmit LED2
Reset FreqCalib interval counter
Wait till cycle end
FreqCalib_Enable
no
yes
Calibrate initial frequencies
startFreqCalib = True StartFreq = not found StopFreq = not found
difference of measured frequency and wanted frequency
measure transmit frequency
yes
calibrate startFreq == True
no
Correct transmit frequency
calculate startFreqDiff
calculate stopFreqDiff
Correct transmit frequency
no
startFreqDiff small enough
stopFreqDiff small enough
no
yes
yes
set startFreq = found
set stopFreq = found
set startFreqCalib = False
startFreq == found && stopFreq == found
no
yes
completed
Auto-calibrate frequency
difference of measured frequency and wanted frequency
measure transmit frequency
yes
calibrate startFreq == True
no
calculate startFreqDiff
calculate stopFreqDiff
startFreqDiff >= deltaDiff
stopFreqDiff >= deltaDiff
yes
yes
correct start frequency
correct stop frequency
completed
Measure transmit frequency
Start sensing transmit signal for 100ms
100ms expired
yes
no
no
rising edge detected
calculate transmit frequency
yes
increment rising edge counter
transmit frequency
Signal Processing
I/Q-data
preserve I/Q raw data
remove I/Q DC-offsets
Apply Hann windowing
Perform complex FFT
Normalize FFT magnitude
Calculate FFT magnitude in log-scale
FFT magnitudes

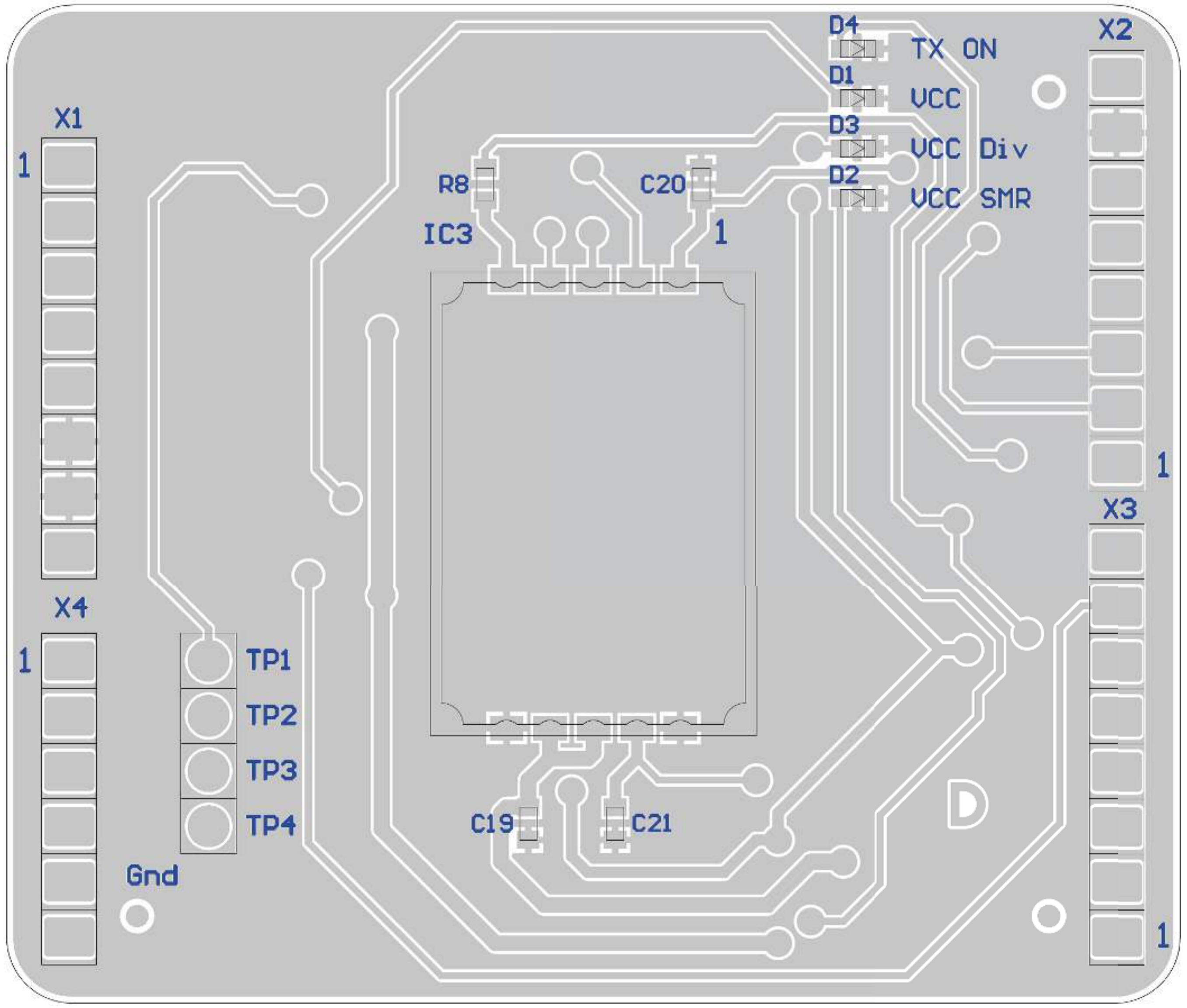
10. Hardware

Schematic circuit

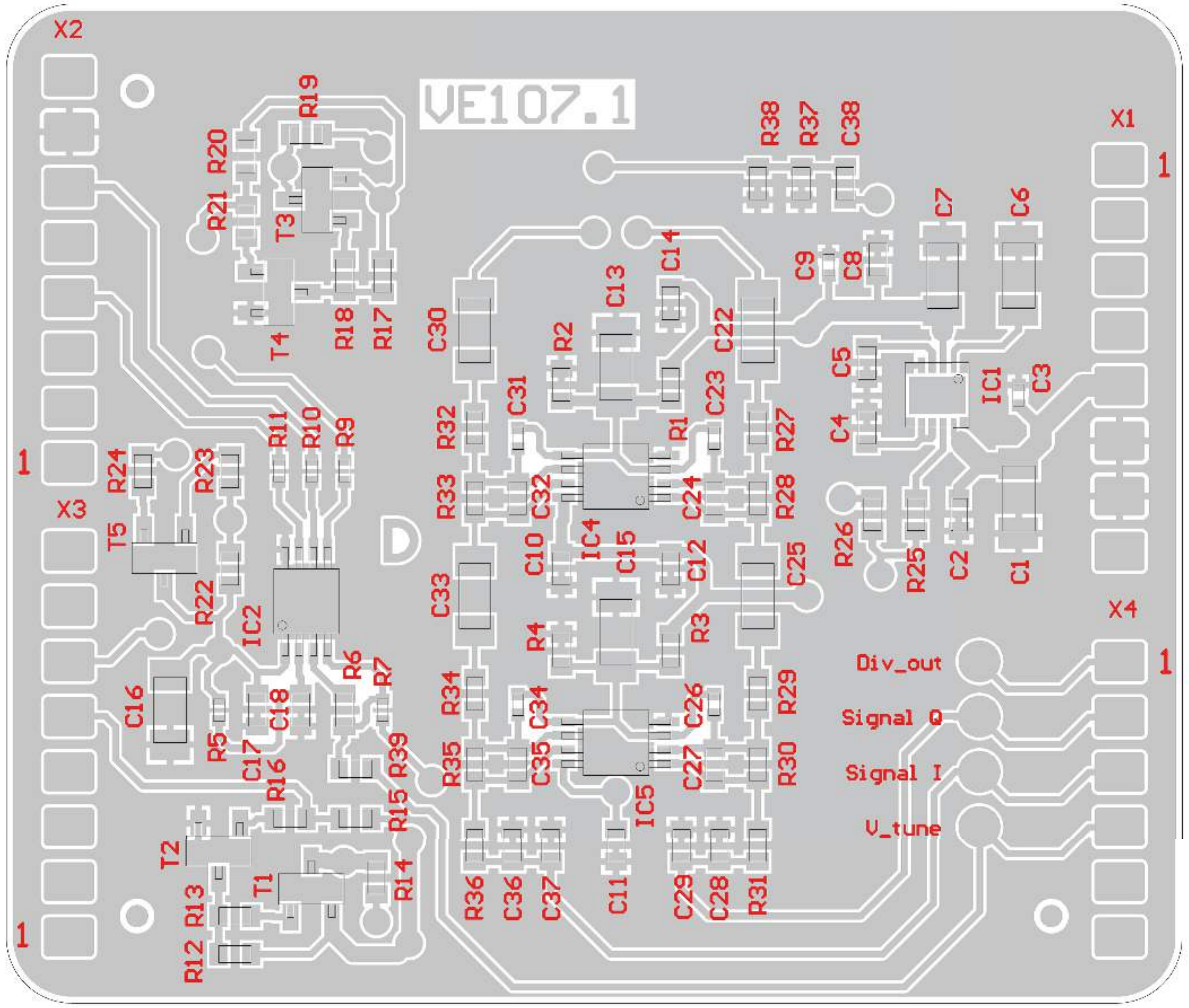


Layout

TOP



BOTTOM



SMR Technical characteristics

See Data Sheet SMR-334

11. References

STM32F401RE Nucleo reference manual:
http://www.st.com/content/ccc/resource/technical/document/reference_manual/5d/b1/ef/b2/a1/66/40/80/DM00096844.pdf/files/DM00096844.pdf/jcr:content/translations/en.DM00096844.pdf

STM32F401xE datasheet: http://www.st.com/content/ccc/resource/technical/document/datasheet/30/91/86/2d/db/94/4a/d6/DM00102166.pdf/files/DM00102166.pdf/jcr:content/translations/en.DM00102166.pdf

STM32 Nucleo-64 board user manual:
http://www.st.com/content/ccc/resource/technical/document/user_manual/98/2e/fa/4b/e0/82/43/b7/DM00105823.pdf/files/DM00105823.pdf/jcr:content/translations/en.DM00105823.pdf

12. History

Document revision	Date	Change log	Author
1	01.03.2017	first release	BL
1.1	20.03.2017	Added description "Transmission State"	SG
1.2	26.07.2017	Release	CD
1.3	22.05.2018	Corrected USART2_TX/RX pins	BL

InnoSenT GmbH

Am Roedertor 30
97499 Donnersdorf
Germany

Tel.: +49-9528-9518-0
Fax.: +49-9528-9518-99
E-mail: info@innosent.de
www.innosent.de