



Content

Introduction	3
Installation	4
Brief Introduction to Radar Principles	6
Getting Started	11
Radar configurable parameters	19
Data Transmission	20
Signal Processing	22
Frequency Auto-Calibration	22
Complete Application Diagram	24
Hardware	25
References	28
History	28

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 •
- •
- 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 •
- Kit

SMR Evaluation Kit user manual

Frequency auto-calibration for compliance with government regulations

Source code with detailed comments giving users full control to the Evaluation

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)



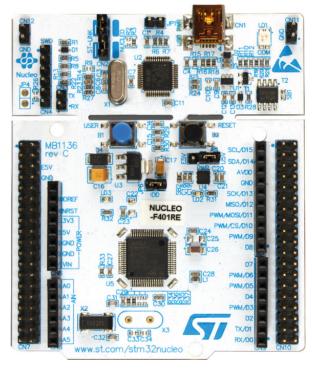
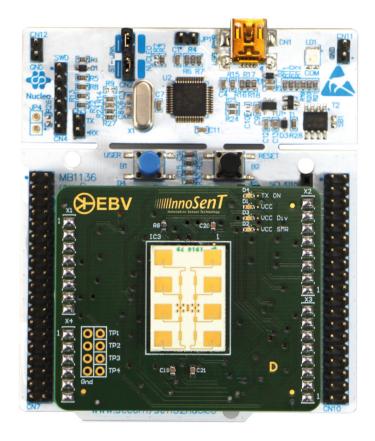


Figure 1: SMR Antenna module

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.
- 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 be checked again before applying the power supply.
- Jumpers on the microcontroller board should be left unchanged.



Software Installation

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

• When the two boards are separated and need to be assembled, connect them as prevent any damage to the system due to wrong connection and the connection should

Figure 3: SMR-board and STM32 board connection

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:

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

- Doppler- or differential frequency f_D
- Transmit frequency of the radar f_o
- V velocity of the moving object

C_o 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:

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

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 900, 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	CO	299792458	m/s
CW-Frequency	F	depending on FREQ_START	Hz
		Example: 24.000	GHz
sampling period	Та	0.0004	S
		400	μs
buffer length (number of samples)	Ν	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

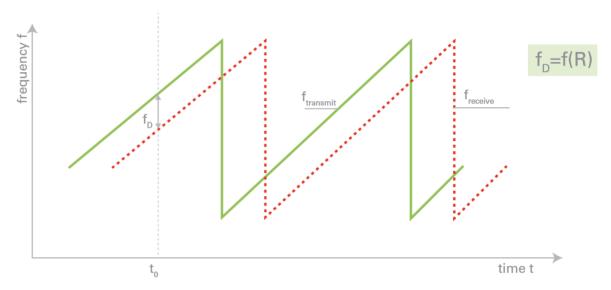
6

$$\mathbf{Tsample} = \mathrm{Ta} * \mathrm{N} = 400\,\mu\mathrm{s} * 128 = \mathbf{51.}\,\mathbf{2ms}$$
$$\mathbf{Vmin} = \frac{C_0}{2 * \mathrm{Tsample} * \mathrm{F}} = \frac{299792458\mathrm{m/s}}{2 * 51.2\mathrm{ms} * 24.000\mathrm{GHz}} = \mathbf{0.122ms}$$
$$\mathbf{Vmax} = \mathrm{Vmin} * \left(\frac{\mathrm{N}}{2} - 1\right) = 0.122ms * \left(\frac{128}{2} - 1\right) = \mathbf{7.68ms}$$
$$\mathbf{LP} = \frac{1}{2 * Ta} = \frac{1}{2 * 400\mu\mathrm{s}} = \mathbf{1250Hz}$$
$$\mathbf{HP} = \frac{1}{Tsample} = \frac{1}{51.2} = \mathbf{19.53Hz}$$

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:

$$R = \frac{C_0}{2} * T *$$

- f_D ∆f differential frequency frequency deviation (bandwidth)
- Т 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 Inno-SenT 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	CO	299792458	m/s
bandwidth	В	depending on FREQ_ START and FREQ_STOP	Hz
		Example: 250	MHz
sampling period	Та	0.0004	s
		400	μs
puffer length (number of samples)	Ν	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

 $FREQ_STOP = 24.250GHz$

$\mathbf{B} = FREQ_{STOP} - FREQ_{START} = 24.000GHz - 24.250GHz = 250MHz$

Tsample =
$$Ta * N = 400 \mu s * 128 = 51.2 ms$$

$$\mathbf{Rmin} = \frac{C_0}{2 * B} = \frac{299792458 \text{m/s}}{2 * 250 \text{MHz}} = \mathbf{0.60m}$$
$$\mathbf{Rmax} = \mathbf{Rmin} * \left(\frac{N}{2} - 1\right) = 0.60m * \left(\frac{128}{2} - 1\right) = \mathbf{37.8m}$$
$$\mathbf{LP} = \frac{1}{2 * ta} = \frac{1}{2 * 400 \mu s} = \mathbf{1250Hz}$$
$$\mathbf{HP} = \frac{1}{\text{Tsample}} = \frac{1}{51.2} = \mathbf{19.53Hz}$$

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

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.

• Click Connect/Disconnect button to connect or disconnect to SMR EvalKit Comport

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:

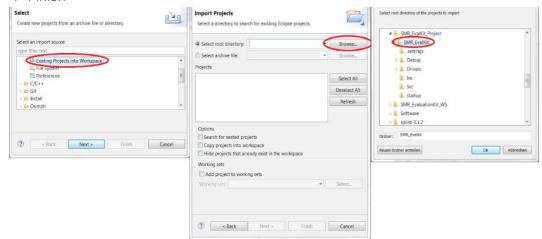
- Create a workspace folder for the project e.g: "C:\SMR_EvaluationKit_WS" a)
- b) Copy the SMR EvalKit firmware project folder into the workspace
- C) Run SW4STM32 software and select the created workspace folder

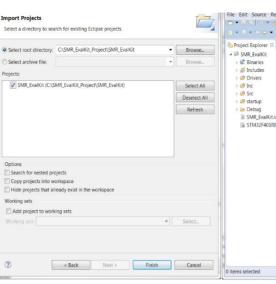
G Workspace Launcher	X
Select a workspace	
Eclipse stores your projects in a folder called a workspace Choose a workspace folder to use for this session.	e.
Workspace: C\SMR_EvaluationKit_WS	Browse
Use this as the default and do not ask again	
	OK Cancel

d) Go to workbench

C/C++ - Eclipse	C/C++ - Eclipse	There is a provide the second second	
File Edit Source Refactor Navigate Search Project Run Window Help		wigate Search Project Run Window Help	
		8 • 5 • 6 • 12 • 0 • 6 • 6 • 1× 6 6 8	
	$[5] = [0] = [0] \oplus [0] \oplus [0] \oplus [0] \oplus [0]$		Quick Access
Workbench	Project Explorer 😂 👘 🗖		<u>*0</u> × *
	E 🤤 🐏 🍯		
Welcome to the Eclipse IDE for C/C++			An outline is not available.
Developers			
Overview Tutorials			
Get an overview of the features Go through tutorials			
Samples A What's New			
Try out the samples V Find out what is new			
			9 7 7 8
		Problems Tasks Console Properties I items	P
 Consider a second state of the se			Resource Path
		я (п	
	0 items selected		

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

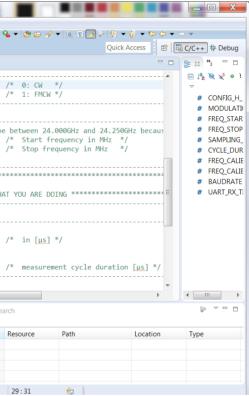




Expand the project and open the file "CONFIG.h" in "Inc" folder b)

1			
ſ	C/C++ - SMR_EvalKit/Inc/C	ONFIG.h - Eclipse	
	File Edit Source Refactor	Navigate Search Project Run Windo	w Help
	📑 🕶 🔒 🕼 📎 🕶 🚳 🖛 🗟	× & the test test test test test test test	• O • 9 • 0
Ľ			
Ľ	Project Exp 🛛 🗖 🗖	CONFIG.h 🛛	
Ľ		28 /* DEFINE the active modula	
l	a 😂 SMR_EvalKit 🔺	29 #define MODULATION	(0)
	Binaries	31 /*	
H	Includes	32	
Ľ	Drivers	33⊖/* frequency settings	
	4 😕 Inc	34 /* !!!ATTENTION: the freque	
Ľ	▷ h ad5662.h ▷ adc.h.	35 #define FREQ_START 36 #define FREQ_STOP	(24050) (24200)
	CONFIG.h	30 #define FREQ_STOP	(24200)
l	b dma.h	380 /*	
l	Framework.h	39 /************************************	*******
	p in Frameworkh p in gpio.h	40	
l	Initialization.	41⊕ /*** MAY ONLY BE CHANGED IF	YOU KNOW WHA
l	Main.h	42 /*	
	Processing.h	44⊖ /* CW-modulation settings	
Π	Protocol.h	45 /* sample time */	
L	In spi.h	46 #define SAMPLING_PERIOD	(400u)
Ľ	▷ In stm32f4xx ha	47	
Π	▷ In stm32f4xx_it.h	48 /* cycle time */ 49 #define CYCLE DURATION	(200000)
	system_clock.	49 #define CYCLE_DURATION 50 /*	(2000000)
Π	⊳ là tim.h	51	
l	b is usart.h	•	
Π	CONFIG.h.bak	🖹 Problems 🛿 🧟 Tasks 🖳 Console 🔲	Droportion @ Con
	Initialization.		Properties Ar Sea
Π	stm32f4xx_it.	0 items	
	▷ 😂 Src	Description	
	Image: Startup		
	🛛 🗁 Debug		
Ľ	SMR_EvalKit.ioc •		
	< <u> </u>		
ſ		Writable	Smart Insert

	Project Run Window Help							
* ≥ ≥ ≤ ≤ ≤								
÷ •	Quick Access 🔡 🛱	C/C++						
K txflashid		O 22 ^a s ^a □ ³ P ^a An outline is not available.						
	🖹 Problems 🗐 Tasks 🔍 Console 🛙 🗖 Properties 📑	0 • 🗂 • ° 8						
	No consoles to display at this time.							



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

) C/C++ - SMR_EvalKit/Inc/C	ONFIG.h - Eclipse Navigate Search Project	Rup Window Help				
	i 🔌 🖾 🏪 🔐 ▼ 🚳 ▼ 🗟		- 💁 - 🍅 🖂 🖌	/ - 🗉 🖬 🍠 💝	9 - A	⇒ -
					Quick Access	🖻 C/C++ 🕸 Debu
Project Exp 🛛 🗖 🗖	B CONFIG.h ∺				- 0	<u>₽</u> ≈ »₃ =
⊟ 🔄 🐌 🗵	28 /* DEFINE the act				· · · · · · · · · · · · · · · · · · ·	🖃 🗦 🖁 🙀 🐒 🖉
🛯 😂 SMR_EvalKit	29 #define MODULATIO	N (1)	/* 0: CW /* 1: FMCV			~
Binaries	30		/~ 1: FMCI	N ~7		# CONFIG
Includes	32					# MODUL
Drivers	330 /* frequency sett	ings				# FREQ_ST
🔺 🐸 Inc	34 /* !!!ATTENTION:	the frequency shoul	d be between 24	4.000GHz and 2	4.250GHz becaus	# FREQ_ST
ad5662.h	35 #define FREQ_STAR					# SAMPLI
Adc.h	36 #define FREQ_STOP	(24200)	/* Stop fr	requency in MH	z */	# CYCLE_D
CONFIG.h	37 38⊕ /*					# FREQ_C/
🖻 庙 dma.h	38 /*					# FREQ_C/
Framework.h	40					# BAUDRA
b 🗟 gpio.h	410 /*** MAY ONLY BE	CHANGED TE YOU KNOW	WHAT YOU ARE D	OOTNG *******	**********	# UART_R
Initialization.	42 /*					
Imain.h	43					
Processing.h	44⊖ /* CW-modulation					
Protocol.h	45 /* sample time *					
b 🗟 spi.h	46 #define SAMPLING_ 47	PERIOD (400u)	/* in [<u>µs</u>]] */		
Image: Stm32f4xx_ha	47 48 /* cvcle time *	/				
Istm32f4xx_it.k	49 #define CYCLE DUR) /* measure	ement cycle du	ration [us] */	
Isystem_clock.	50 /*				(<u>[]]</u> /	
b lim.h	51				-	
b la usart.h	4				4	•
CONFIG.h.bak	🖹 Problems 🛛 🧔 Tasks 📃	Consolo 🔲 Droportios 👌	9 Caarab			5 V P
Initialization.		Console im Properties A	Search			
stm32f4xx_it.ł	0 items	~				
Src	Description		Resource	Path	Location	Туре
Image: Startup						
Debug						
SMR_EvalKit.ioc -						
· · · · · · · · · · · · · · · · · · ·						

Build the project after changing radar setting in "CONFIG.h" file. Note: It may be d) 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.

⊖ c/c	++ -	- SMR_EvalKit/Src/main.c - Eclipse		_ C	X
File E	Edit	Source Refactor Navigate Search Project Run Wind	low Help		
- 12		🔝 🗞 🕶 🚳 💌 🕹 🐂 🔂 🕶 🖓 🕶 🕑 🕶	‡> • • • • • • • • • • • • • • • • • • •		
12 -		$\mathbf{v} \div \mathbf{\phi} \mathbf{v} \Rightarrow \mathbf{v}$	Quick Access	C++ *	🌾 Debug
🕒 Pro	ject	Explorer 🛛 🗖 🗖 Processing.c 📝 main.c	2	· 🗆	»₃ □
		New	: main.c.	*	9 E I
▲ 😂		Go Into		=	~
- P		Open in New Window	xx_hal.h"		≌ m ∎ st
Þ	D	Copy Ctrl+C	ath.h" onst structs.h"		∎ a
Þ	ß	Paste Ctrl+V	blist_structs.n		😐 d
₽	×	Delete Delete			🖬 st
₽	<u>.</u>	Remove from Context Ctrl+Alt+Shift+Down			🖳 ti
₽		Source +			🖬 u:
		Move	•		∎ g
		Rename F2			s)
		Import	N Includes */		u lo
		Export	lock.h"		S Fr
		Build Project	1"		🖬 Pi
		Clean Project	ization.h" 'k.h"		🖬 ai
	8	Refresh F5	ing.h"		÷ Ei
		Close Project Close Unrelated Projects	h"		• m
			Includes */		• Ei
		Make Targets	4 111	*	 ✓ □ ✓ □
		Index Build Configurations			
			sole 🛙 🔲 Properties		
		Show in Remote Systems view			
			Ac6 STM32 Debugging] C:/Ac6/SystemWorkbench/pluging	s/fr.ace	.mcu.exter
		Add/Remove Splint Nature Run As	inguage does not match this frame.		-
		Debug As			
		Profile As	III		- F
🐸 SMI		Restore from Local History	:		
	C.	Convert to C++	1		

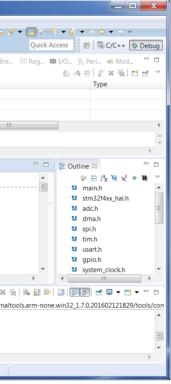
Upload the modified source code onto microcontroller board. Right click on e) project folder (or left click on debug arrow button) -> Debug As -> Ac6 STM32 C/C++ Application

C/C++ - SMR_EvalKit/Src/main.c - Ecli	pse						
File Edit Source Refactor Navigate	Search	n Projec	t Run Windo	w Help			
📑 🛨 🔛 🐚 🛞 🕶 🗞 🕶 🗟 🛨 🚳	▼ C ²	• 🞯 •	\sim				
∮ + ∅ + ♥ Φ + ⇒ +			(no launch	history)			
Project Explorer 🛛 🗖 🗖		IFIG.h	Debug As				
🗏 😫 🔮 ▽	127	/* St	-	nfigurations			
🔺 🚰 SMR_EvalKit	128 129	HAL_	Organize I	-avorites			
 Includes Brivers 	130	/* US	SER CODE END	2 */			
A 🖓 Inc	131						
⊳ 🖻 ad5662.h	132 133		finite loop SER CODE BEG				
⊳ 🖻 adc.h	134	while					
▷ 🖻 CONFIG.h ▷ 🖻 dma.h	135 136	{ /* II	SER CODE END	WHTLE */			
Framework.h	137			ate == HAL_UAR1			
⊳ 🖻 gpio.h	138 139	{		T Receive IT(&			
Initialization.h	140		{	I_RECEIVE_II(di			
Main.h	141			Handler();			
 Processing.h Protocol.h 	142 143	}	}				
⊳ 🖻 spi.h	144						
Istm32f4xx_hal_conf.h	145 146		ActivateComm ExecuteComma				
▶ 🗎 stm32f4xx_it.h	147		SER CODE BEG				
▷ là system_clock.h ▷ là tim.h	148	1					
▷ 🖻 usart.h	149	}		1			
stm32f4xx_it.h.bak							
⊳ 😂 Src				ole 🛛 🔲 Properties			
Balance			<pre>le [SMR_EvalKit * Build of c</pre>	onfiguration D			
Debug SMR_EvalKit_V1.ioc	make a	all					
STM32F401RETx_FLASH.Id			le:/start CU GCC Assem	up/startup_stm			
			it Proiect\[
	•						
0 T				" D			
f) To run the a	ppli	catio	on. Click	on "Rur			
Debug - SMR_EvalKit/Src/main.c - Eclip							
(Projec	t Run Window	v Help			
File Edit Source Refactor Navigate	Search						
(Search						
File Edit Source Refactor Navigate	Search	東武	S = 4 ★	0 - 9 - 12 0			
File Edit Source Refactor Navigate	Search ତ: i≓	東武	S = 4 + S	0 - 9 - 12 0			
File Edit Source Refactor Navigate	Search ତ: i≓	東武	S = 4 ★	0 - 9 - 12 0			
File Edit Source Refactor Navigate	Search I≫: i≓ gging]	東武	S = 4 ★	O ▼ 9 ▼ 2 0 00			
File Edit Source Refactor Navigate	Search I≫ i≓ gging] ⊌kpoint)	東武	S = 4 ★	O ▼ 9 ▼ 2 0 00			
File Edit Source Refactor Navigate	Search gging] skpoint) 3e2c	司 武 後 #	& 1 ₩ ★ ▼	O ▼ 9 ▼ 2 0 00			
File Edit Source Refactor Navigate	Search gging] skpoint) 3e2c	司 武 後 #	& 1 ₩ ★ ▼	O ▼ 9 ▼ 2 0 00			
File Edit Source Refactor Navigate	Search gging] skpoint) 3e2c	司 武 後 #	& 1 ₩ ★ ▼	O ▼ 9 ▼ 2 0 00			
File Edit Source Refactor Navigate Image: Constraint of the second se	Search gging] skpoint) 3e2c	司 武 後 #	& 1 ₩ ★ ▼	O ▼ 9 ▼ 2 0 00			
File Edit Source Refactor Navigate File Edit Source Refactor Navi	Search gging] skpoint) 3e2c ins/fr.ac	- = * # # 6.mcu.ext	& 1 ₩ ★ ▼	O ▼ 9 ▼ 2 0 00			
File Edit Source Refactor Navigate Image: Source Refactor Navigate<	Search gging] skpoint) 3e2c	- = * # # 6.mcu.ext	& 1 ₩ ★ ▼	O ▼ 9 ▼ 2 0 00			
File Edit Source Refactor Navigate File Rome Source File Rome Rome Source File Rome Rome Rome Rome	Search gging] akpoint) 3e2c ins/fr.ac	- = * # # 6.mcu.ext	& 1 ₩ ★ ▼	O ▼ 9 ▼ 2 0 00			
File Edit Source Refactor Navigate Image: Source Refactor Navigate<	Search gging] akpoint) 3e2c ins/fr.ac	- = * # # 6.mcu.ext	& 1 ₩ ★ ▼	O ▼ 9 ▼ 2 0 00			
File Edit Source Refactor Navigate File Rome Source File Rome Rome Source File Rome Rome Rome Rome	Search gging] skpoint) 3e2c ins/fr.ac main.	 ₹	& 1 ₩ ★ ▼	O ▼ 9 ▼ 2 0 00			
File Edit Source Refactor Navigate File Source Source Refactor Navigate File Source Sour	Search gging] skpoint) 3e2c ins/fr.ac main.	 ₹	& 1 ₩ ★ ▼	O ▼ 9 ▼ 2 0 00			
File Edit Source Refactor Navigate File Edit Source Refactor Navigate File Edit Source Refactor Navigate SMR_EvalKitelf [Ac6 STM32 Debu SMR_EvalKitelf A P Thread #1 (Suspended : Bread main() at main.cr35 0x800 J openocd C/Ac6/SystemWorkbench/plug C/Ac6/SystemWorkbench/plug File Name 33 /# Tholudes 34 #include "main.h" 35 #include "main.h" 37 //#include "arm_const_st 38 #include "adc.h"	Search gging] skpoint) 3e2c ins/fr.ac main.	 ₹	& 1 ₩ ★ ▼	O ▼ 9 ▼ 2 0 00			
File Edit Source Refactor Navigate File Edit Source Refactor Navigate SMR_EvalKitelf [Ac6 STM32 Debu	Search gging] skpoint) 3e2c ins/fr.ac main.	 ₹	& 1 ₩ ★ ▼	O ▼ 9 ▼ 2 0 00			
File Edit Source Refactor Navigate File SMR_EvalKitelf [Ac6 STM32 Debu	Search gging] kpoint) 3e2c ins/fr.ac main. h" ructs.	> ₹ % # 6.mcu.ext	& 1 ₩ ★ ▼	O ▼ 9 ▼ 2 0 00			
File Edit Source Refactor Navigate File Edit Source Refactor Navigate SMR_EvalKitelf [Ac6 STM32 Debu	Search gging] kpoint) 3e2c ins/fr.ac main. h" ructs.	 ₹	& 1 ₩ ★ ▼	• • • • • • • • • • • • • • • • • • •			
File Edit Source Refactor Navigate File Edit Source Refactor Navigate SMR_EvalKitelf [Ac6 STM32 Debu SMR_EvalKitelf SMR_EvalKitelf SMR_EvalKitelf SMR_EvalKitelf Smain() at main.c75 0x800 C/Ac6/SystemWorkbench/plug File Name C/Ac6/SystemWorkbench/plug File Name SMR SMR SMR SMR SMR SMR SMR SM	Search gging] kpoint) 3e2c main. h" ructs.	• 司 武 後 絆 6.mcu.ext h* m ables ()	Semaltools.arm-	• • • • • • • • • • • • • • • • • • •			
File Edit Source Refactor Navigate File Edit Source Refactor Navigate Smc EvalKitelf [Ac6 STM32 Debu B SMR EvalKitelf B SMR EvalKitelf B Smc EvalKitelf B main() at main.c75 0x800 B openocd C /Ac6/SystemWorkbench/plug File Name B Tread #1 Include "main.h" B #include "main.h" B #include "arm_const_st B #include "adc.h" File Name B #include "adc.h" B #include "tim.h" File Name B #include #include#include#include#include#include#include#include#include#include	Search gging] kpoint) 3e2c main. h" ructs.	• 司 武 後 絆 6.mcu.ext h* m ables ()	Semaltools.arm-	• • • • • • • • • • • • • • • • • • •			
File Edit Source Refactor Navigate File Edit Source Refactor Navigate SMR_EvalKitelf [Ac6 STM32 Debu SMR_EvalKitelf SMR_EvalKitelf SMR_EvalKitelf SMR_EvalKitelf Smain() at main.c75 0x800 C/Ac6/SystemWorkbench/plug File Name C/Ac6/SystemWorkbench/plug File Name SMR SMR SMR SMR SMR SMR SMR SM	Search gging] kkpoint) kkpoint) search kcolor main. h" ructs.	R R R R R R R R R R R R R R R R R R R	Si % ▼ ■ □ i+ ▼ ■ □ iernaltools.arm- i i Memory orkbench/plugin	Vari 23 • Br Name			
File Edit Source Refactor Navigate File Edit Source Refactor Navi	Search gging] gging] kkpoint) 3e2c ins/fr.ac main. h" ructs. - - - - - - - - - - - - -	الله	Si % ▼ ■ □ i+ ▼ ■ □ iernaltools.arm- i i Memory orkbench/plugin	Vari 23 • Br Name			
File Edit Source Refactor Navigate File Edit Source Refactor Navi	Search gging] gging] kkpoint) 3e2c ins/fr.ac main. h" ructs. - - - - - - - - - - - - -	R R R R R R R R R R R R R R R R R R R	Si % ♥ ♥ i* ♥ ■ □ iernaltools.arm- iernaltools.arm- ion this frame	Vari 23 • Br Name			
File Edit Source Refactor Navigate File Edit Source Refactor Navi	Search gging] kkpoint) kkpoint) kkpoint) search search main. h" ructs. Search C/Ac6/ does r	R R R R R R R R R R R R R R R R R R R	Si % ♥ ♥ ♥ i ♥ ♥ ■ □ i P ♥ ■ □ i				

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

	X
🐂 😕 🕞 🖋 🔻 📝 🐳 🗉 🔳	
Quick Access 😰 😼	C/C++
● 1 Ac6 STM32 C/C++ Application	2, 0
2 Local C/C++ Application	» El
	~ • •
_	u m
	🖬 st
	🖬 ai
	u u
	st
	ti
_STATE_READY)	ч и: ч а
-	■ g ■ s\
nuart2, &Uart.rxData, 1) != /	
Ē _	🖬 In
	🔛 Fr
	🖬 Pi
	a a
	÷ Ei
	● m ● Ei
_	ø ∎ ø a:
	(III)
; ↓ ☆ 🚯 💷 🔉 = 🔍 🗹 🛡 ▾ 📬 ▾	
ebug for project SMR_EvalKit ****	
	=
32f401xe.s'	
	-
	P.

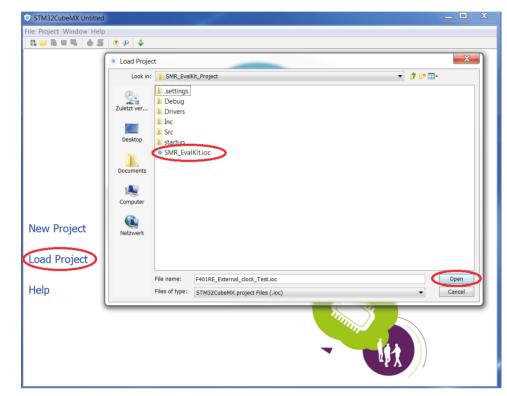
n" button.

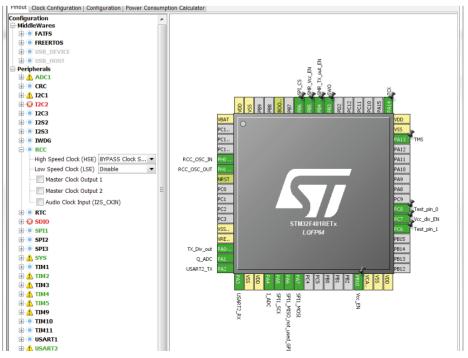


SMR-EvalKit Pin layout and software components

STM32Cubemx a)

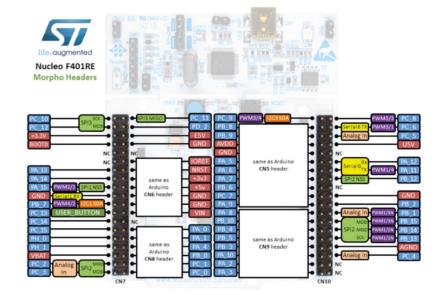
The SMR-EvalKit pin layout as well as software components configurations can be viewed using STM32Cubemx. To do this, open STM32Cubemx software and load the file "SMR_ EvalKit.ioc" in the SMR_EvalKit_Project folder.





Morpho headers

These headers give access to all STM32 pins.



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

SMR-EvalKit Pin Configuration b)

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
ТСК	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

Software components brief description C)

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 registersWrite 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 *):

A careful consideration should be taken when changing the parameter "SAMPLE_ 1*): 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
Con- tent	SD	FC	Mod- ula- tion	Number sample	Raw I1	Raw I128	Raw Q1	Raw Q128	FFT-mag 1	FFT-mag 128	CS	ED

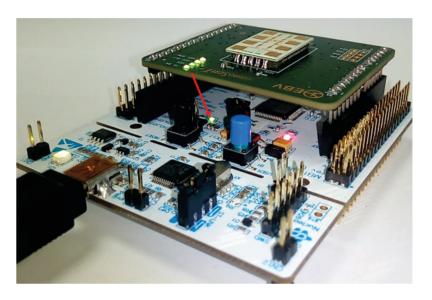
SD: start delimiter = 162 (0xA1)					
FC: function code = 224 (0xE0)	uint8				
Modulation: 0-Doppler, 1-FMCW	uint8				
Number sample: number of captured samples for single measurement	uint16				
Raw I1: sample 1st of raw I signal					
Raw I128: sample 128th of raw I signal					
Raw Q1: sample 1st of raw Q signal					
Raw Q128: sample 128th of raw Q signal					
FFT-mag 1: magnitude of FFT sample 1st in logarithmic scale					
FFT-mag 128: magnitude of FFT sample 128th in logarithmic scale					
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.



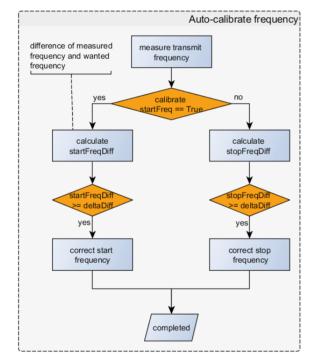
Signal Processing

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



Frequency Auto-Calibration 8.

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.



In the first step the transmit frequency of SMR module is measured. a) 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.

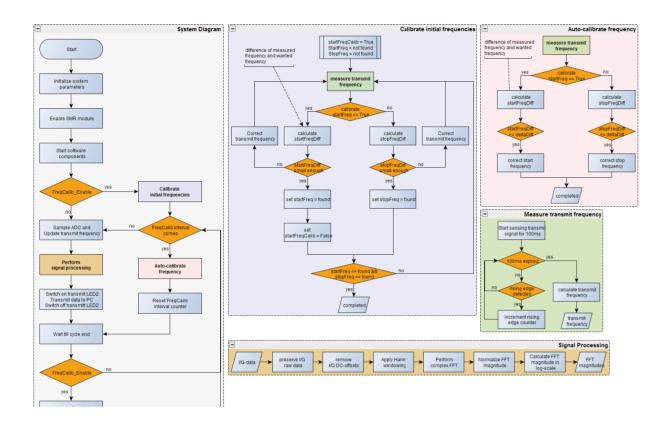
Transmit frequency correction C)

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

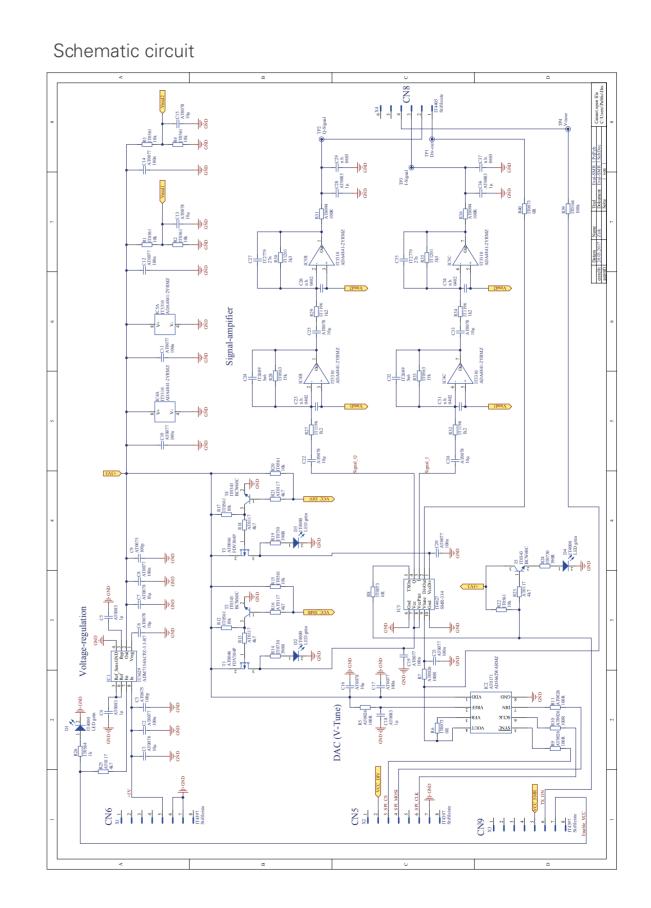
This frequency auto-calibration routine is repeated after a defined duration and this defined duration can be configured by the users. After calibration process, the application will wait for one measurement cycle before continuing with measurements.

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

9. Complete Application Diagram

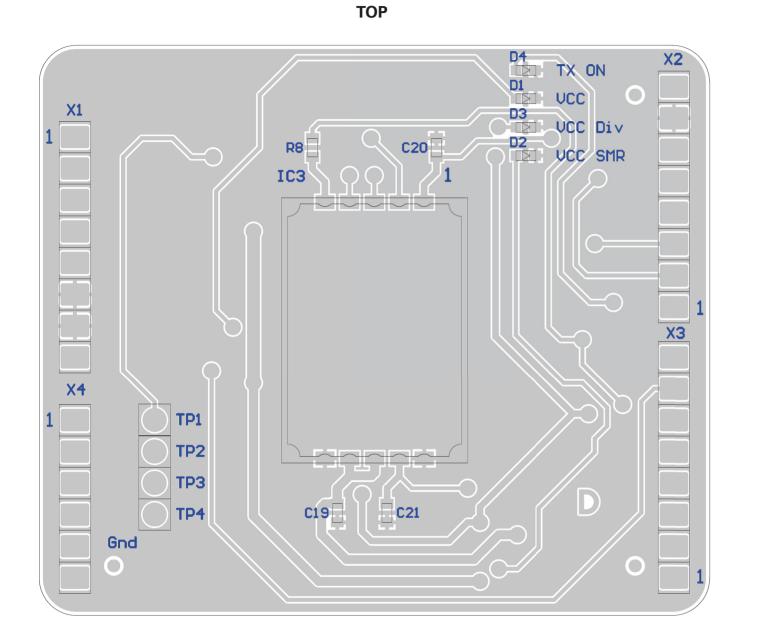


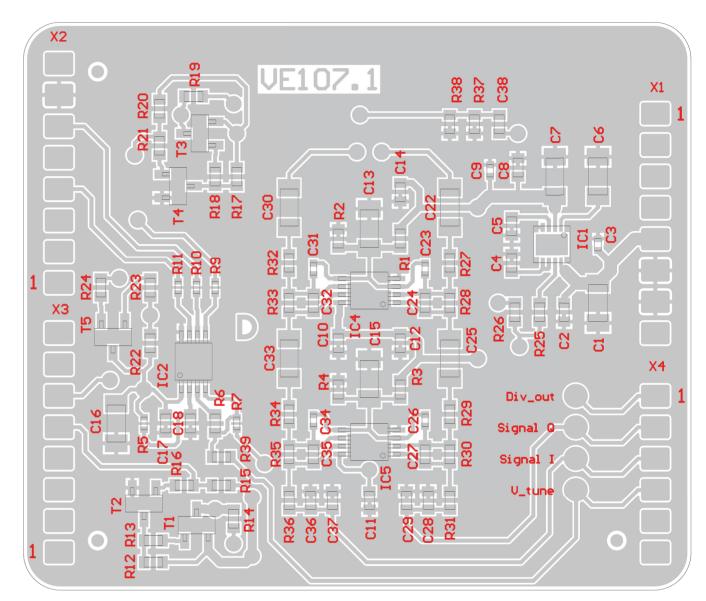
10. Hardware



24

Layout





SMR Evaluation Kit user manual

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

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