5G Communication with a Heterogeneous, Agile Mobile network in the Pyeongchang Winter Olympic CompetitioN

Grant agreement n. 723247

Deliverable D3.3
Beamforming antennas and front-end integration

Date of Delivery:	15 January 2018
Editor:	Giuseppe Destino (UOULU)
Associate Editors:	
Authors:	Tommi Kallio (Nokia), Aki Korvala (Nokia), Giuseppe Destino (UOULU), Marko Leinonen (UOULU), Praneeth Susarla (UOULU), Olli Kursu (UOULU), Jani Saloranta (UOULU), Aarno Pärssinen (UOULU), Marko Pettisalo (NOKIA)
Dissemination Level:	PU
Security:	Public
Status:	Final
Version:	V0.3
File Name:	5GChampion_D3_3
Work Package:	WP3

Abstract
Following up the specifications of the mmW transceiver in D3.1, the work on the implementation and integration of the mmW transceiver was carried on. In this deliverable we provide a thorough description of the component, integration design as well as inter-component interfaces of the RF front-end. Focus is on the key components of the mmW transceiver, namely, the antenna, the RF beamformer, the RF digital-control and RF-software. Measurement results of specific components are shown to prove the working integration of the prototype.

Index terms
5G, mmWave, backhaul link, algorithm, beamforming.

Contents

List of Acronyms

3GPP	3rd Generation Partnership Project
5G	5th Generation
5GTN	5G Test network
AGC	automatic gain control
BB	Baseband
BRU	Backhaul Radio Unit
BTS	Base station
CA	Carrier aggregation
CC	Component carrier
C-RAN	Cloud radio access network
CRC	Cyclic redundancy check
dBi	decibel isotropic
DU	Digital unit
EIRP	Effective isotropic radiated power
EU	European union
GaN	Gallium nitride
GP	Guard period
HAL	Hardware Abstraction Layer
HW	Hardware
IF	intermediate frequency
KPI	Key performance indicator
KR	Korea
LNA	low noise amplifier
LO	local oscillator
LoS	line-of-sight
MCS	Modulation coding scheme
MHN-E	Mobile Hotspot Network Enhancement
MIMO	Multiple-input-multiple-output
MME	Mobility Management Entity
MMIC	monolithic microwave integrated circuit
mmW	millimetre wave
MUX	Multiplexer
MPI	Message Passing Interface
MWB	Mobile wireless backhaul
OFDM	Orthogonal frequency-division multiplexing
OIP	output intercept point
PA	Power amplifier
PCB	Printed circuit board
PoC	Proof of Concept
QAM	Quadrature amplitude modulation
QPSK	Quadrature Phase Shift Keying
RF	Radio Frequency

RF-DFE	Radio Frequency Digital Frontend
RoF	Radio-over-Fiber
RSS	Received signal strength
RSSI	Received signal strength indicator
RU	Radio Unit
SDN	Software defined networking
SFBC	Spatial frequency block code
SNR	Signal-to-Noise Ratio
SPDT	Single point double throw
TBS	Transport block size
TDD	Time Division Duplex
TE	Terminal equipment
UE	User equipment
WP	Work package

Table of Figures

1 Introduction

Millimeter wave communication is one of the main disruptive technologies in coming 5G mobile network. One of the first candidate applications, commercially ready by 2020, is the wireless backhaul link or wireless last mile communication in the 24-29GHz (see D2.2 Chapter 3.1 for a specific discussion on frequency allocation) frequency band.

In the literature, however, only a few implementation of 28GHz transceivers have been proposed. For instance, in [1] a 28GHz CMOS direct conversion transceiver is proposed with 8 antennas integrated in the same package. However, there is a limitation of transmission power since single amplifier saturated output power (Psat) is 10.5dBm. This limits conducted transmission power 3dBm with LTE modulation. In [2] 32 path transceiver is presented with 16dBm Psat output power from each path and this indicates ~6 dBm conducted modulated signal power with 10dB peak-to-average ratio (PAR).

In the 5GCHAMPION project the focus is on the design, implementation and validation of an RF transceiver [3] operating in the frequency-band 27.5 − 28.5 GHz that, on the one hand, it represents a partially common frequency band endorsed by the European, Korean, US and Japanese Frequency Spectrum Authorities, and on the other hand, it is specific for 5G trials at the Winter Olympics in Korea [2]. The proposed mmW solution can be considered the first complete solution, close to a pre-commercial product, developed in a research problem achieving millimeter wave (mmW) urban backhaul requirements, *i.e.,* high-data rate (>2Gbit/s) and medium-long range coverage and robustness to dynamic changes of the environment.

The proposed approach is capable of 24 dB more conductive power than the IC-based prototype shown in [1],[2]. In fact, the 5GCHAMPION mmW RF unit is capable to product radiated power 60dBm Effective Isotropic Radiated Power (EIRP), while IC based transceiver are providing 24dBm EIRP [1] and 28dBm [2]. Also, it supports higher MIMO capabilities and adaptive bamforming to optimize the quality and reliability of the transmission.

The reminder of this deliverable is organized as follows. In Section 2, a brief review of the mmW transceiver design and specifications. In Section 3 we show the implementation of the mmW components and relative measurement results. In Section 4, RF control software is described from an architecture and functionality viewpoint. In Section 5, the inter-component interfaces are detailed and relative test results are provided. Finally, in Section 6 concluding remarks are given.

2 Overview of the demonstration platform

The demonstration mmW platform developed by EU partners comprises of three units; BBU, TRX Radio Unit and Antenna Unit. Detailed specifications are provided in the previous deliverables D2.1 and D2.2. Key design parameters are summarized in **Erreur ! Source du renvoi introuvable.**.

Design Parameters	Values
Frequency	28 GHz
Bandwidth	up-to 800MHz
EIRP	up-to 60dBm

Modulation order	QPSK, 16QAM, 64QAM
MIMO configuration per Antenna unit	2x2 MIMO
Maximum throughput of MWB (demonstration)	2.5Gbps

Table 1: Summary of the RF unit specification

The mmW backhaul link is established with two complete set of these systems, which can be located in LOS and NLOS outdoor environment.

Starting from the radio side, a backhaul system includes two Antenna Units, which are synchronized and controlled by a master RF control unit. These units are entirely designed within the 5GCHAMPION project. Each antenna unit is designed for a 2x2 MIMO link using antenna polarization separation. The distance between the antenna units is adjustable to reduce the inter-beam interference. Each antenna can steer independently two beams allowing robustness to non-line-of-sight channel conditions as well as dynamic changes of the environment. The maximum radiation power of the one antenna unit is (approximately) 60dBm

The TRX unit, also referred to as Digital Front End (DFE), is a prototype developed by NOKIA. It includes 8 digital paths. The Base-band unit is separated from TRX Radio Unit and Antenna Units and is connected to those via optical cable.

In Figure 1 the demonstration platform is shown. Notice however that current assembling has only one antenna unit per pole. Later, two radio units will used to provide full data-rate capacity.

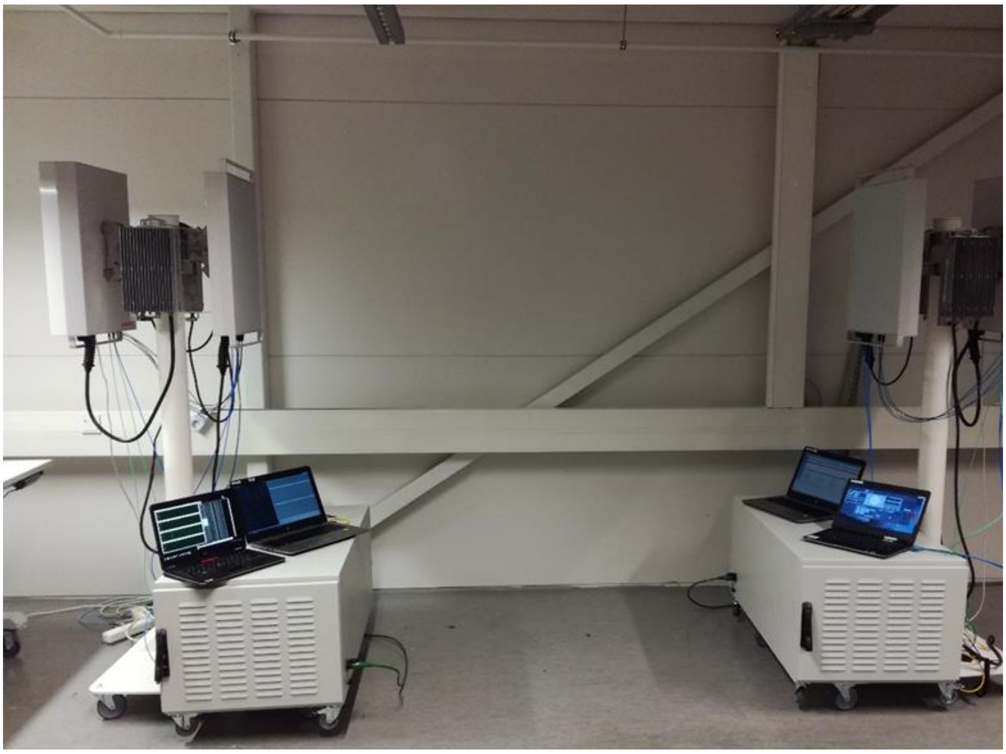


Figure 1: 5GCHAMPION EU mmW testbed

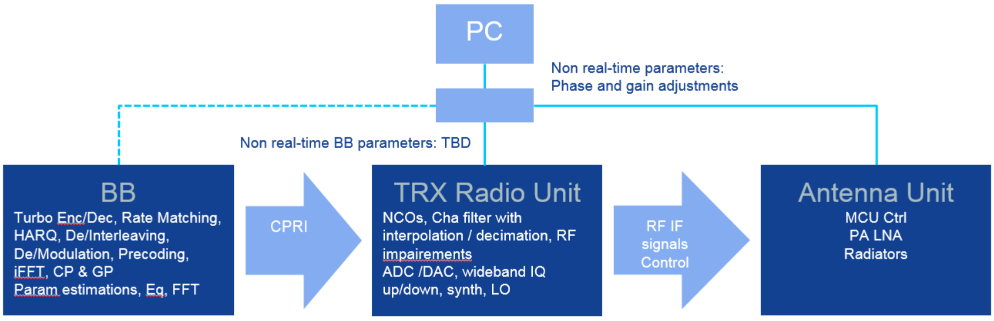
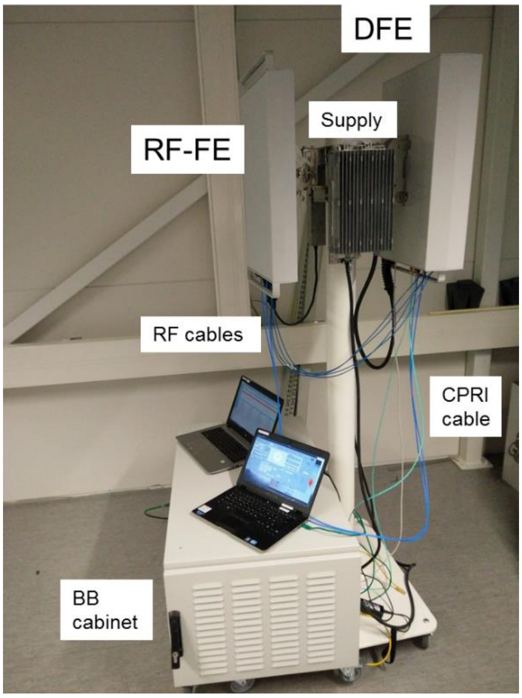


Figure 2 Functional split of different modules in the 5GCHAMPION EU mmW backhaul testbed

Figure 2 shows the block diagram of the system integration. The BB unit implements a 5GTF-like / type physical layer and it is fully developed by NOKIA. It supports up-to 8 MIMO-OFDM channels, with 100MHz Carrier Component and up-to 64QAM modulation. Multiple carrier components can be aggregated in the frequency domain, reaching an overall maximum bandwidth of 800 MHz. In a configuration with 4 BB cards (maximum capacity for the DFE), 20Gbps link can be achieved. Operational frequency at 26.5-29.3 GHz, focus on 26.5 to 27.5 GHz used in the Olympics.

Figure 3shows the integration of the aforementioned blocks.



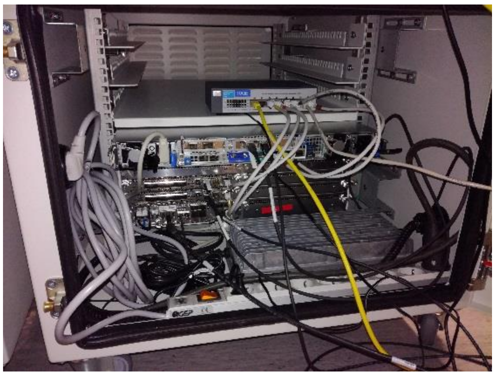


Figure 3: Integration of RF-FE, DFE and BB (left). Detail of the BB (right)

The TRX unit (Digital front-end) is a pre-commercial product developed by NOKIA and implements all ADC/DAC conversions, interpolations/decimations, RF impairments corrections, wideband IQ up/down conversion, etc. In transmission mode, it generates up-to eight RF signals in the intermediate frequency with maximum bandwidth of 400 MHz.
Also the TRX unit provides synchronization signals to the antenna unit, for instance, to switch between TX and RX mode, measure RSS power and change beamforming weights.

A master PC controller is only utilized in the testing and calibration phase of the antenna units.

3 Antenna Unit

3.1 RF beamformer

3.1.1 Architecture and implementation

A block diagram of an architecture of the antenna unit which is shown in Figure 4. The block diagram presents main system blocks and interfaces to the antenna unit. The antenna unit includes three main blocks: Two similar radio boards, one AUX board and one RF digital control unit. In full operational mode, two antenna unit are used.

The radio board receives and transmits the mmW signal from the antenna unit. The design is explained in details in D3.1 and [5]. The RF beamformer is implemented in this unit and it drives a rectangular array with 16 ports. The prototype of one RF board is shown in Figure 5.

The AUX board generates the operational voltages for the antenna unit and distributes the reference clock signals within the antenna unit. Additionally it routes incoming reference signal to the second antenna unit. The main physical interfaces are summarised in the Table 2.

The control unit is the STM32 Nucleo-144 development board with STM32F429ZI microcontroller in LQFP144 package. It controls the operation of the antenna unit (beamformer, local oscillator, power measurement, scheduling and amplifiers). The control unit is synchronized with the TRX radio unit via timing signals that indicates TX/RX slot as well as radio-frame start (10ms periodic signal). In operational mode, the control unit can autonomously change the received and transmitted signal power by controlling the radio boards, adapt beamdirection and beamwidth based on some optimization criteria, e.g., maximize the signal-to-noise-ratio. Also in the set-up with four antennas, two control units are networked and configured in a master-slave mode.

Table 2 Interfaces of the antenna unit

Signal reference number	Signal name	Signal function
1	REF_CLK_IN	Reference clock input
1a and 1b	-	Filtered reference clock signal for RF signal generation
2	REF_CLK_OUT	Reference clock output
3	TXRX	Transmission and reception timing signal
3a and 3b	-	Control signals of the radio boards
4	DC48V	48V operational voltage
5	GND	Ground signal
6	FRAME_START	Start of the signal frame
7	-	Regulated operational voltages of the radio board
IF_IN	IF_IN	Communication signal interface

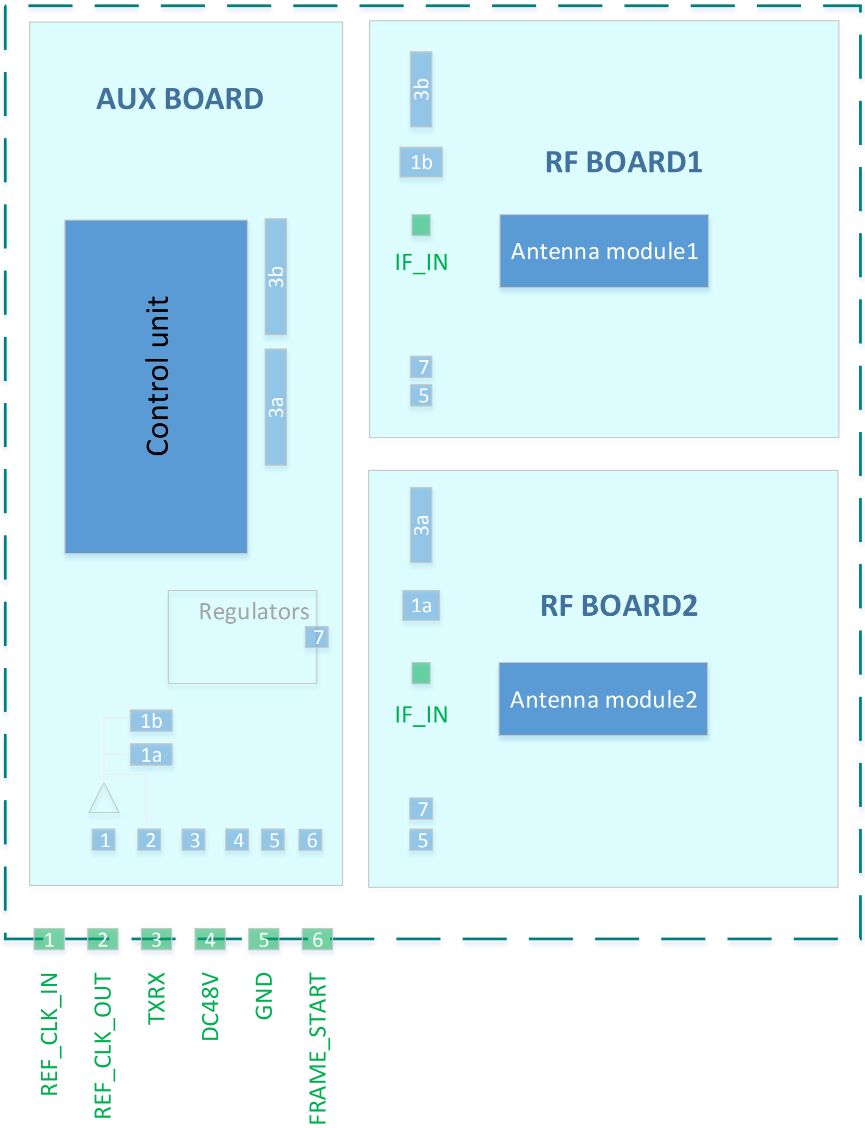


Figure 4 Block diagram of the EU mmW antenna unit

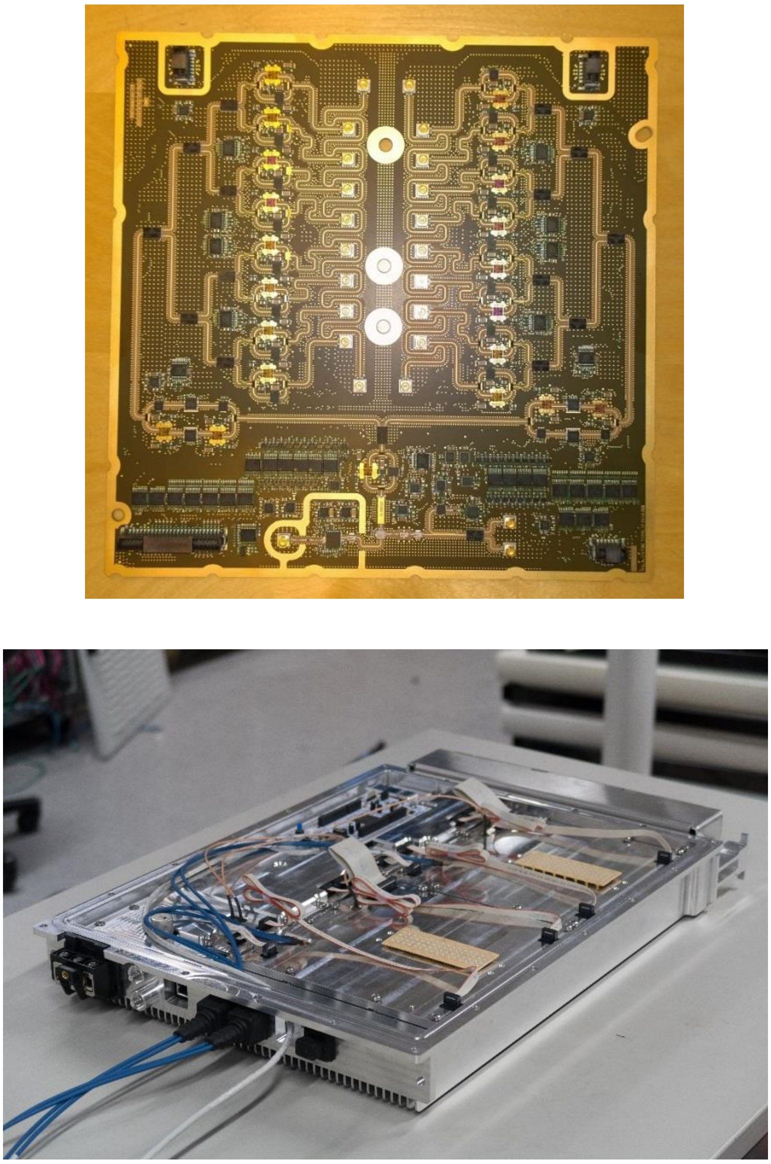


Figure 5: RF board prototype. RF board (top) and RF with mechanics (down)

3.1.2 Antenna

The EU mmW antenna unit has two antenna modules and those are connected to two radio cards. The antenna implementation is described in details in [4] [5]. A first prototype of the 2×8 antenna matrix is shown in Figure 6. It comprises of 16 sub-arrays Thus, the total number of patch elements in the array matrix become 64 and the size of the array is 90 × 34 mm^2 (*W*x*L*). The structure is presented without SMPM-connectors and metal cover over the feed network

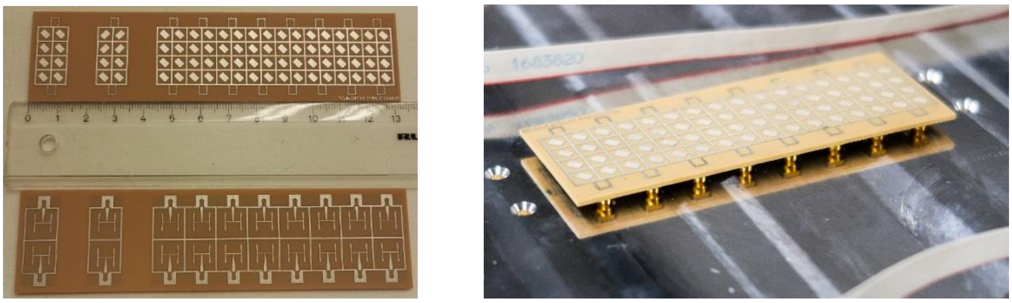


Figure 6. Photograph of the prototype antenna array..(Left) Antenna array with testing subarray. (Right) Antenna mounted on the platform

Figure 7 presents the measured radiation patterns in azimuth plane (XZ-cut) at 26.50 GHz and 27.50 GHz. The radiation patterns are measured element by element and the results are summed in post-processing. Sidelobe level are 15 dB below the maximum gain which was predicted. The maximum gain is around 20 dBi as the theory is predicting max. 21.5 dBi so there is a good correlation between the results.

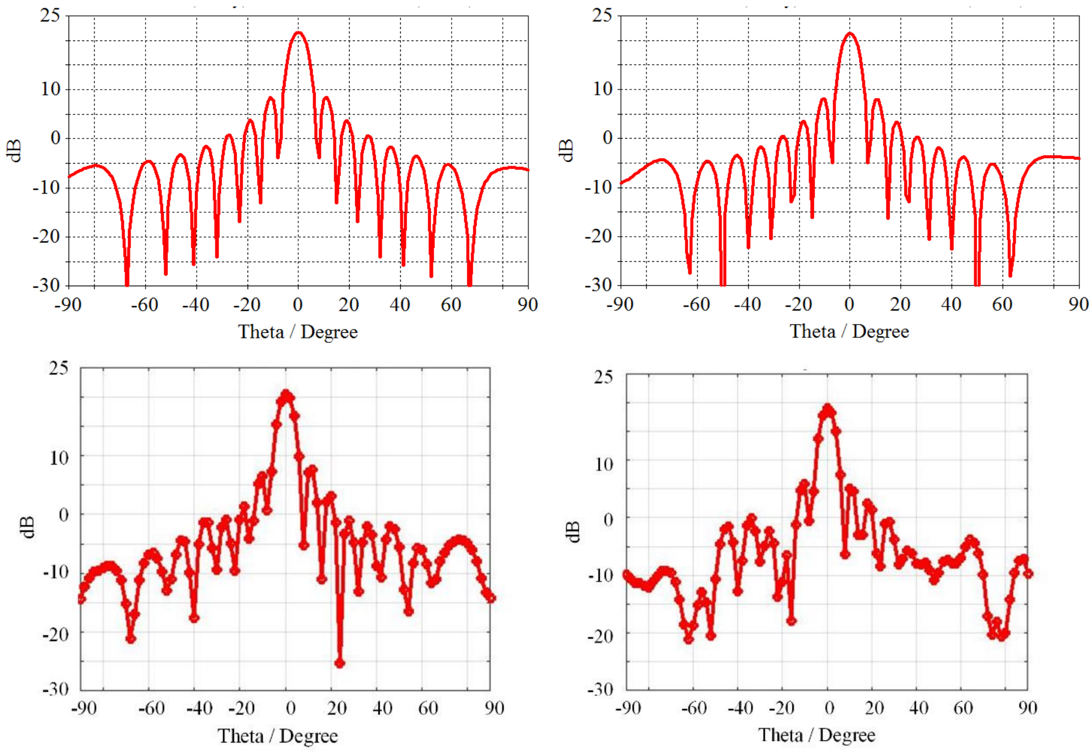


Figure 7: Simulated (left) and measured (right) radiation patterns of the array matrix in azimuth plane at 26.50 GHz and 27.50 GHz.

3.1.3 Beamformer

The radio module includes 16 similar radio signal paths which all support both transmission and reception. The TX/RX switch is directly driven by the DFE. The transmission and reception signals are combined with four cascaded two-port Wilkinson power dividers/combiners. The architecture of the radio solution or the beamformer is shown in Figure 8.

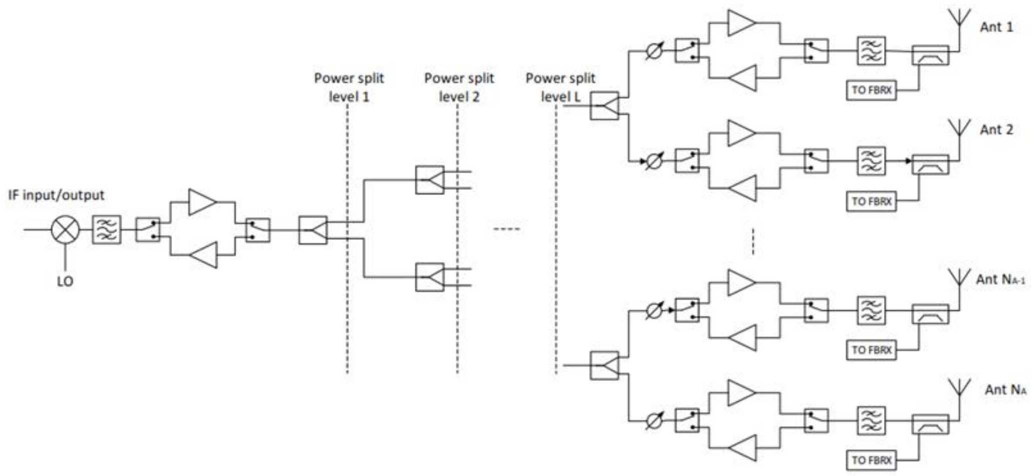


Figure 8 Architecture of the radio system or radio beam former

A direction of the mmW beam is controlled with passive mmW phase shifters which are used for both transmission and reception beam controls. The phase of value of each transmitter and receiver path can be controlled with 11.25° steps since the phase value is controlled with 5 control bits. The initial beam direction can be selected according to a method described in [3].

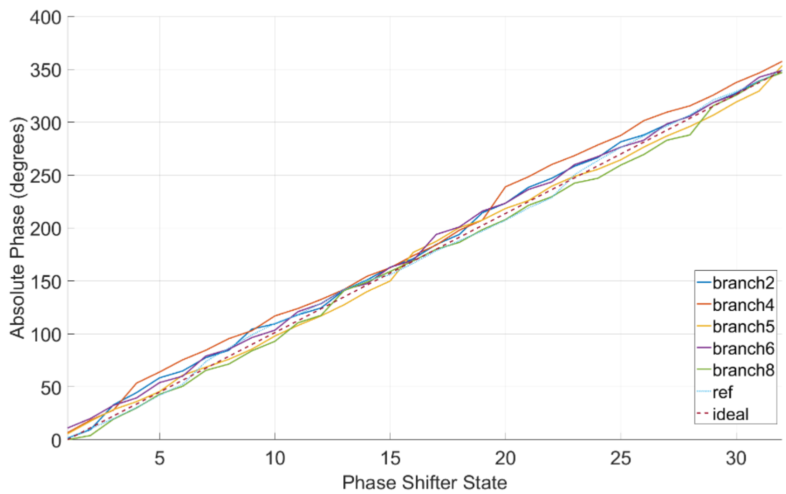


Figure 9: Measurements of the phase values for selected phase-shifters.

Figure 9 shows the comparison between the measured phased value ad reference ones.

3.2 RF control unit

3.2.1 Architecture

The main control unit (MCU) is located at the AUX (auxiliary) board. The control unit is implemented with a ST Nucleo processor off-the-shelf module. The AUX board includes the following functions:

- Provides an interface between one Nucleo MCU and two RF boards
- Provides voltage regulation for both RF boards, including 20V, 6V and -15 V power supplies regulated from a single 48 V telecom power supply.
- Controls PA bias-up and -down procedures.
- Divides, pulse shapes and routes the 230.4 MHz reference clock to both RF boards and the second Antenna unit.
- Receives differential TXRX and Frame start signals from the TRX unit and routes these as single-ended signals to both the MCU and two radio boards.
- Receives and A/D converts RSSI information from the RF units and routes these to the control MCU.
- Provides fan control for cooling the antenna unit.

A photograph of the AUX board is shown in Figure 10. The blue module is the Nucleo module and the bottom of the antenna unit is at the right side of the picture. The reference clock signal cables are seen at the middle of the right. The black ICs are regulators of the operational voltages of the RF boards. A wide signal connector at the left convoy RF communication signals to the radio board.

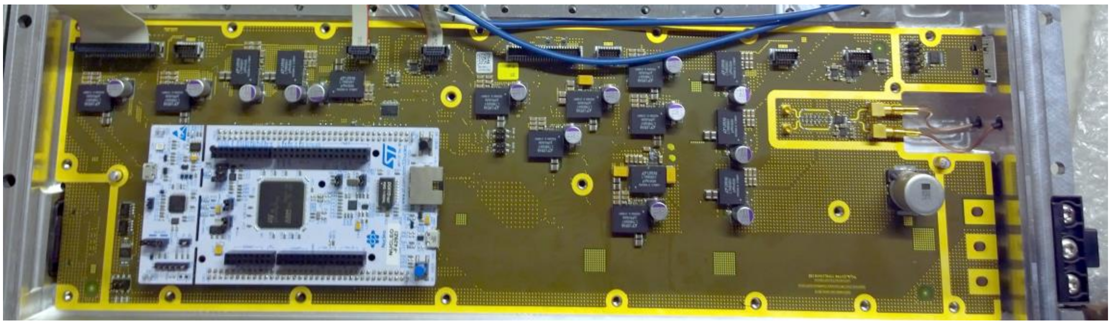


Figure 10. Photograph of the AUX board

3.2.2 MCU and peripheral devices

In addition to direct I/O controls, two SPI buses and two I2C buses on the Nucleo MCU are used to control two RF boards.

The SPI buses are used to control the local oscillator (LO) generation and the beamformer shift register that controls both the phase shifter settings and the mmW signal attenuator settings.

The I2C buses are used to control multiple devices, most importantly PA gate bias programming. The structure of the I2C bus is shown on Figure 11.

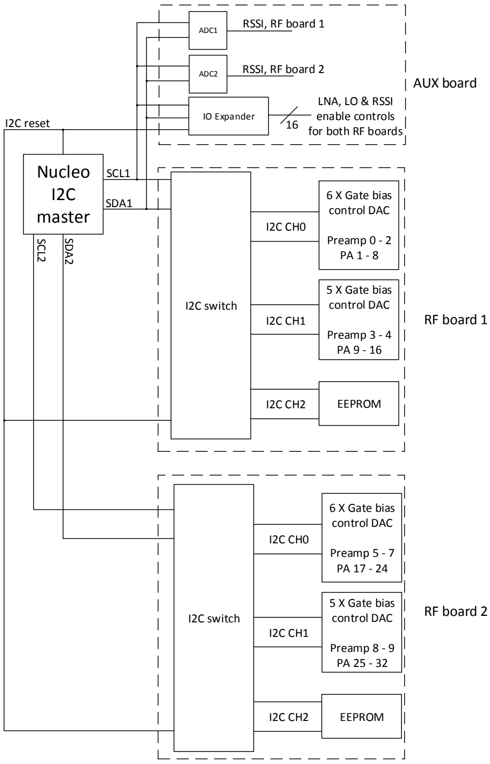


Figure 11. Structure of the implemented I2C bus configuration.

3.2.3 PA control

The AUX board routes 42 PA gate bias enable control lines as direct I/O controls from the Nucleo to two RF boards. Gate bias voltage is controlled via the I2C bus, as shown in Figure 11. Enabling the 20 V drain supply of PA is controlled by the Nucleo MCU. The Nucleo controls the enable inputs of twelve 48 V to 20 V switching regulators located on the AUX board.

3.2.4 BF control

The shift registers which are controlled via SPI buses change the phase values of the mmW phase shifters.

3.2.5 LO control

The LO is controlled with SPI interface by loading the needed divider values to the synthesizer circuitry. The LO reference frequency is 230.4MHz which is received via AUX board from reference clock as shown in Figure 4.

3.2.6 RSS control

A received signal strength (RRS) indicated the received mmW signal power in the output of the radio receiver. The RSS is measured with a received signal strength indicator (RSSI) circuitry and the block diagram of the RSS circuitry is shown in Figure 12 [6]. The attenuation of the received signal is controlled based on RSSI signal level.

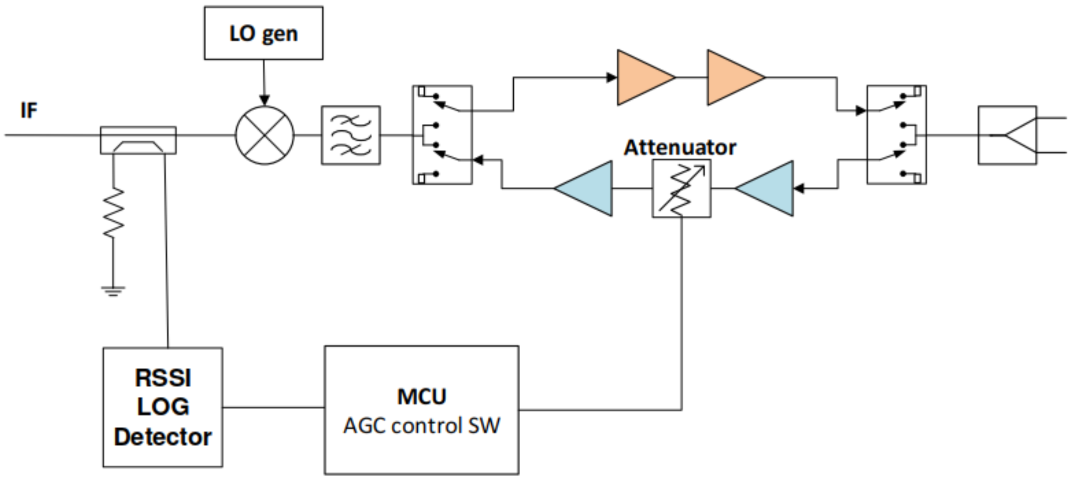


Figure 12. RSSI measurement topology.

4 RF control software

The RF-control software architecture is depicted above. It comprises of:

- **Application layer**: BTS and UE software. These applications manages the overall functionality of the radio boad. In the 5GCHAMPION project they are mostly responsible of the beam alignment procedure and beamforming.
- **Middleware**: The middleware comprises of all control threads, which will run simultaneously, communicate via message passing interface (MPI) and use the hardware abstraction layer (HAL) library.
- **Hardware abstraction layer**: The HAL is a library of functionalities that control the drivers.
- **Driver**: The drivers are HW specific and include procedures that operate directly on signals (input/output of the MCU).

4.1 Thread Model

The middleware contains control threads mentioned below :

Middleware Thread name	Thread ID value
SCHEDULER_CTRL	0
BF_CTRL	1
PA_CTRL	2
FREQUENCY_CTRL	3

AGC_CTRL	4
COM_TX_CTRL	5
COM_RX_CTRL	6
RSS_CTRL	7

Each thread follows the following implementation model. All threads are initiated simultaneously and become active at the occurrence of an event.

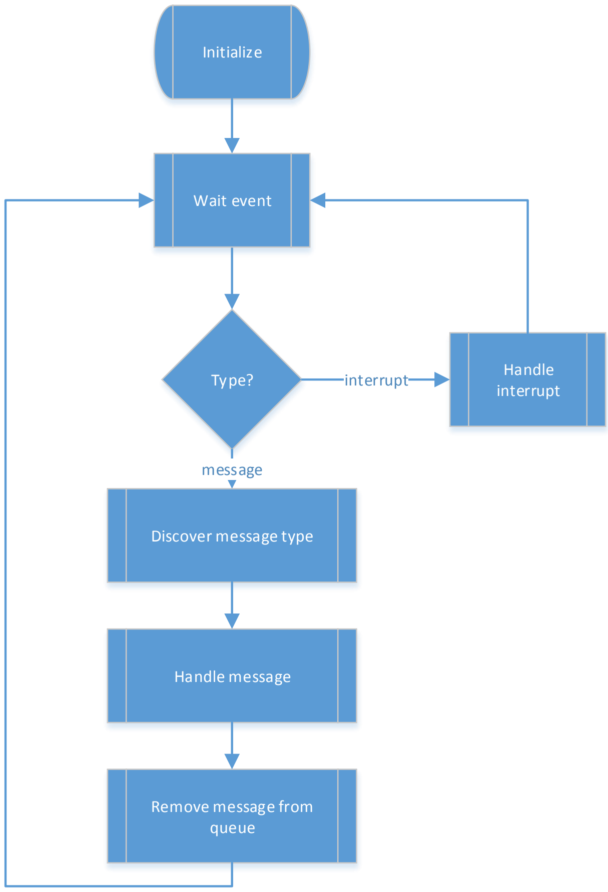


Figure 13: Thread model

Events are notified by signals that, in some cases, are also associated to a message. The message has a source, a destination and a payload. The signal is denoted by MPI_SIGNAL, whereas the message by MPI_PACKET. In the message, a specific container for the signal receiver is defined, i.e., msg_data.

4.2 MPI communication protocol

Signals are managed by a Message Passing Interface, which is built upon the signalling functionalities of the operating system. All communications between threads uses the MPI protocol. Figure 14 shows two different interfaces which are identified by specific MPI signals.

The interface between internal threads of the RF software is implemented with MPI_SIGNAL, MPI_SIGNAL_ACK and corresponding messages, MPI_MESSAGE and MPI_ACK, respectively.

An interface with interrupt is defined between the TRX_MODE_CTRL and the HAL of the BF_CTRL. This serves for the synchronization of the beamformer register latching.

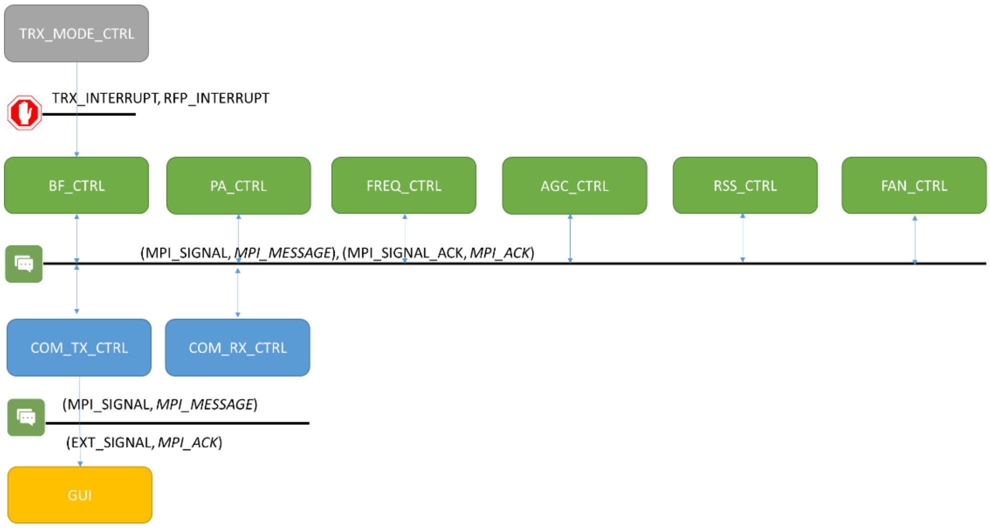


Figure 14: MPI communication model

4.2.1 MPI_packet format

The MPI_PACKET format is described below.

src_thread_id	rec_thread_id	pdu_id	pdu_len	msg_data (void pointer)

```
typedef struct
{
	uint8_t src_thread_id;
	uint8_t rec_thread_id;
	uint8_t pdu_id;
	uint8_t pdu_len;    // Length of header plus message
	uint8_t *msg_data;  // Pointer to the data
```

```
} __attribute__((packed)) MPI_PACKET;
```

It includes the following fields:

- *src_thread_id* is the source thread ID
- *rec_thread_id* is the receiver thread ID
- *pdu_id* are ID's specific to a thread/operation.
- *pdu_len* is the length of header plus message. maximum PDU length is 500 bytes.
- *msg_data* is pointer to the information to be sent over MPI

4.2.2 MPI_Client format

Each thread is associated to an MPI_client data model.


```
typedef struct
{
       uint8_t       registered;
       osThreadId    thread_id;
}__attribute__((packed)) MPI_CLIENT;
```

The fields of this data model are:

- *registered* is to check if the thread is in registered or unregistered state
- *thread_id* is the ID related to client thread

4.2.3 MPI Interrupt

The interrupt used for the synchronization of the beamformer shift-register writing is defined as follows. The callback function commands the SPI interface to start shift register writing.

Event	Code	From	To	Purpose
RFP_INTERRUPT	Fall-edge	GPIO	HAL_BF	synchronize the update of the shift-register beamformer

4.2.4 MPI Signals

The MPI_SIGNAL and MPI_SIGNAL_ACK are defined as follows.

Event	Code	From	To	Purpose
MPI_SIGNAL	0x0002	All	All	notify incoming MPI message
MPI_SIGNAL_ACK	0x00F2	receiver	sender	notify that message has received

No recovery mechanisms are implemented in case of failure (ACK not received).

4.2.5 MPI PDU control ID

The type of packet data unit (PDU) is thread specific. The following table shows the complete set and corresponding purpose.

PDU NAME	ID	Purpose
BEAMFORMER CTRL		
MPI_BF_SET_ID	0	Set the beamformer weights
MPI_BF_RESET_ID	1	Set default value for beamformer weights
MPI_BF_TUNE_ID	2	Tune specific weight/attenuation of the beamformer
MPI_BF_TRIGGER_ID	3	External control of latching
MPI_BF_TRIGGER_AUTO_ID	4	Auto latching command from MCU
FREQUENCY CTRL		
MPI_FREQ_START_ID	0	Initialize local oscillator frequency
MPI_FREQ_CHANGE_ID	1	Change frequency
MPI_FREQ_RESET_ID	2	Reset local oscillator frequency
MPI_SYNTHESIZER_ON_ID	3	Power on the synthesizer
MPI_SYNTHESIZER_OFF_ID	4	Power off the synthesizer
RSS CTRL		
MPI_RSS_START_PERIODIC	0	to notify periodic reading of RSS signal
MPI_RSS_READ_SINGLE	1	to notify single reading of RSS signal
MPI_RSS_STOP_PERIODIC	2	to stop periodic reading of signal
FAN CTRL		
MPI_FAN_OFF	0	Power on the fan
MPI_FAN_ON	1	Power off the fan

MPI_FAN_SPEED	2	Set fan speed
POWER AMPLIFIER CTRL		
MPI_PA_CALIB_ADDVALUE	1	to notify calibrating PA with given value
MPI_PA_CALIB_DONE	2	to notify calibration stop
MPI_PA_LNA_ENABLE	3	to notify enabling LNA
MPI_PA_LOAD_EEPROM	4	to notify loading EEPROM values
MPI_PA_DEBUG_MODE	5	to notify the a different mode of debugging (no voltage calibration)
MPI_PA_OPERATIONAL_MODE	6	to notify that PA do not need debug

The transmission of packet is indicated by an MPI_SIGNAL and the payload is sent in the MPI_MESSAGE. The latter is constructed with the *mpi_packet* data structure. In this data structure, the actual MPI_PDU is pointed by msg_data.

Each control thread after receiving the information in MPI packets converts the data into their respective structure formats internally in their implementation. Such structure formats of all control threads are defined under *mpi_pdu.h*

4.2.5.1 MPI_BF_CTRL_message

```
typedef struct
{
	uint8_t board_index;
	enum BFTYPE type;
	float  angle;
	uint8_t 	level;
	float attenuation[4]; //TX:1, TX:2, RX:1, RX:2
} __attribute__((packed)) MPI_BF_CTRL_message;
```

4.2.5.2 MPI_BF_CTRL_TUNE_message

```
typedef struct
{
	uint8_t board_index;
	enum TUNETYPE type;
	uint8_t 	el_index;
	float  value;
} __attribute__((packed)) MPI_BF_CTRL_TUNE_message;
```

4.2.5.3 MPI_FREQ_CTRL_message

```
typedef struct
{
	uint8_t board_index;
	float frequency;
} __attribute__((packed)) MPI_FREQ_CTRL_message;
```

4.2.5.4 MPI_RSS_CTRL_message

```
typedef struct
{
	uint8_t board_index;
	float power;
} __attribute__((packed)) MPI_RSS_CTRL_message;
```

4.2.5.5 MPI_PA_CALIB_CTRL_message

```
typedef struct
{
	uint8_t board_index;
	uint8_t pa_index;
	float bias_value;
	uint8_t store;
} __attribute__((packed)) MPI_PA_CALIB_CTRL_message;
```

4.2.5.6 MPI_PA_LNA_ENABLE_message

```
typedef struct
{
	uint8_t board_index;
	int value;
} __attribute__((packed)) MPI_PA_LNA_ENABLE_message;
```

4.2.5.7 MPI_PA_LOAD_EEPROM_message

```
typedef struct
{
	uint8_t board_index;
	uint8_t pa_index;
	float bias_value[NUMBER_OF_TOTAL_PA];
} __attribute__((packed)) MPI_PA_LOAD_EEPROM_message;
```

4.3 Middleware

The middleware logic for the software comprises of the following control threads:

4.3.1 Beamformer_ctrl_thread

Purpose

The thread acts as a middleware between the input received via MPI from communication_ctrl_thread and beamforming algorithm functionality. Beamformer control thread decodes the MPI message and calls the corresponding lower level functions.

Features

The thread includes functionalities such as setting the beamforming type and options resetting the all beamforming related options to initial values, tuning the beamforming based on individual phase shifter values, setting the triggering mode. Each one of these functionalities are performed with the lower level of the architecture using MPI communication protocol.

4.3.2 Communication_ctrl_thread

Purpose

The thread acts as a bridge and helps in communicating the data between Nucleo and external application layer using Ethernet packets and message passing mechanisms. In its implementation, the data received from application layer, are decoded based on their PDU id and sent to the corresponding control thread. Similarly, the thread receives information from other control threads via MPI and transmits the data to application layer via UDP.

Features

The thread includes functionalities such as ‘communication send’ and ‘communication receive’. ‘Communication send’ handles MPI thread communication where the message received from other control threads are to be transmitted via UDP to the application layer. ’Communication send’ handles RSSI and PA control thread transmissions to the application layer. Similarly, ‘communication receive’ helps in receiving the Ethernet packets from application layer and pass the information to RSS and PA control threads through MPI.

4.3.3 Frequency_ctrl_thread

Purpose

The thread acts as a controller of the local osclllator in the radio boards. It can be enquired from internal or internal applications.

Features

The thread contains functionalities such as set LO frequency, change LO frequency, reset LO frequency, switching on the synthesizer and switching off the synthesizer. Each one of these functionalities are performed with the lower level of the architecture using MPI communication protocol.

4.3.4 PA_ctrl_thread

Purpose

The thread helps in controlling the PA’s on RF boards. All the PA related information, coming from the application layer are received as MPI packets by this thread. Based on the received information, the logic then invokes corresponding HAL layer functionality related to PA’s.

Features

The thread includes functionalities such as adding a calibration value to PA, stop calibration of PA, Loading EEPROM at application layer with calibrated PA's and enabling particular PA/LNA during calibration and debug phases. Each one of these functionalities are performed using MPI communication protocol.

4.3.5 RSS_ctrl_thread

Purpose

The thread reads RSSI signal information from HAL layer on the Nucleo and sends the information as MPI data to communication control thread.

Features

The thread includes functionalities such as reading a RSS signal from RF unit periodically, a single RSS read from RF unit and stop reading RSS signal periodically. Each one of these functionalities are performed using MPI communication protocol.

4.3.6 Scheduler_ctrl_thread

Purpose

This thread helps in scheduling the signal synchronization tasks, periodic RF beam alignments of the transceiver and handling communication between BTS and UE in application layer while fixing RF beam directions

Features

The thread maintains the synchronization between control threads by scheduling interrupts at the falling edge of transmission signal

4.3.7 Fan_ctrl_thread

Purpose

This thread controls the fan speed of the RF board.

Features

Based on temperature readings from the RF board internal sensors, temperature is kept within the optimal value range.

4.3.8 AGC_ctrl_thread

Purpose

This thread performs the automatic gain control on the receiving signal. It is key for handling mobility

Features

It operates on the beamformer attenuators to maintain the receiving signal power within the optimal operational range.

4.3.9 Hardware Abstraction Layer (HAL)

HAL library for this project consists of the following abstraction, almost one for each control thread in the middleware

4.3.9.1 HAL_BF

Purpose

HAL_BF helps in providing the beamforming weights to steer the beam to angle direction and also the root-beam specific for a BF type.

Features

HAL_BF has got abstracted functionalities such as

- ‘quantize angle’ and ‘quantize attenuation’ to map the angle to a phase-shifter state
- ‘HAL_BF_beamSteer’ to provide the beamforming weights for RF beam steering
- ‘HAL_BF_rootbeaming’ to provide specific BF root-beaming based on narrow, subarray and deactivation beam forming techniques
- ‘HAL_BF_init’ for initializing the beamformers to a factory values
- ‘HAL_BF_set’ and ‘HAL_BF_reset’ to set and reset desired beam former weights after computing them
- ‘HAL_BF_Tune’ to tune the RF beam to a selected parameter value
- ‘HAL_BF_trigger’ and ‘HAL_BF_trigger_auto’ to set the triggers in manual and auto mode respectively

4.3.9.2 HAL_LO

Purpose

HAL_LO helps in providing abstractions for configuring LO register values as well to measure, set and updating LO frequencies.

Features

HAL_LO has got abstracted functionalities such as

- ‘HAL_LO_init’ to initialize the driver for synthesizer control in all RF boards
- ‘HAL_LO_config’ to configure the LO register variable
- ‘HAL_LO_getFactoryValue’ to read the default LO register values
- ‘HAL_LO_set_frequency’ and ‘HAL_LO_update_frequency’ to set the LO frequency to default and to update them with new values respectively
- ‘HAL_LO_enable’, ‘HAL_LO_disable’ and ‘HAL_LO_reset’ to enable, disable and reset the LO synthesizer value respectively

4.3.9.3 HAL_MPI

Purpose

HAL_MPI defines an MPI packet structure and provides abstract level functions for MPI communication protocol

Features

HAL_MPI has got abstracted functionalities such as

- 'mpi_register' to register a thread as an MPI client
- 'mpi_check_client_status' to verify MPI client registration
- 'mpi_check_client_id' to return the registered MPI client id for a thread
- 'mpi_packet' to create a MPI packet for a thread
- 'mpi_send' and 'mpi_receive' to send and receive an MPI message to the receiver and from the sender respectively
- 'mpi_send_signal' to send a given signal to the receiver
- 'mpi_send_ack' to send an Acknowledgement message to the source
- 'mpi_free' to free the message packet allocated from the pool

4.3.9.4 HAL_PA

Purpose

HAL_PA provides abstract level functions for calibration, debug and operational mode of Power Amplifiers

Features

HAL_PA has got abstracted functionalities such as

- 'HAL_PA_startBiasCalibration' to start the calibration mode of PA's
- 'HAL_PA_stepBiasCalibration' to adjust the selected amplifier DAC bias voltages with signed offset value and also to update DAC control register
- 'HAL_PA_stopBiasCalibration' to terminate the calibration phase and accept the calibrated value as well as to save it to EEPROM for a particular PA
- 'HAL_PA_stopAllBiasCalibration' to perform stopBiasCalibration on all PA's of the RF board
- 'HAL_PA_LNA_Enable' to enable/disable the LNA pins on the AUX board using IO expander
- 'HAL_getPA' to read all calibrated PA values from EEPROM

4.3.9.5 HAL_RSS

Purpose

Provide abstract level function to compute RSSI power from RF boards

Features

HAL_RSS has got the abstracted functionality 'HAL_RSS_Read' which reads the RSS voltage value from the driver and computes its power

4.3.10 Drivers

4.3.10.1 amplifier_driver

Purpose

This includes procedures that operate directly on amplifier signals of both RF boards

Features

The driver has got functionalities such as

- 'amplifier_ctrl_init' to initialize Drain PA, LNA and PA I/O control lines
- 'wait_minus_5v_pgood' to wait for -5V to come up
- 'enable_drain_pa' to turn the 20V drain PA power feed on or off
- 'get_drain_state' to read the 20V drain PA power feed state
- 'enable_lna' to enable or disable LNA on RF board
- 'get_lna_enable_state' to read the RX LNA enable states on RF board

4.3.10.2 dac_driver

Purpose

This includes procedures that operate directly on PA DAC's of both RF boards

Features

The driver has got functionalities such as

- 'dac_drvn_init' to initialize driven gate drive control GPIO lines
- 'dac_drv_enable' to enable/disable driven gate drive of amplifier
- 'dac_get_drv_state' to read the driven gate drive state of amplifier
- 'dac_ovrd_init' to initialize DAC control for calibration
- 'set_dac_voltage' to write calibrated bias voltage level on DAC override register
- 'dac_read_calib_values' to read DAC values from internal/external EEPROM. It's used to verify the calibration values on each DAC EEPROM, hence for debug purposes only
- 'dac_store_calib_values' to write DAC calibrated values either to RAM or internal EEPROM
- 'dac_burn_eeprom' to write DAC calibration values to the DAC internal EEPROM
- 'dac_validate_eeprom' to verify that the last DAC EEPROM read or write has completed successfully
- 'dac_read_temp_sensor' to read the value from DAC temperature sensor
- 'dac_change_access_level' to change DAC memory access locations to read or write
- 'dac_set_override' to enable DAC voltage calibration using override values
- 'dac_read_register' and 'dac_write_register' to read and write values to the DAC register via I2C respectively
- 'dac_read_block' and 'dac_write_block' to read and write consecutive DAC registers with a multi-byte sequence respectively

4.3.10.3 eeprom_driver

Purpose

This includes procedures that operate directly on EEPROM on both RFboards via I2C

Features

The driver has got functionalities such as

- 'eeprom_read' to read the PA voltage data from EEPROM via I2C
- 'eeprom_write' to write the PA voltage data to EEPROM via I2C

Though, the prototypes of these functions are maintained under this driver, the actual implementation of these drivers are included in 'shared_i2c_driver'

4.3.10.4 ethernet_driver

Purpose

This includes procedures that operate directly with Ethernet port on Nucleo board

Features

The driver has got functionalities such as

- 'eth_init' to create an Ethernet network for the Nucleo board
- 'eth_packet' to create Ethernet packet from data
- 'send_eth_message' to send Ethernet packet from Nucleo board
- 'receive_eth_message' to receive Ethernet packet on a socket

4.3.10.5 fan_driver

Purpose

This includes procedure that operate directly on fan system on the RF boards

Features

The driver contains 'fan_speed' function in order to set the fan speed via PWM

4.3.10.6 i2c_rss_driver

Purpose

This includes procedures that operate directly on RSSI ADC of Nucleo board

Features

The driver has got functionalities such as

- 'rss_init' to initialize the RSSI Nucleo GPIO

- ‘rss_enable’ to turn on or off the RSSI power regulator
- ‘rss_amp_enable’ to enable/disable the RSS amplifier
- ‘rss_adc_trigger’ to trigger the RSSI ADC conversion on a particular RF board
- ‘rss_adc_configure’ to select the active ADC conversion, reference input and interrupt pin output
- ‘adc_set_cycle_time’ to set delay and automatic sampling interval on ADC
- ‘write16bit_register’ to perform i2c block write of 2 bytes of RSSI data
- ‘rss_adc_set_limits’ to set alert limits on ADC
- ‘rss_adc_read_result’ to read the conversion result on the ADC
- ‘rss_adc_read_alert’ to read the alert status register on the ADC

4.3.10.7 io_expander_driver

Purpose

This includes the procedures that operate directly on IO expander device on AUX board and RF boards

Features

The driver has got functionalities such as

- ‘configure_io_expander’ to initialize the I2C IO expander to output mode
- ‘toggle_io_expander’ to toggle virtual GPIO line via I2C IO expander
- ‘write_io_expander’ to set virtual GPIO lines via I2C IO expander
- ‘read_io_expander_state’ to read the state of virtual GPIO lines
- ‘query_io_expander’ to read the virtual GPIO lines on I2C IO expander. Used only for debug purposes

4.3.10.8 shared_i2c_driver

Purpose

This includes procedures that help in transferring information to the hardware via I2C bus

Features

The driver has got functionalities such as

- ‘i2c_init’ to initialize the common I2C bus
- ‘i2c_reset’ to reset pulses on I2C buses
- ‘int_read_register’ and ‘int_write_register’ to read and write I2C device register without locking
- ‘i2c_read_register’ and ‘i2c_write_register’ to read and write I2C device register
- ‘i2c_read_block’ and ‘i2c_write_block’ to read and write consecutive I2C device registers with multi-byte sequence
- ‘switch_i2c_channel’ to connect DAC’s and EEPROM on each RF board via switched i2c sub-channels

4.3.10.9 shift_register_driver

Purpose

This includes procedures that operate directly on shift registers of both RF boards via SPI interface

Features

The driver has got functionalities such as

- ‘init_board_register’ to initialize SPI interface for shift registers
- ‘push_shift_register_msg’ to transfer register content to RF board via SPI
- ‘spi_callback’ provides a callback function for SPI event
- ‘write_shift_register_msg’ to write data into shift register
- ‘reset_register’ to reset shadow register values
- ‘set_register_values’ to write shadow register values with given Antenna Branch Enable, Phase and Attenuation control.
- ‘set_phase_field’ ,‘set_branch_field’ and ‘set_attenuation_field’ to write into shadow register
- ‘latch_shift_register’ to latch the shift registers
- ‘stcp_callback’ to reset the STCP signal

4.3.10.10 synthesizer_driver

Purpose

This includes procedures that operate directly on synthesizer registers of RF boards via SPI interface

Features

The driver has got functionalities such as

- ‘push_synth_msg’ to transfer the data to synthesizer registers via SPI
- ‘spi_callback’ to handle errors or ACK message to thread during an SPI event
- ‘synth_le_callback’ to set back load enable to high
- ‘init_board_synth’ to initialize SPI interface for the synthesizer
- ‘set_synth_register_values’ to set synthesizer register reference values
- ‘reset_synth_register’ to reset synthesize registers to default values
- ‘update_synth_register’ to update synthesizer registers to given values
- ‘synth_enable’ and ‘synth disable’ to turn on and off the synthesizer respectively

4.3.10.11 txrx_driver

Purpose

This includes procedures that operate directly on TX/RX lines for switches on RF boards

Features

The driver has got ‘set_txrx_state’ functionality to set the state to TX/RX for RF switches

5 Inter-component interface

5.1 Interface between RF beamformer and phased-array antenna

The interface between the radio board and the antenna array is implemented with RF connectors. The SMPM connectors are used at the both ends and positions of those in the antenna module are shown in Figure 6.

5.2 Interface between RF beamformer and RF-control unit

The physical interface connection between Nucleo MCU and RF beamformer or RF board is done with 72-pin header connectors. The control commands are routed with SPI and I2C buses as described in chapter 3.2.2.

From a software point of view the following signal and packets have been defined

- Packet to BF control thread
- Packet to RSS control thread

5.3 Interface between RF beamformer and TRX unit

The communication between RF beamformer and TRX unit is implemented with a sub 6 GHz radio signal. The interface signals directly connected to the RF boards via custom made RF cable shown in Figure 15.

Figure 15. Sub 6 GHz IF cable that connects the TRX unit to two RF boards

Figure 16 shows the measured frequency response of the QMA – SMPM cable.

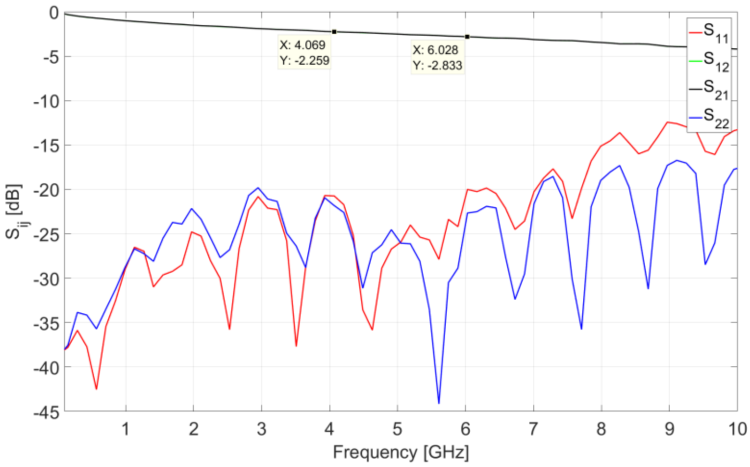


Figure 16. Radio interface cable frequency response.

This measurement result includes an additional loss of about 0.5 dB caused by and SMPM – 2.92 mm adapter for the measurement purposes. We were unable to remove this effect with calibration of vector signal analyser. This means that the total loss of the interface cable is about 1.7 dB at the 4 GHz which is marked in the Figure 16.

5.4 Interface between RF control unit and antenna unit

The TRX unit provides the 230.4 MHz reference clock signal, TXRX signal and Frame start signal for the antenna unit. These are connected to the AUX board that is used to shape the square wave clock signal into a sine wave and 1:3 divide it to the RF boards and another antenna unit, and route and buffer the TXRX and Frame start signals for both the control MCU and Radio boards.

We have measured the reference clock signal from the output of the filter from the first Antenna unit and a daisy-chained antenna unit. The former setup is shown in Figure 17 and the daisy-chained setup is shown in Figure 19. The corresponding measurement results are shown in Figure 18 and Figure 20. A 160 degree phase difference exists between the radio boards due to the square wave – sine wave filter and a 40 degree phase difference exists between the two antenna units.

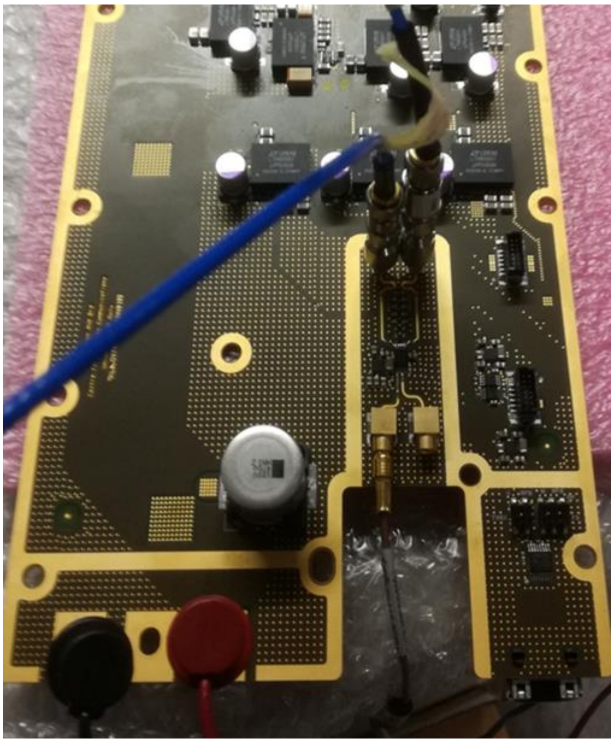


Figure 17. 230 MHz reference clock measurement setup from a single TRX unit.

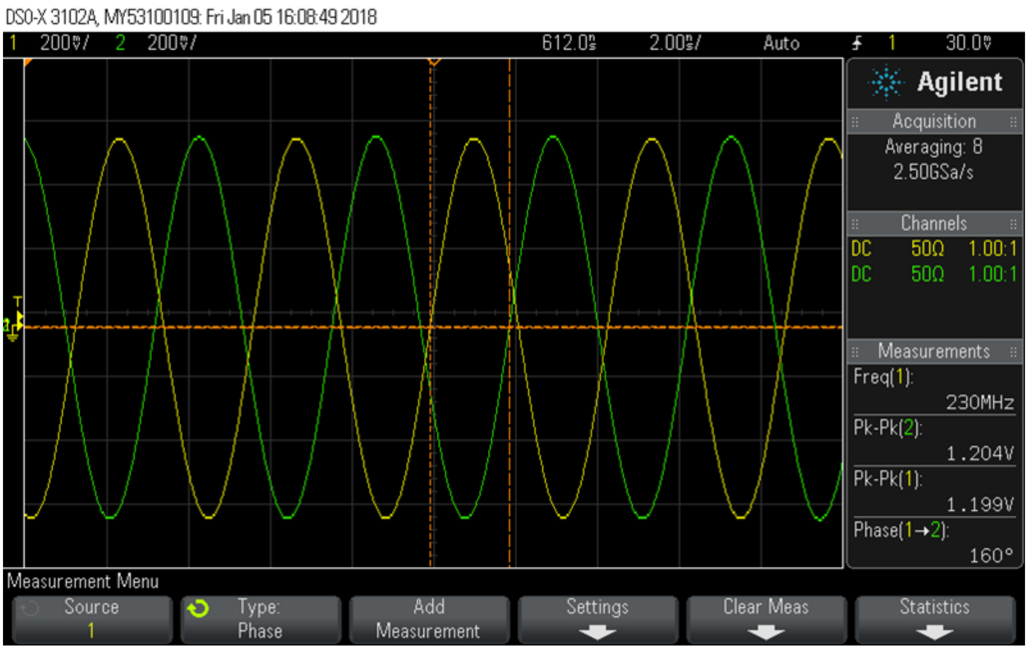


Figure 18. 230 MHz reference clock measured from one TRX unit.

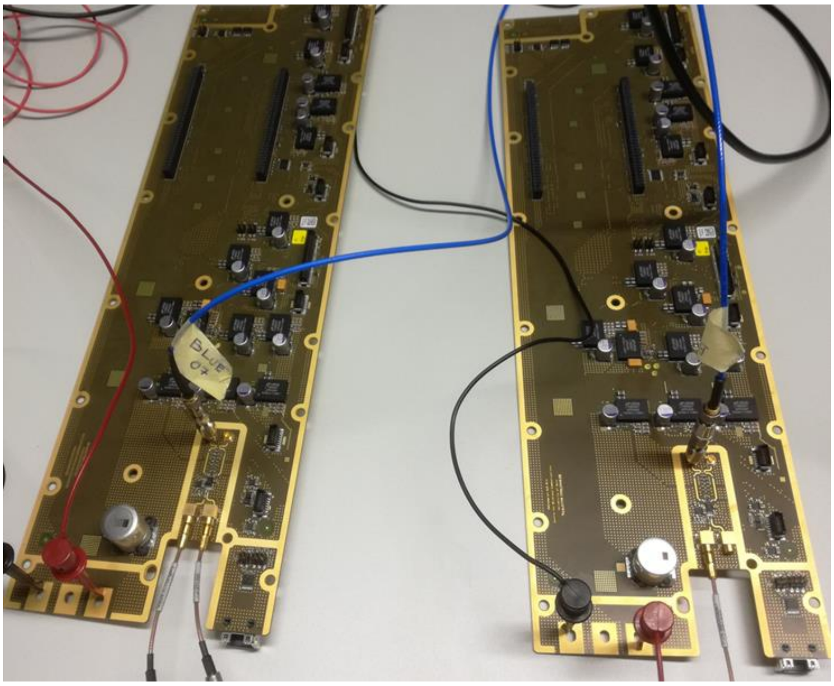


Figure 19. Reference clock measurement setup from two daisy-chained TRX units.

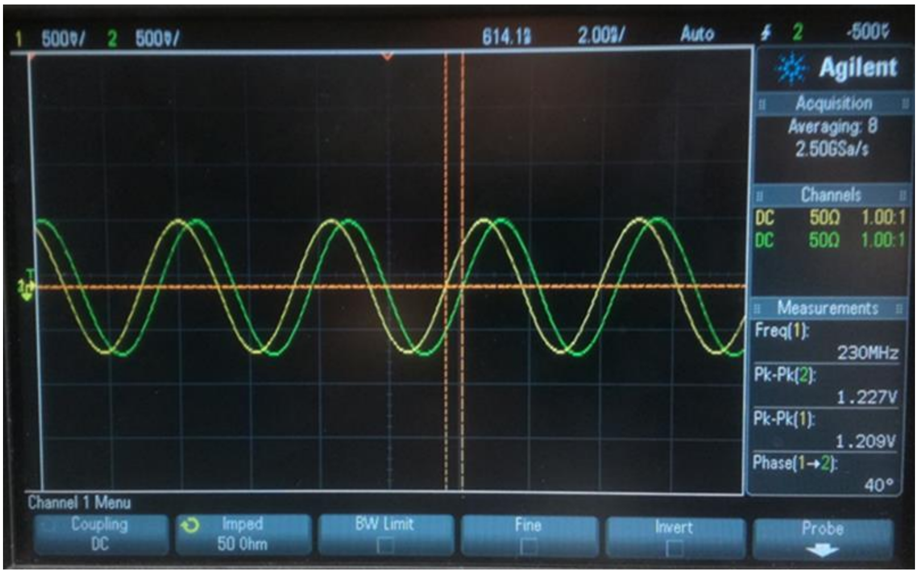


Figure 20. 230 MHz reference clock measured from the first and a daisy-chained TRX unit.

The TXRX and Frame start signals were connected from the TRX unit to the AUX board. The measured waveforms from the Nucleo MCU connector are shown in Figure 21.

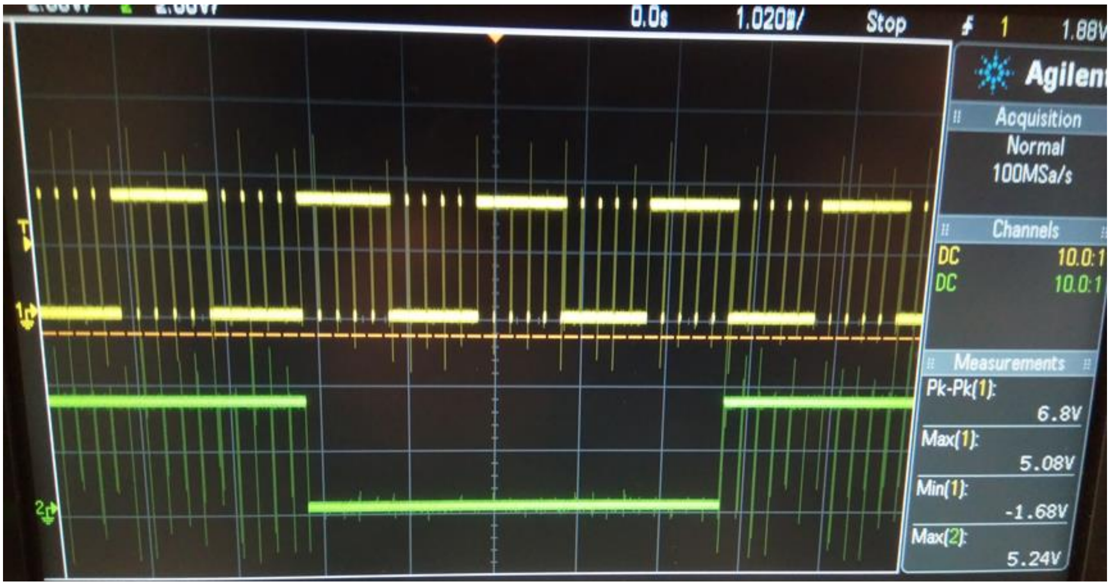


Figure 21. TXRX and Frame start signals measured from the AUX board.

5.5 Interface between GUI control and RF control unit

The interface between the graphical user interface (GUI) (implemented in Matlab) and RF control unit is implemented as Ethernet connection. GUI interface provides user control for following functionalities:

- Select Unit to use via GUI.
- Query SW version from the selected RF control unit, and enable functionalities for GUI accordingly.
- Select Board to interact from the selected unit.
- Manual control for TX/RX mode selection.
- Beamforming control: Type, parameters, rotation.
- Attenuation for TX and RX side.
- Tuning single phase element.
- Select triggering mode (auto = controlled by RF control unit, manual = trigger command send by user when pressing trigger button at GUI).
- Measure received signal strength (RSS) .
- Set frequency (and show corresponding registry values from the RF board).
- Enable or Disable invidual PA/LNA on TX/RX side.
- Set bias voltage values for selected PA component.

GUI illustrates the beam width and direction, maximum gain, phase values for each phase sifter component, and active components reflecting the user selection of BF type and options. Indexing follows the layout on the HW design.

SW functionalities are described more carefully in Section 3.3.2, GUI interface is shown in Figure 22.

The communication between GUI and RF control unit uses standard UDP packets with following structure encapsulated to the UDP data field:

…

Ethernet UDP packet structure:

```
uint8_t mcu_index;      // Antenna unit index
uint8_t board_index;    // RF board index
uint8_t thread_id;      // Thread type id
uint8_t pdu_id;         // PDU data type id
uint8_t pdu_len;        // Length of PDU data
uint8_t data[];         // Encapsulated PDU data
```

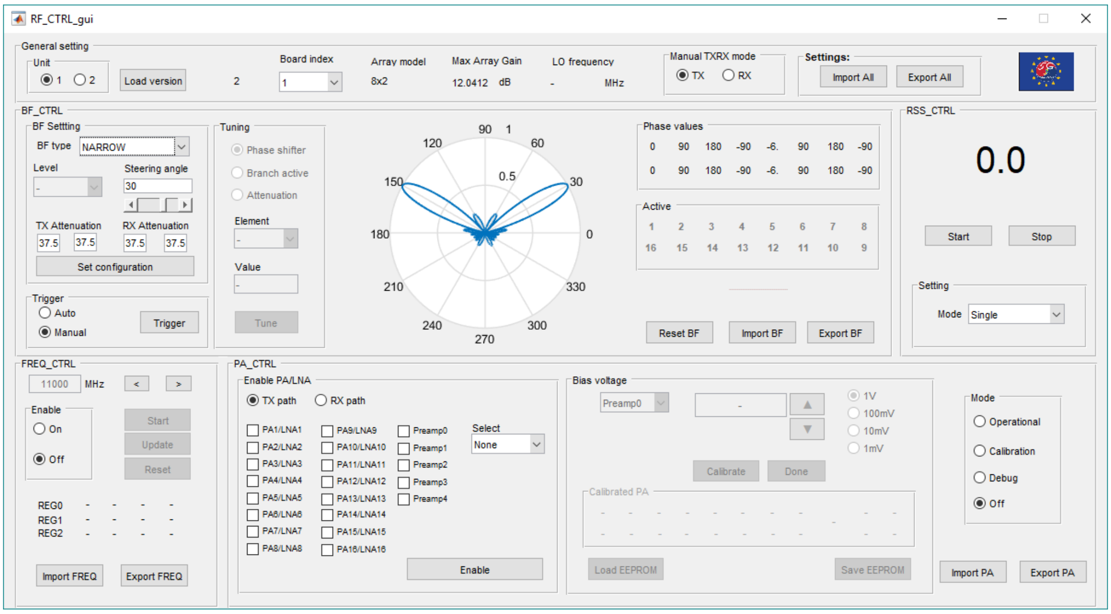


Figure 22. RF control GUI window.

6 Conclusion

In this report we have described the mmWave RF front-end integration and mmW beamforming capability of the implemented 5GCHAMPION wireless backhaul solution.

The interfaces from and to the antenna unit has been validated with measurements which are included into the report. The measurements shows that the two antenna units can be daisy-chained to each other to reduce the cabling between antenna units and TRX unit.

References

[1] H. Kim et al., “A 28GHz CMOS Direct Conversion Transceiver with Packaged Antenna Arrays for 5G Cellular System,” presented at the IEEE Radio Frequency Integrated Circuits Symposium. 2017.

[2] B. Sadhu et al., “A 28GHz 32-Element Phased-Array Transceiver IC with Concurrent Dual Polarized Beams and 1.4 Degree Beam-Steering Resolution for 5G Communication,” presented at the IEEE International Solid-State Circuits Conf. 2017

[3] G. Destino et al, ”Multiple antenna techniques for device pairing of a mmWave high capacity backhaul system,” EUCnC2017, June 12-15, Oulu, Finland, pp.1-4

[4] 5GCHAMPION report “Electronically Reconfigurable Antenna Arrays for Backhauling & Fronthauling (D3.2),” 2017

[5] M. Sonkki et al, “Linearly Polarized 64-element Antenna Array for mm-Wave Mobile Backhaul Application,” EUCAP2018, London 9-13.4.2018, in press

[6] G. Destino “System Analysis and Design of mmW Mobile Backhaul Transceiver at 28GHz”, EUCnC2017, June 12-15, Oulu, Finland, pp.1-5