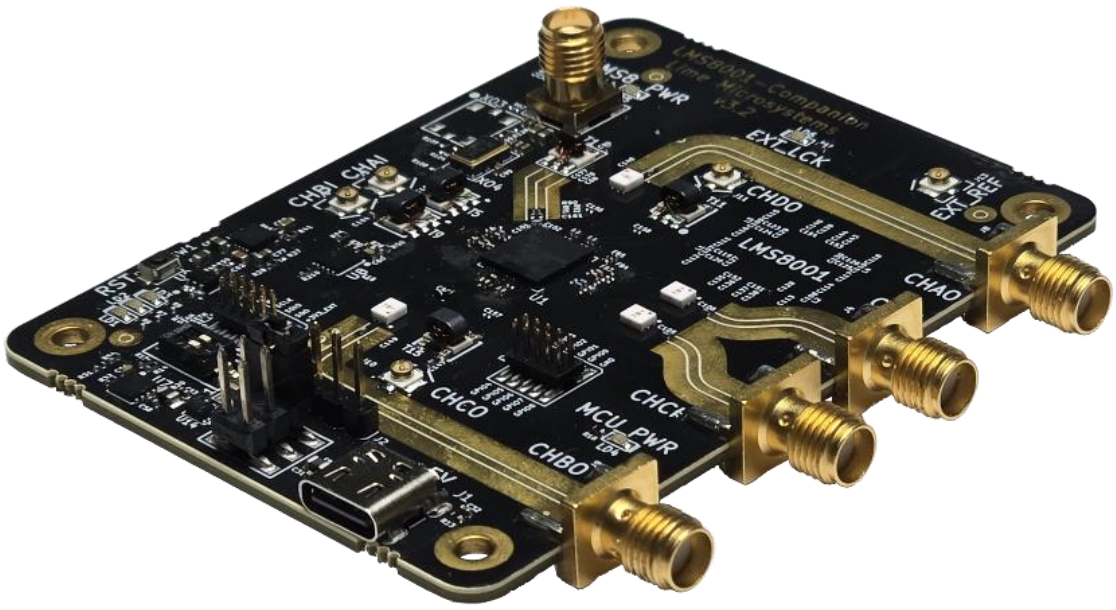
Lime Microsystems Limited

Surrey Technology Centre
Occam Road
The Surrey Research Park
Guildford, Surrey GU2 7YG
United Kingdom

Tel: +44 (0) 1483 685 063
e-mail: enquiries@limemicro.com

LMS8001 Companion Board
Settling Times for Mixer Channel Programs Configuration



Chip version:	LMS8001A
Chip revision:	2
Document version:	2.0
Document revision:	0

DOCUMENT REVISION HISTORY

The following table shows the revision history of this document:

Date	Version	Description of Revisions
17/03/2026	1.0.0	Initial version created.
24/03/2026	2.0.0	Section 4 updated: ON to OFF settling time measured. Section 5 updated: ON to OFF functionality added. Minor text update.

CHIP AND BOARD REVISION HISTORY

The following table shows the current chip and the board version:

Chip/Board	Version /Revision
Chip version	LMS8001A
Chip revision	2
Board version	3
Board revision	2

Contents

1

Overview

This document describes the measurement results of the settling time between two RF mixer channel configuration states. The mixer channel configuration is controlled by external GPIO signals. This is the fastest method that can be used to change the current RF mixer configuration.

Chapter 2**, Hardware Description**, explains hardware configuration related to the RF channel mixer, LMS8001A.

Chapter 3, **Software setup**, outlines the board setup in GUI for LMS8001A.

Chapter 4, **Testing the board**, provides a description of the test bench and includes screenshots from the spectrum analyser taken during different mixer channel configurations.

Chapter 5, **Appendix**, provides the C-codes for the RP2040 that were used in the test bench for the microcontroller board, which generates the external signals connected to the GPIO pins of the Companion Board and external trigger for the spectrum analyser.

2

Hardware Description

2.1 RF mixer channel configuration - LMS8001A

A structure of LMS8001A channel is given in Figure 2:1.

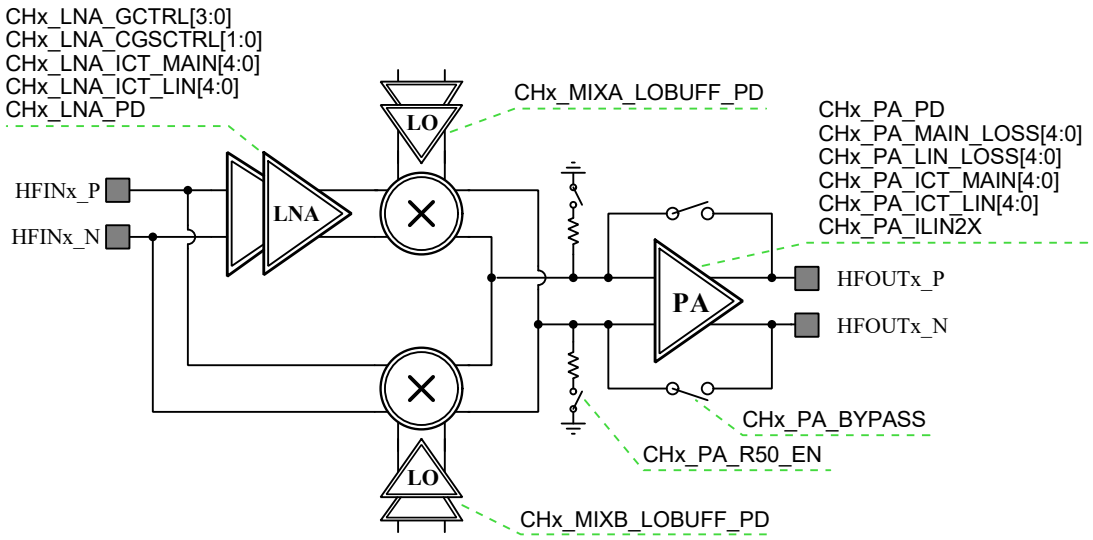


Figure 2:1: RF Channel - LMS8001A

Block diagram which depicts control signals of mixer channel digital circuit is given in Figure 2:2. Control signals are divided into three groups, each with four sets of values, which results in total of 64 possible configurations. Signal group multiplexer control signals are generated by MUXSEL macro, allowing the control via GPIO pins, SPI registers, or combination of them. Registers relevant to RF Channel configuration are grouped into register banks Channel_x (x=A,B,C,D).

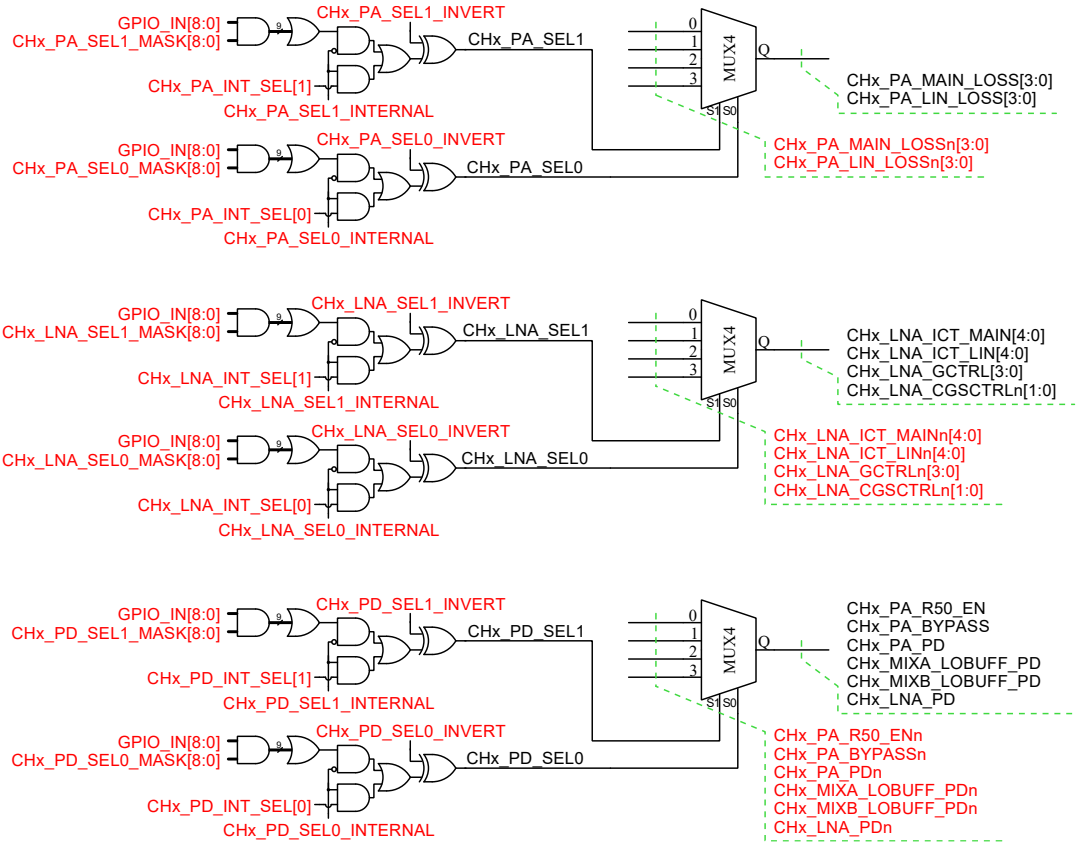


Figure 2:2: RF Channel control signal multiplexing

3

Software setup

The software setup consists of several steps, which are described below. This chapter describes the setup of an LMS8001 mixer channel.

Since GPIO signals will be used, it is necessary to connect the appropriate external signals to the P2 connector, shown in Figure 3:1.

Channel A is used for the measurement, therefore, RF connector J5 should be connected to the signal generator and J8 should be connected to the spectrum analyser.

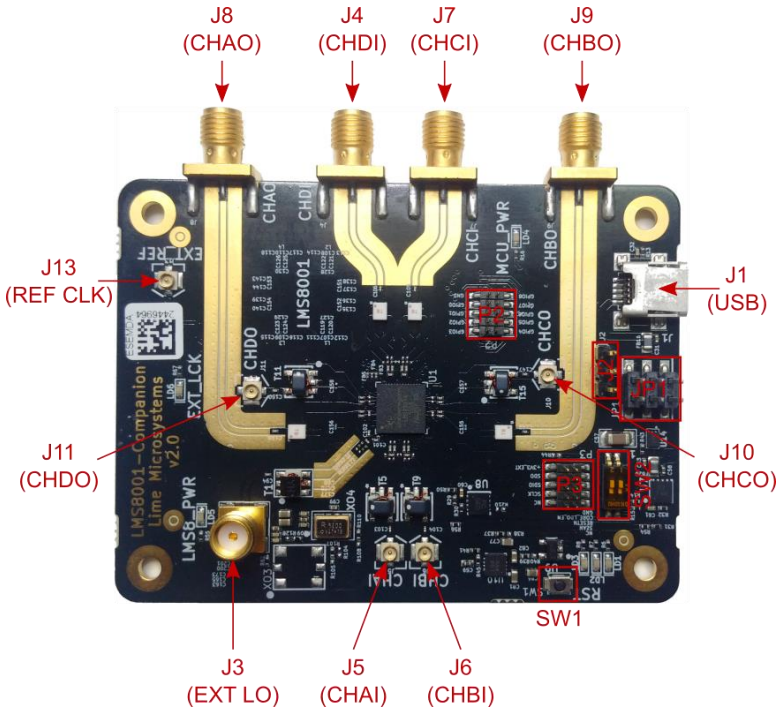


Figure 3:1: LMS8001 Companion Board RF connectors

3.1 Setting RF mixer channel programs for LMS8001A

Most of tests described in Sections 4.1 and 4.2 will use only the change in PA state inside the RF mixer channel (from OFF to ON state).

However, the last part given in Section 4.3 will use three tests:

1. PA from OFF to ON state
2. LNA from OFF to ON state
3. LNA and PA from OFF to ON state

For each of these tests, two different configurations for RF Mixer Channel A will be defined (OFF and ON configuration). Fast configuration selection is achieved using two external signals connected to the P2 connector pins, GPIO0 and GPIO1. The configuration procedure for the LMS8001A chip is described below:

1. Connect the appropriate DC supply via connector J2 and select the correct DC supply source using connector JP1, shown in Figure 3:1.
2. Connect the board to the PC using a USB cable. Refer to the USB Type-C connector J1 shown in Figure 3:1.
3. Connect two external signals to the GPIO0 and GPIO1 pins of connector P2. These signals control the current state of RF Mixer Channel.
   Pay attention to the DC voltage levels of these signals. They must be 3.3 V when the LMS8001 Companion Board is used in the experiment. Also keep in mind that GND must be connected between the LMS8001 Companion Board and external signal generator for GPIO0 and GPIO1 signals. Here in this test the Nano RP2040 Connect Board is used, so 3.3 V is the maximum external signal value, and GND is connected between boards.
4. Start the GUI application, lms8suite. After this step, the startup window should appear.
5. Connect to the board via serial port
   a. Click the *Options* button and select Connection Settings
   b. The *Connection Settings* window will appear. Select the appropriate serial port
   c. Click the *Connect* button
6. Turn off pull-up resistors
   a. Select the *Chip Configuration* tab
   b. Uncheck GPIO0 and GPIO1 boxes for Pull-up
   This step turned out to be necessary for the Nano RP2040 Connect board to drive the GPIO0 and GPIO1 signals on the LMS8001 Companion board.
7. Select the RF Mixer Channel
   a. Select the *LMS8001A – Channel* tab
   b. Select *Channel A*
8. Set the two predefined configurations – Programs 0 and 1
   a. Navigate to Programs 0 and 1
   b. Enable or disable mixer channel subblocks according to the specified configuration, given in Table 1.

Please keep in mind that Programs 2 and 3 are not changed from its default configuration, as well as all values given in Table 2 and Table 3. These values are colored green.

Table 1 – RF Mixer Channel Programs Configurations: Set 1

	LNA_PD	MIXA_PD	MIXB_PD	PA_R50	PA_PD	PA_BYPASS
PA from OFF to ON state						
Program 0	Checked	Checked	Unchecked	Checked	Checked	Checked
Program 1	Checked	Checked	Unchecked	Checked	Unchecked	Unchecked
Program 2	Checked	Checked	Checked	Checked	Checked	Unchecked
Program 3	Checked	Checked	Checked	Checked	Checked	Unchecked
LNA from OFF to ON state						
Program 0	Checked	Checked	Unchecked	Checked	Checked	Checked
Program 1	Unchecked	Unchecked	Checked	Checked	Checked	Checked
Program 2	Checked	Checked	Checked	Checked	Checked	Unchecked
Program 3	Checked	Checked	Checked	Checked	Checked	Unchecked
PA and LNA from OFF to ON state						
Program 0	Checked	Checked	Unchecked	Checked	Checked	Checked
Program 1	Unchecked	Unchecked	Checked	Checked	Unchecked	Unchecked
Program 2	Checked	Checked	Checked	Checked	Checked	Unchecked
Program 3	Checked	Checked	Checked	Checked	Checked	Unchecked

Table 2 – RF Mixer Channel Programs Configurations: Set 2

	LIN ref. current	MAIN ref. current	Cgs additional	Gain Control
Program 0	16	16	2	8
Program 1	16	16	2	8
Program 2	16	16	2	8
Program 3	16	16	2	8

Table 3 – RF Mixer Channel Programs Configurations: Set 3

	LIN gain control	MAIN gain control
Program 0	0	0
Program 1	0	0
Program 2	0	0
Program 3	0	0

9. Configuring the control GPIO signals
    a. Select Bit 0 or Bit 1
    b. Uncheck box INTERNAL
    c. Connect control Bit 0 or Bit 1 to the desired GPIO pins by selecting the corresponding checkboxes in the GPIO Mask.

    The same procedure should be applied to all Bit 0 and Bit 1, in all three sets.

10. Finalizing the setup procedure
    a. Select the *PLL Profiles* tab
    b. Set the LO frequency
    c. Click the *Smart Tune* button
    d. Check if LO frequency is successfully generated
    e. Connect the LO signal to Channel A and verify its operation

After completing all the steps described above, the LMS8001A chip is properly configured. The signal should appear on the spectrum analyser, and its changes should be synchronized with the two external signals connected to the GPIO0 and GPIO1 pins of connector P2.

4

Testing the board

This test was designed to verify fast switching of the mixer channel configuration on the Companion Board. The mixer channel configuration is controlled by two external signals, which are generated by an RP2040 microcontroller board. A detailed list of the hardware devices used in this test is provided below:

1. RF mixer board: The LMS8001 Companion Board 3v2
2. Micro-controller board: Nano RP2040 Connect board
3. Spectrum analyser: Agilent E4440A
4. Signal generator: Agilent E8267D
5. DC supply unit: Rigol DP832
6. Oscilloscope: Rigol MSO2302A
7. Computer: Any PC with lms8sutie

Connectors of the LMS8001 Companion board from the figures below are listed here:

1. J1: USB Type-C connector
2. J2: 5V DC power connector
3. J5: Channel A input
4. J8: Channel A output
5. P2: GPIO connector

The most relevant parameters for the tests below are:

1. Signal generator output power: –10 dBm
2. Signal generator output frequency: 2.1 GHz
3. Spectrum analyser central frequency: 10.0 GHz
4. LO frequency, generated with Smart Tune: 7.9 GHz
5. The Companion Board - Channel A: Up-Conversion

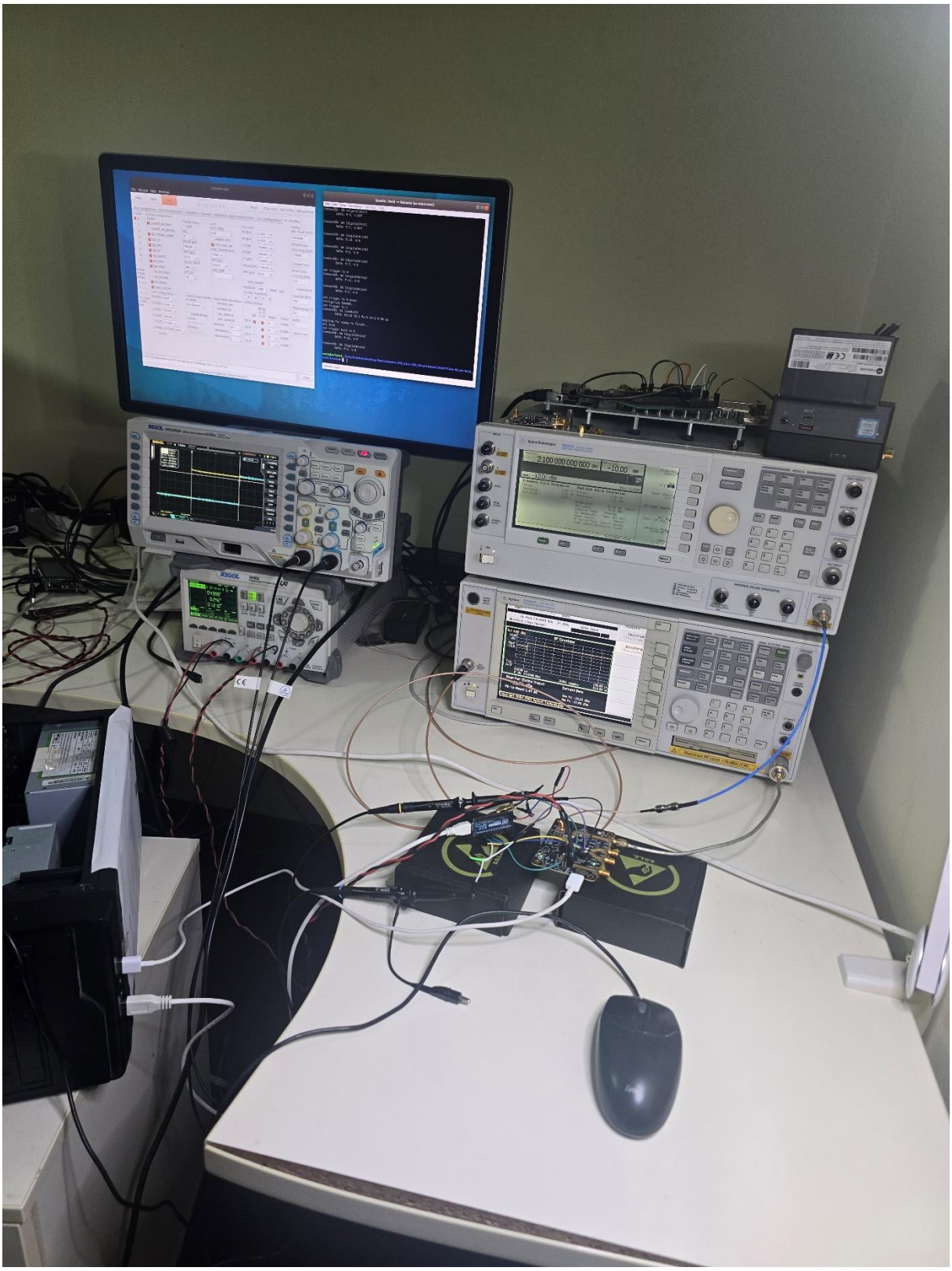


Figure 4:1: Photo of the test bench

4.1 Simple Test measurements

This test should be performed only to confirm that both the LMS8001 Companion board and the Nano RP2040 Connect board are properly configured. It is necessary to upload the C-code provided in Section 5.1. This program is configured to toggle the GPIO0 state every 3 seconds, while GPIO1 remains at 0. In this way, GPIO0 switches the RF mixer channel configuration between Program 0 and Program 1. The switching interval is long enough for both output signals to be easily observed on an Agilent E4440A spectrum analyser. The spectrum analyser is set to "Spectrum Analysis" mode, which is its default mode.

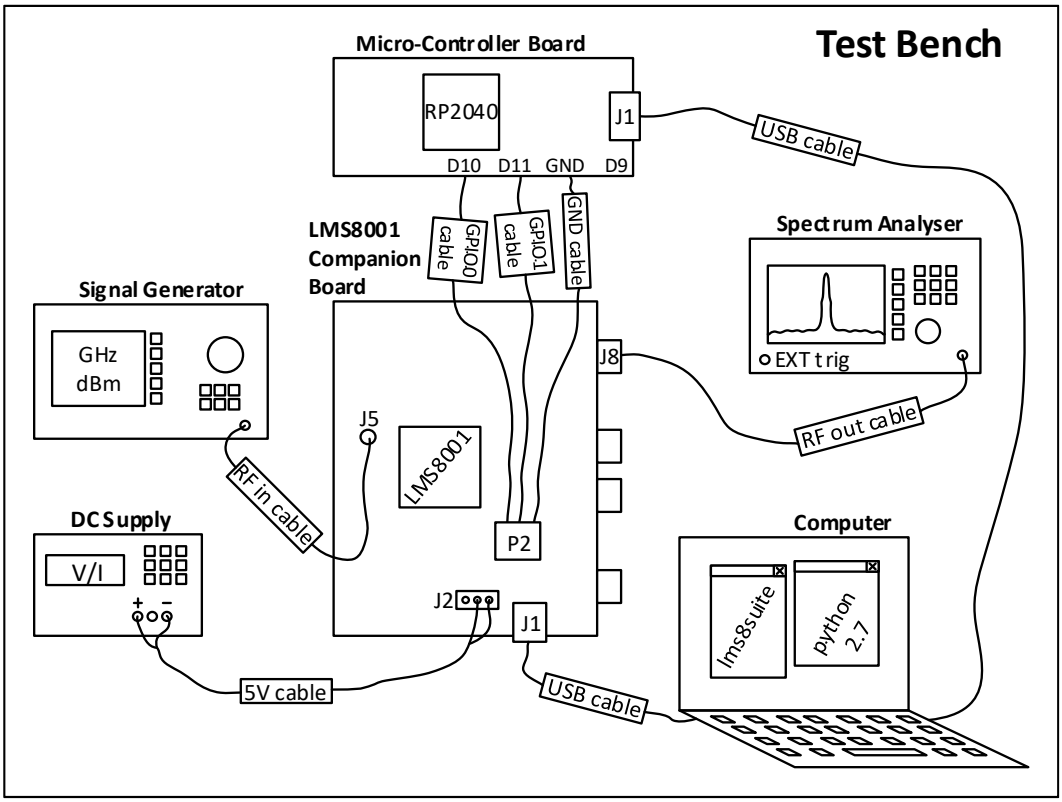


Figure 4:2: Test bench – Simple test

Hardware settings are following:

1. Signal generator is set to generate –10 dBm at 2.1 GHz
2. Spectrum analyser is manually set with the following parameters:
    a. Mode: "Spectrum Analysis" mode
    b. Central frequency: 10.0 GHz
    c. Span: 100 MHz
3. LMS8001 Companion board is set according to Chapter 3, where RF mixer channel Program 0 and Program 1 are given in Table 4 bellow.
4. Nano RP2040 Connect board is set by uploading the program from Section 5.1.

Table 4 – RF Mixer Channel Programs Configurations: Set 1

	LNA_PD	MIXA_PD	MIXB_PD	PA_R50	PA_PD	PA_BYPASS
PA from OFF to ON state						
Program 0	Checked	Checked	Unchecked	Checked	Checked	Checked
Program 1	Checked	Checked	Unchecked	Checked	Unchecked	Unchecked

Results are shown in Figure 4:3 and Figure 4:4 below.

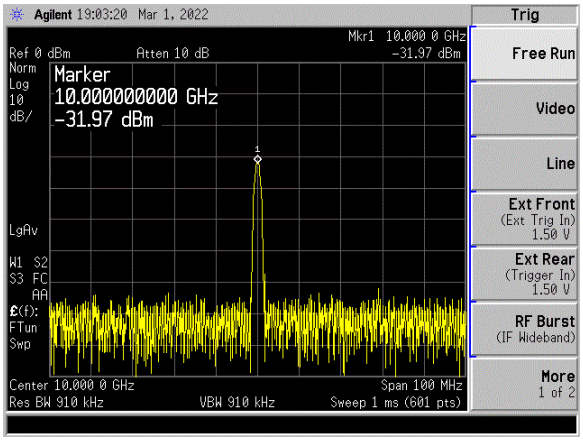


Figure 4:3: Spectrum of RF signal at 10.0 GHz when PA is OFF and BYPASSED

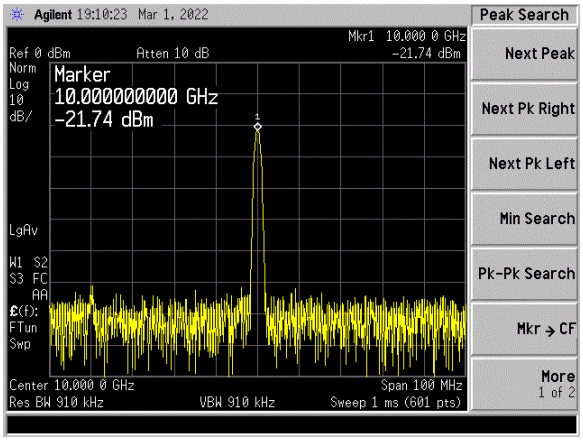


Figure 4:4: Spectrum of RF signal at 10.0 GHz when PA is ON

4.2 RF Envelope Mode measurement

For this measurement, the spectrum analyser is set to “Basic” mode. The purpose is to capture the envelope of the RF signal at 10.0 GHz. The envelope should follow the external control signal applied to the GPIO0 pin. This is a qualitative test corresponding to the procedure described in Section 4.1 and is intended to demonstrate that very fast switching between configurations can be achieved with the LMS8001 Companion board. In this test, the RP2040 C-code provided in Section 5.2 is configured for two cases: toggling the GPIO0 pin at 20 KHz and 1 MHz. For this purpose, Program 0 and Program 1 are set according to Table 4, with only the PA being switched ON and OFF.

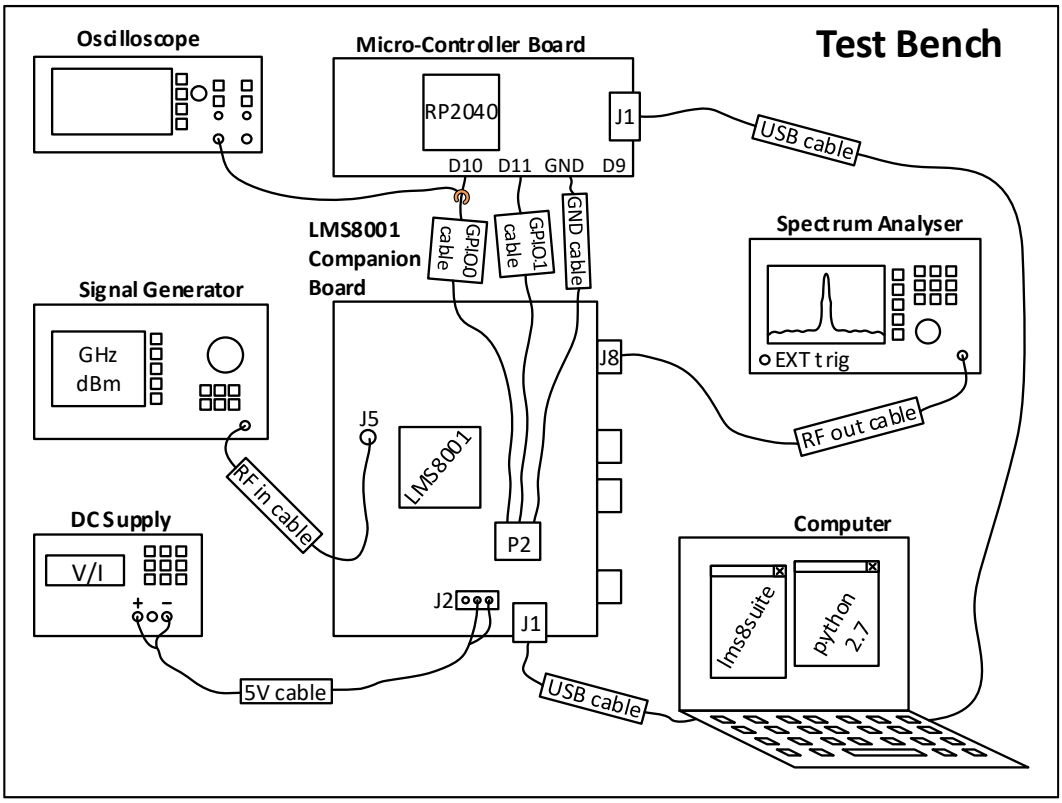


Figure 4:5: Test bench – Envelope Mode test

Hardware settings are as follows:

1. Signal generator set to generate –10 dBm at 2.1 GHz.
2. Spectrum analyzer is manually set with the following parameters:
   a. Mode: “Basic” mode
   b. Central frequency: 10.0 GHz.
   c. Measure: Waveform
   d. Measure Setup / Meas Time:
   500, 10, 50 and 1 us (Figures Figure 4:8, Figure 4:9, Figure 4:10 and Figure 4:11, respectively)
   e. Measure Setup / IF BW: 80 MHz
   (it is recommended to set the maximum value to capture the sharpest edges during transitions)
   f. Measure Setup / Trig Source: Video (IF Envelope)
   g. Trig / Video (IF Envelope) / Level: –40 dBm
   (this value is approximately between the values obtained for the two RF mixer configurations from Program 0 and Program 1, as described in Section 4.1)
   h. The reference level was set to –35 dBm, and the Y amplitude was set to 2 dB/div
   (this was adjusted to enlarge the envelope along the Y-axis on the screen)
3. LMS8001 Companion board is set according to Chapter 3 where RF mixer channel Program 0 and Program 1 are given in Table 4.
4. Nano RP2040 Connect board is set by uploading the program from Section 5.2.
5. The oscilloscope is configured to capture a periodic signal on Channel 1 (yellow). Two cases are measured: moderate switching speed, with the time scale set to 50 us/div and the voltage scale set to 1 V/div; and fast switching speed, with the time

scale set to 500 ns/div and the voltage scale set to 1 V/div. The Channel 1 probe is connected to the GPIO0 signal.

Results are shown in Figures Figure 4:6, Figure 4:7, Figure 4:8, Figure 4:9, Figure 4:10 and Figure 4:11, given below.

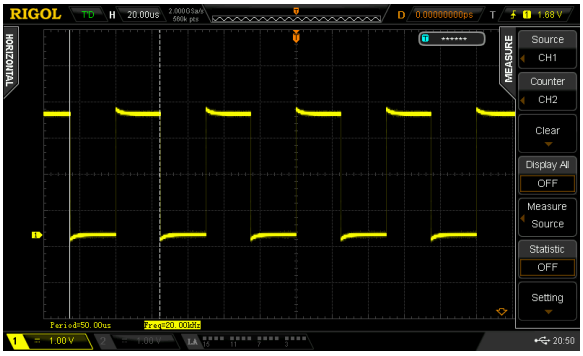


Figure 4:6: Oscilloscope screenshot
GPIO0 control signal frequency 20 KHz

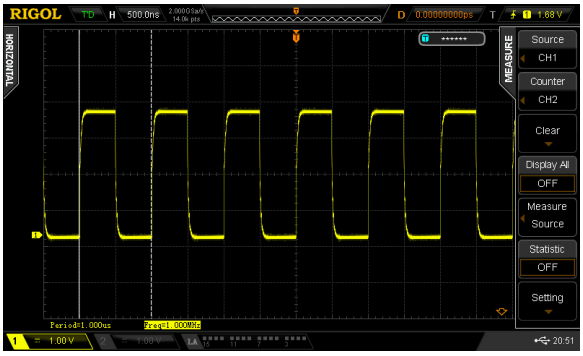


Figure 4:7: Oscilloscope screenshot
GPIO0 control signal frequency 1 MHz

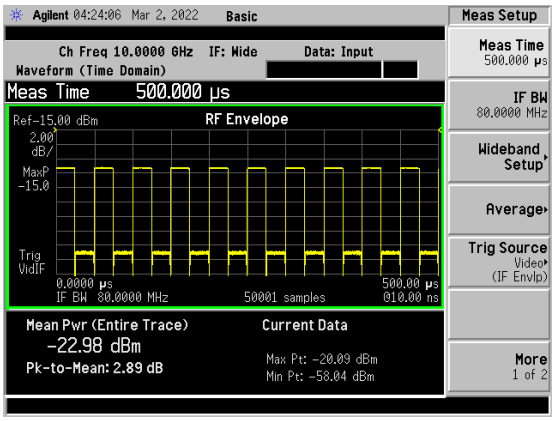


Figure 4:8: Spectrum analyser screenshot
Envelope of the RF signal
frequency 20 KHz and sweep time of 500 us

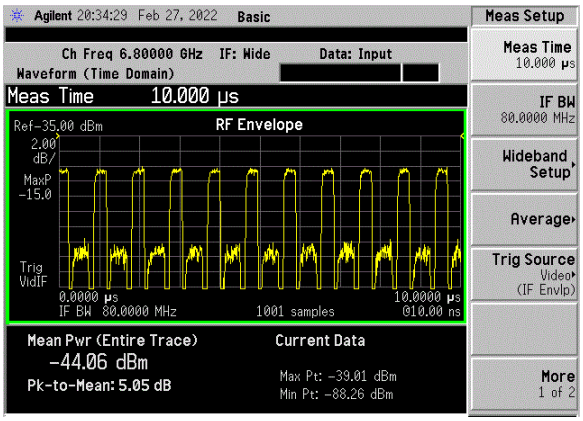


Figure 4:9: Spectrum analyser screenshot
Envelope of the RF signal
frequency 1 MHz and sweep time of 10 us.

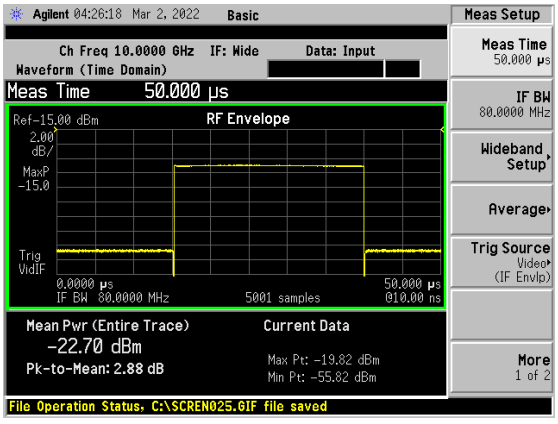


Figure 4:10: Spectrum analyser screenshot
Envelope of the RF signal
frequency 20 KHz, sweep time 50 us

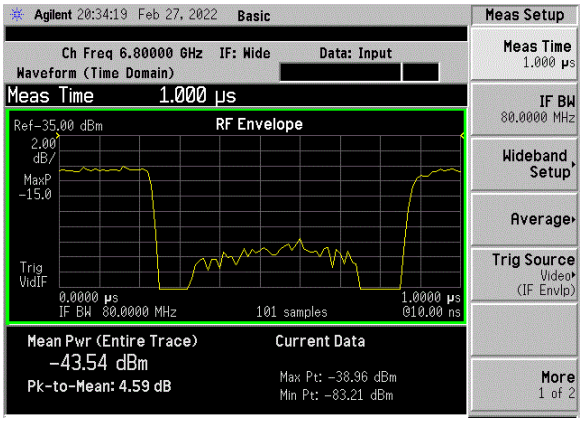


Figure 4:11: Spectrum analyser screenshot
Envelope of the RF signal
frequency 1 MHz, sweep time 1 us

4.3 IQ Mode measurement

For this measurement, the spectrum analyser is set to its “Basic” mode. Its purpose is to capture the baseband I and Q samples of the RF signal at 10.0 GHz. The RF signal changes according to the external signal connected to the GPIO0 pin. This is the most accurate test, as the transient time can be captured and precisely measured. In this test, the RP2040 C-code provided in Section 5.3 is configured to listen on its serial port. A Python script was written to control the spectrum analyser. This Python script also sends a command to the Nano RP2040 Connect board, after which the board generates the GPIO0, GPIO1, and spectrum analyser trigger signals. The Nano RP2040 Connect board generates a rising edge on GPIO0 and the spectrum analyser trigger signal with an exact time delay specified by the Python script. In this way, the spectrum analyser starts capturing I and Q samples shortly before the GPIO0 signal changes the RF mixer channel configuration.

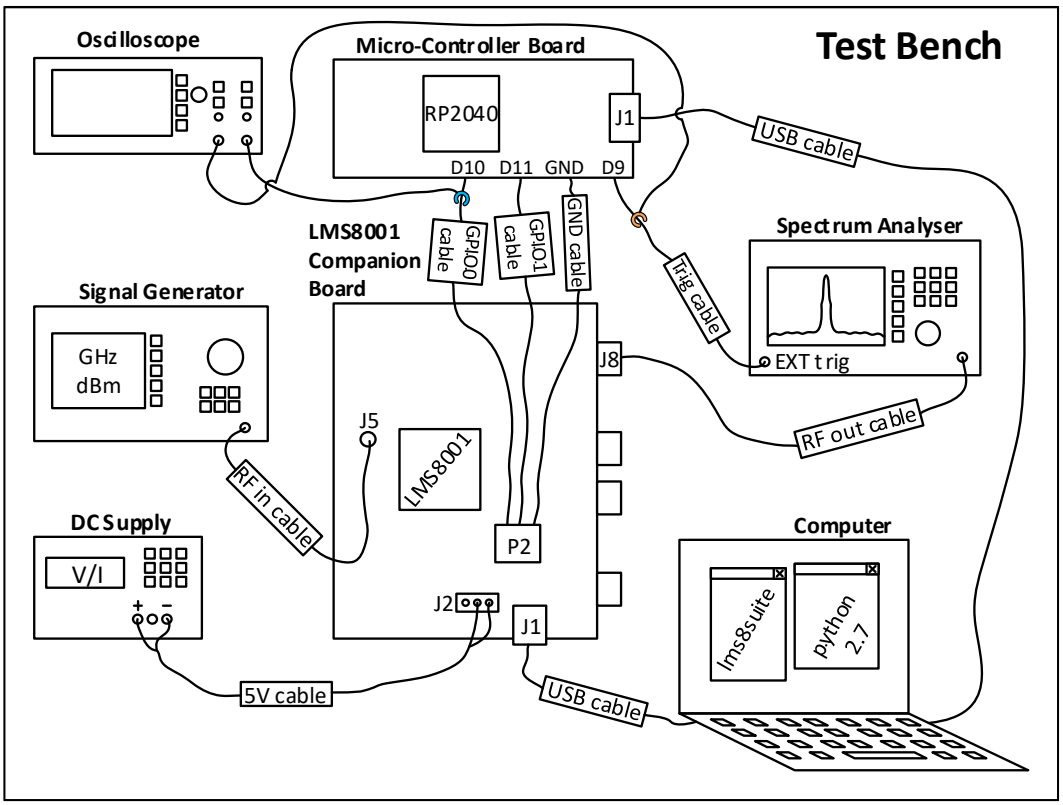


Figure 4:12: Test bench – IQ Mode test

Three tests were performed:

1. PA settling time only.
2. LNA settling time only.
3. Combined LNA and PA settling time.

For these tests, Program 0 and Program 1 are configured according to Table 5.

Table 5 – RF Mixer Channel Programs Configurations for settling time tests

	LNA_PD	MIXA_PD	MIXB_PD	PA_R50	PA_PD	PA_BYPASS
PA from OFF to ON state						
Program 0	Checked	Checked	Unchecked	Checked	Checked	Checked
Program 1	Checked	Checked	Unchecked	Checked	Unchecked	Unchecked
LNA from OFF to ON state						
Program 0	Checked	Checked	Unchecked	Checked	Checked	Checked
Program 1	Unchecked	Unchecked	Checked	Checked	Checked	Checked
LNA and PA from OFF to ON state						
Program 0	Checked	Checked	Unchecked	Checked	Checked	Checked
Program 1	Unchecked	Unchecked	Checked	Checked	Unchecked	Unchecked

Hardware settings are as follows:

1. Signal generator is set to generate –10 dBm at 2.1 GHz.
2. Spectrum analyzer is set using the Python script, with the following parameters:
    a. Mode: “Basic” mode
    b. Central frequency: 10.0 GHz.
    c. Sweep time: 250 us
    d. Sample rate: 100 MHz
    e. Trigger source: External trigger
    f. Trigger slope:
        i. Positive slope for measuring the OFF to ON settling time
        ii. Negative slope for measuring the ON to OFF settling time
    g. Trigger level: 1.6 V
    h. Measure: Waveform
    i. IF Path: Wide
3. LMS8001 Companion board is set according to Chapter 3, where RF mixer channel Program 0 and Program 1 are given in Table 5.
4. Nano RP2040 Connect board is set by uploading the program from Section 5.3.
5. The oscilloscope is configured to trigger on the rising edge when measuring OFF-to-ON settling time (the falling edge is used for ON-to-OFF settling time). The trigger signal is connected to Channel 1. The time scale is set to 5 us/div, and the voltage scale is set to 1 V/div. The Channel 1 probe (yellow) is connected to the spectrum analyzer trigger signal (EXT trigger), while the Channel 2 probe (blue) is connected to the GPIO0 control signal.

Results for measuring PA settling times are shown in the Figures Figure 4:13, Figure 4:14, Figure 4:15, Figure 4:16, Figure 4:17 and Figure 4:18, given below.

Results for measuring LNA settling times are shown in the Figures Figure 4:19, Figure 4:20, Figure 4:21, Figure 4:22, Figure 4:23 and Figure 4:24, given below.

Results for measuring LNA and PA settling times are shown in the Figures Figure 4:25, Figure 4:26, Figure 4:27, Figure 4:28, Figure 4:29 and Figure 4:30, given below.

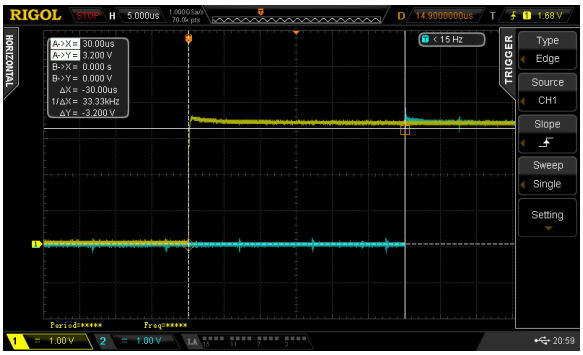


Figure 4:13: Oscilloscope screenshot
EXT trigger (yellow) and GPIO0 (blue) control signals
Rise time measurement

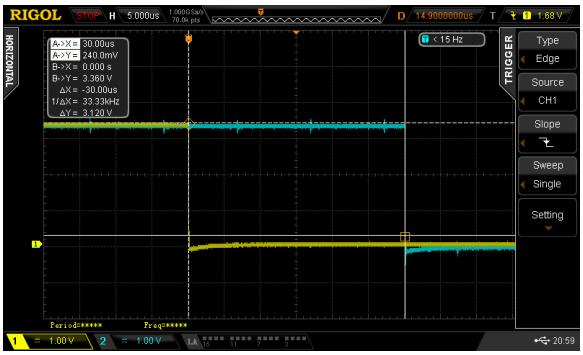


Figure 4:14: Oscilloscope screenshot
EXT trigger (yellow) and GPIO0 (blue) control signals
Fall time measurement

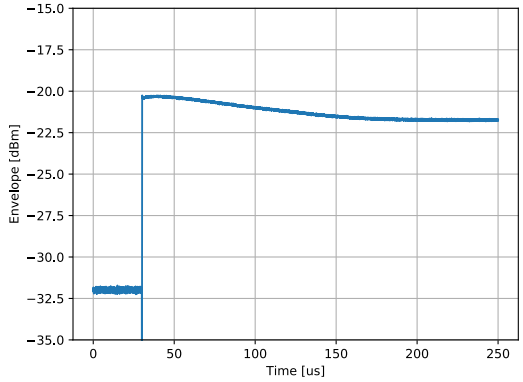


Figure 4:15: PA from OFF to ON
sweep time 250 us

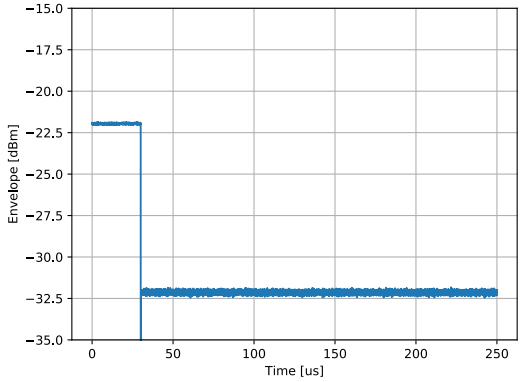


Figure 4:16: PA from ON to OFF
sweep time 250 us

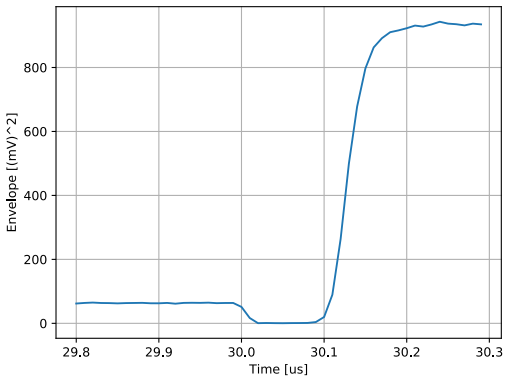


Figure 4:17: PA from OFF to ON
zoomed sweep time

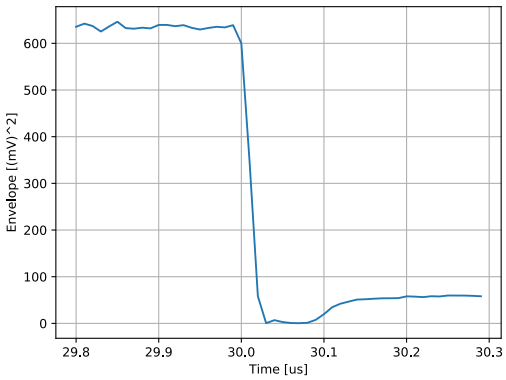


Figure 4:18: PA from ON to OFF
zoomed sweep time

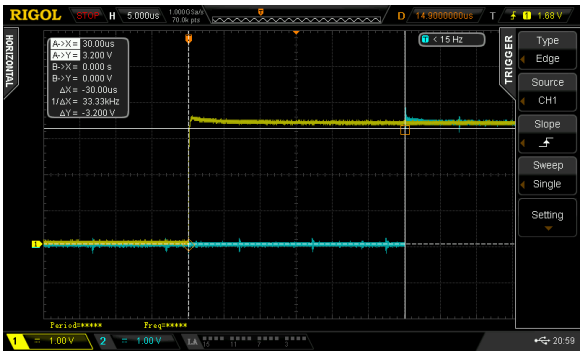


Figure 4:19: Oscilloscope screenshot
EXT trigger (yellow) and GPIO0 (blue) control signals
Rise time measurement

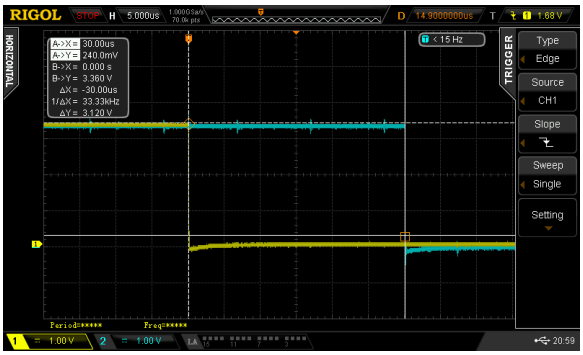


Figure 4:20: Oscilloscope screenshot
EXT trigger (yellow) and GPIO0 (blue) control signals
Fall time measurement

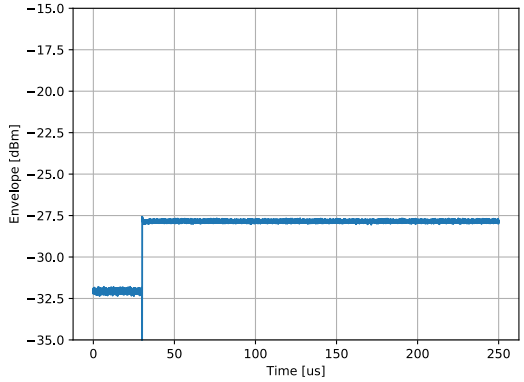


Figure 4:21: LNA from OFF to ON
sweep time 250 us

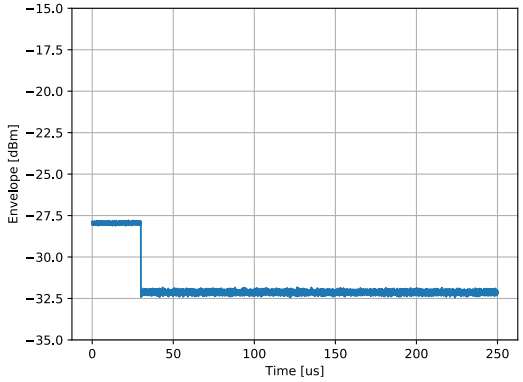


Figure 4:22: LNA from ON to OFF
sweep time 250 us

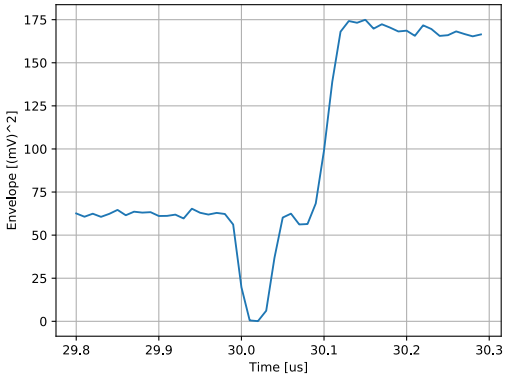


Figure 4:23: LNA from OFF to ON
zoomed sweep time

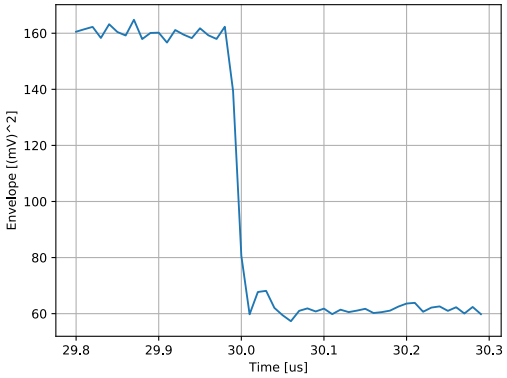


Figure 4:24: LNA from ON to OFF
zoomed sweep time

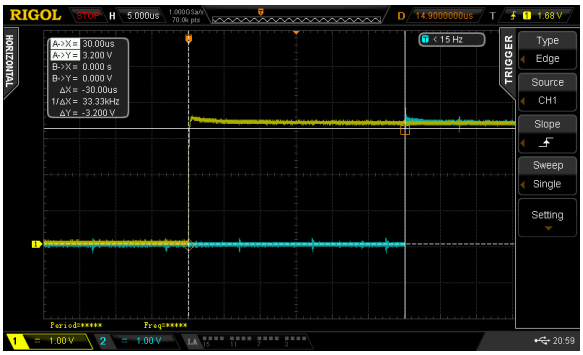


Figure 4:25: Oscilloscope screenshot
EXT trigger (yellow) and GPIO0 (blue) control signals
Rise time measurement

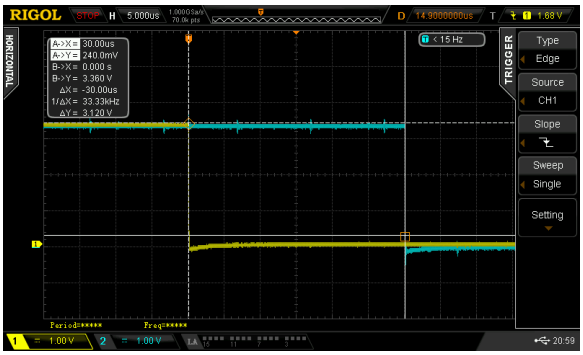


Figure 4:26: Oscilloscope screenshot
EXT trigger (yellow) and GPIO0 (blue) control signals
Fall time measurement

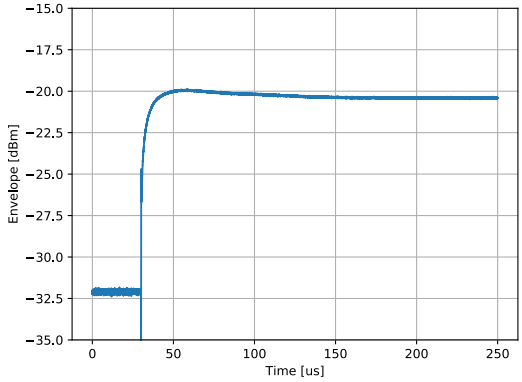


Figure 4:27: PA and LNA from OFF to ON
sweep time 250 us

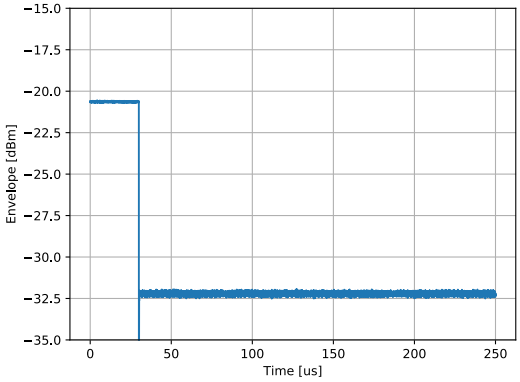


Figure 4:28: PA and LNA from ON to OFF
sweep time 250 us

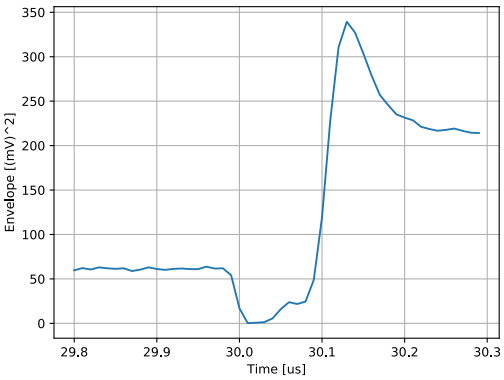


Figure 4:29: PA and LNA from OFF to ON
zoomed sweep time

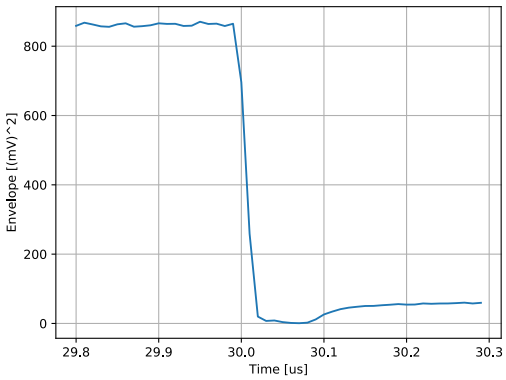


Figure 4:30: PA and LNA from ON to OFF
zoomed sweep time

4.4 Comments

For measurements where the PA is switched ON and OFF, the output signal power is approximately –30 dBm when the PA is OFF and approximately –20 dBm when the PA is ON. These power levels can be observed in all three tests described in Sections 4.1, 4.2 and 4.3.

The RF mixer channel settling time, from OFF to ON state, is in the range of 200 – 300 ns in all cases. These results are obtained from Section 4.3, but also the same results can be clarified with results given in Section 4.2.

The RF mixer channel settling time, from ON to OFF state, is in the range of 150 – 250 ns in all cases. These results are obtained from Section 4.3, but also the same results can be clarified with results given in Section 4.2.

A small overshoot of a fraction of a dB can be observed at the moment the PA is turned on in the RF mixer channel, as shown in Figure 4:15, Figure 4:27 and Figure 4:29. This is expected behavior and is characteristic of the turn-on transient of amplifier circuits. A similar effect is expected in Figure 4:8, Figure 4:10; however, it may not be visible due to the relatively constant operating temperature (resulting from continuous pulse-mode operation) and the relatively large span of the y-axis.

One additional effect can be observed: the gains of the PA (about 10 dB) and LNA (about 2.7 dB) do not sum when both are operating simultaneously (the combined gain is about 10.4 dB). This is expected behavior due to non-ideal matching between these two blocks within the RF mixer channel. Consider that LNA operates at 2.1 GHz, while PA operates at 10 GHz. Also, the LNA by itself does not provide much gain, as it is not optimized for these up-conversion frequencies (IF = 2.1 GHz).

5

Appendix

5.1 RP2040 C-code for the Simple Test measurements

5.1.1 main.c

```
#include "pico/stdlib.h"

#define LMS_GPIO0 5 // Arduino GPIO D10 is RP2040 GPIO 5
#define LMS_GPIO1 7 // Arduino GPIO D11 is RP2040 GPIO 7
//  https://docs.arduino.cc/tutorials/nano-rp2040-connect/rp2040-01-technical-reference/

#define LED_PIN 6
#define toggleTime 3000

int main() {
    // Initialize standard I/O (USB/UART)
    stdio_init_all();

    // Initialize the GPIO pins
    gpio_init(LMS_GPIO0);
    gpio_init(LMS_GPIO1);
    gpio_init(LED_PIN);

    // Set the pin directions to output
    gpio_set_dir(LMS_GPIO0, GPIO_OUT);
    gpio_set_dir(LMS_GPIO1, GPIO_OUT);
    gpio_set_dir(LED_PIN, GPIO_OUT);

    // Set init values
    gpio_put(LMS_GPIO0, 0);
    gpio_put(LMS_GPIO1, 0); // LMS_GPIO1 stays 0
    gpio_put(LED_PIN, 0);

    // Main loop
    while (1) {
        // Set both pins HIGH
        gpio_put(LMS_GPIO0, 1);
        gpio_put(LED_PIN, 1);
        sleep_ms(toggleTime);

        // Set both pins LOW
        gpio_put(LMS_GPIO0, 0);
        gpio_put(LED_PIN, 0);
        sleep_ms(toggleTime);
    }
}
```

5.2 RP2040 C-code for the RF Envelope Mode measurement

5.2.1 gpio.pio

```
.program toggle_pin
; Total loop must be 125 cycles for 1MHz frequency @ 125MHz sys_clk

loop:
    set pins, 1         ; [1] Pin HIGH
    set x, 29           ; [1] Load loop counter
high_delay:
    nop                 ; [1]
    jmp x--, high_delay ; [1] -> (30 iterations * 2 cycles = 60 cycles)

    set pins, 0         ; [1] Pin LOW
    set x, 29           ; [1] Load loop counter
low_delay:
    nop                 ; [1]
    jmp x--, low_delay  ; [1] -> (30 iterations * 2 cycles = 60 cycles)

    jmp loop            ; [1] Jump back to start
```

5.2.2 main.c

```
#include "pico/stdlib.h"
#include "hardware/pio.h"
#include "hardware/clocks.h"
#include "gpio.pio.h"

#define LMS_GPIO0 5 // Arduino GPIO D10 is RP2040 GPIO 5
#define LMS_GPIO1 7 // Arduino GPIO D11 is RP2040 GPIO 7
//  https://docs.arduino.cc/tutorials/nano-rp2040-connect/rp2040-01-technical-reference/

int main() {
    stdio_init_all();

    // Initialize the LMS_GPIO1
    gpio_init(LMS_GPIO1);
    gpio_set_dir(LMS_GPIO1, GPIO_OUT);
    gpio_put(LMS_GPIO1, 0); // LMS_GPIO1 stays 0

    // Use PIO block 0
    PIO pio = pio0;

    // PIO setup
    uint sm = pio_claim_unused_sm(pio, true);
    uint offset = pio_add_program(pio, &toggle_pin_program);
    pio_sm_config c = toggle_pin_program_get_default_config(offset);
    sm_config_set_set_pins(&c, LMS_GPIO0, 1);
    pio_gpio_init(pio, LMS_GPIO0);
    pio_sm_set_consecutive_pindirs(pio, sm, LMS_GPIO0, 1, true);
    sm_config_set_clkdiv(&c, 1.0f); // for fast switching speed of 1 MHz
    // sm_config_set_clkdiv(&c, 50.0f); for moderate switching speed of 20 KHz
    pio_sm_init(pio, sm, offset, &c);

    // Enable the state machine to start execution
    pio_sm_set_enabled(pio, sm, true);

    // Main loop: keep the processor busy while PIO runs independently
    while (1) {
        tight_loop_contents();
    }
}
```

5.3 RP2040 C-code for the IQ Mode measurement

5.3.1 gpio_pin1_fall.pio

```
.program pin1_fall
start:
    pull block      ; TX FIFO → OSR
    mov x, osr      ; OSR → x
delay:
    jmp x-- delay

    set pins, 0     ; PIN LOW
    jmp start
```

5.3.2 gpio_pin1_rise.pio

```
.program pin1_rise
start:
    pull block      ; TX FIFO → OSR
    mov x, osr      ; OSR → x
delay:
    jmp x-- delay

    set pins, 1     ; PIN HIGH
    jmp start
```

5.3.3 gpio_pin2_fall.pio

```
.program pin2_fall
start:
    pull block      ; TX FIFO → OSR
    mov x, osr      ; OSR → x
delay:
    jmp x-- delay

    set pins, 0     ; PIN LOW
    jmp start
```

5.3.4 gpio_pin2_rise.pio

```
.program pin2_rise
start:
    pull block      ; TX FIFO → OSR
    mov x, osr      ; OSR → x
delay:
    jmp x-- delay

    set pins, 1     ; PIN HIGH
    jmp start
```

5.3.5 main.c

```
/*
-------------------------------------------------------------------------------
Pin mapping (relevant pins)

Arduino pin    RP2040 GPIO    LMS8001 GPIO   Spec. Analyser   Oscilloscope
-----------    -----------    ------------   -----------      -----------
D9             GPIO21                        EXT Trigger      Channel 1 (yellow)
D10            GPIO5          LMS_GPIO0                       Channel 2 (blue)
D11            GPIO7          LMS_GPIO1
LED            GPIO6
-------------------------------------------------------------------------------
*/

#include <stdio.h>
#include <string.h>
#include "pico/stdlib.h"
#include "hardware/pio.h"
#include "hardware/clocks.h"
#include "tusb.h"

#include "gpio_pin1_fall.pio.h"
#include "gpio_pin1_rise.pio.h"
#include "gpio_pin2_fall.pio.h"
#include "gpio_pin2_rise.pio.h"

// Mapping to RP2040 GPIO
uint8_t pinNumToSIO_GPIO_OUT_move(uint8_t pin) {
    switch(pin) {
        //  https://docs.arduino.cc/tutorials/nano-rp2040-connect/rp2040-01-technical-
reference/

        // Digital Arduino pins D0-D13
        case 0:  return 0;      // D0 TX
        case 1:  return 1;      // D1 RX
        case 2:  return 25;     // D2  -> RP2040 GPIO 25
        case 3:  return 15;     // D3  -> RP2040 GPIO 15
        case 4:  return 16;     // D4  -> RP2040 GPIO 16
        case 5:  return 17;     // D5  -> RP2040 GPIO 17
        case 6:  return 18;     // D6  -> RP2040 GPIO 18
        case 7:  return 19;     // D7  -> RP2040 GPIO 19
        case 8:  return 20;     // D8  -> RP2040 GPIO 20
        case 9:  return 21;     // D9  -> RP2040 GPIO 21
        case 10: return 5;      // D10 -> RP2040 GPIO 05
        case 11: return 7;      // D11 -> RP2040 GPIO 07
        case 12: return 4;      // D12 -> RP2040 GPIO 04
        case 13: return 6;      // D13 -> RP2040 GPIO 06

        // Analog Arduino pins A0-A5
        case 14: return 26;     // A0 & D14 -> RP2040 GPIO 26
        case 15: return 27;     // A1 & D15 -> RP2040 GPIO 27
        case 16: return 28;     // A2 & D16 -> RP2040 GPIO 28
        case 17: return 29;     // A3 & D17 -> RP2040 GPIO 29
        case 18: return 12;     // A4 & D18 -> RP2040 GPIO 12
        case 19: return 13;     // A5 & D19 -> RP2040 GPIO 13

        //case 20: return       // A6 analog only
        //case 21: return       // A7 analog only

        default: return 0xFF;   // nonvalid pin
    }
}
```

```
#define LMS_GPIO1 7 // Arduino GPIO D11 is RP2040 GPIO 7
#define LED_PIN 6   // Indicator for Serial message is received


// Variable definitions
// pin1 and PIN1 refer to trigger of the spectrum analyser
// pin2 and PIN2 refer to trigger of the LMS8001 Companion board
uint8_t pin1, pin2, val1, val2; // pin1 and pin2 are for RP2040 GPIO mapping
uint8_t PIN1, PIN2;             // PIN1 and PIN2 are for Arduino GPIO mapping
uint16_t delay_us;

uint offset_pin1_fall, offset_pin1_rise;
uint offset_pin2_fall, offset_pin2_rise;

PIO pio = pio0;
uint sm1 = 0;
uint sm2 = 1;


void gpio_set(){
    // Set initial state: opposite of the target value
    uint8_t init_val1 = (val1 == 1) ? 0 : 1;
    uint8_t init_val2 = (val2 == 1) ? 0 : 1;

    gpio_init(pin1);
    gpio_put(pin1, init_val1);
    gpio_set_dir(pin1, true);

    gpio_init(pin2);
    gpio_put(pin2, init_val2);
    gpio_set_dir(pin2, true);
}
```

```
void pio_set(){
    uint offset1, offset2;
    uint pin1_init_val, pin2_init_val;
    pio_sm_config c1, c2;

    // PIO 1 Logic
    if(val1 == 0){
        offset1 = offset_pin1_fall;
        c1 = pin1_fall_program_get_default_config(offset1);
        pin1_init_val = (1u << pin1);
    } else {
        offset1 = offset_pin1_rise;
        c1 = pin1_rise_program_get_default_config(offset1);
        pin1_init_val = 0;
    }

    // PIO 2 Logic
    if (val2 == 0) {
        offset2 = offset_pin2_fall;
        c2 = pin2_fall_program_get_default_config(offset2);
        pin2_init_val = (1u << pin2);
    } else {
        offset2 = offset_pin2_rise;
        c2 = pin2_rise_program_get_default_config(offset2);
        pin2_init_val = 0;
    }

    // SM1 SETUP
    sm_config_set_set_pins(&c1, pin1, 1);
    sm_config_set_clkdiv(&c1, 125.0f);   // divider=125 sets clk to 1 MHz
    pio_sm_init(pio, sm1, offset1, &c1);
    pio_sm_set_pins_with_mask(pio, sm1, pin1_init_val, (1u << pin1));
    pio_sm_set_pindirs_with_mask(pio, sm1, (1u << pin1), (1u << pin1));
    pio_gpio_init(pio, pin1);
    pio_sm_put_blocking(pio, sm1, 0);

    // SM2 SETUP
    sm_config_set_set_pins(&c2, pin2, 1);
    sm_config_set_clkdiv(&c2, 125.0f);   // divider=125 sets clk to 1 MHz
    pio_sm_init(pio, sm2, offset2, &c2);
    pio_sm_set_pins_with_mask(pio, sm2, pin2_init_val, (1u << pin2));
    pio_sm_set_pindirs_with_mask(pio, sm2, (1u << pin2), (1u << pin2));
    pio_gpio_init(pio, pin2);
    pio_sm_put_blocking(pio, sm2, (uint32_t)delay_us);
}
```

```
int main() {
    stdio_init_all();
    tusb_init();

    // Initialize the LMS_GPIO1 and LED_PIN
    gpio_init(LMS_GPIO1);
    gpio_set_dir(LMS_GPIO1, GPIO_OUT);
    gpio_put(LMS_GPIO1, 0);    // LMS_GPIO1 stays 0
    gpio_init(LED_PIN);
    gpio_set_dir(LED_PIN, true);
    gpio_put(LED_PIN, 0);
    sleep_ms(100);

    // Load PIO programs into instruction memory
    offset_pin1_fall = pio_add_program(pio, &pin1_fall_program);
    offset_pin1_rise = pio_add_program(pio, &pin1_rise_program);
    offset_pin2_fall = pio_add_program(pio, &pin2_fall_program);
    offset_pin2_rise = pio_add_program(pio, &pin2_rise_program);

    uint8_t rxBuf[10];

    while (1) {

        int c = getchar_timeout_us(10000);
        if (c == PICO_ERROR_TIMEOUT) continue;

        uint8_t header = (uint8_t)c;
        rxBuf[0] = header;
        int expected_len = 0;

        // define the length of package
        if (header == 0x89 || header == 0x90) expected_len = 3;
        else if (header == 0x91) expected_len = 7;

        // read the rest if header is OK
        if (expected_len > 0) {
            int i = 1;
            while (i < expected_len) {
                int next_c = getchar_timeout_us(100);
                if (next_c != PICO_ERROR_TIMEOUT) rxBuf[i++] = (uint8_t)next_c;
                tud_task();
            }
        }
```

```
        if (rxBuf[0] == 0x89) {
            PIN1 = rxBuf[1];
            val1 = rxBuf[2];
            uint8_t pinTemp = pinNumToSIO_GPIO_OUT_move(PIN1);
            if (pinTemp == 0xFF) {
                printf("Error: Invalid Arduino Pin!\n");
                stdio_flush();
                rxBuf[0] = 0;
                continue;
            }
            char msgOut[5] = "";
            gpio_init(pinTemp);
            if(val1 == 0){
                gpio_set_dir(pinTemp, GPIO_IN);
                strcpy(msgOut, "IN");
            } else {
                gpio_set_dir(pinTemp, GPIO_OUT);
                strcpy(msgOut, "OUT");
            }
            // print to Serial for debug
            printf("CommandID: 89 [digitalInit]\n\tDATA: P:%d, V:%s \n", pinTemp, msgOut);

            stdio_flush();
            while (getchar_timeout_us(0) != PICO_ERROR_TIMEOUT);
            gpio_xor_mask(1u << LED_PIN);
        }

        else if (rxBuf[0] == 0x90) {
            PIN1 = rxBuf[1];
            val1 = rxBuf[2];
            uint8_t pinTemp = pinNumToSIO_GPIO_OUT_move(PIN1);
            if (pinTemp == 0xFF) {
                printf("Error: Invalid Arduino Pin!\n");
                stdio_flush();
                rxBuf[0] = 0;
                continue;
            }
            gpio_put(pinTemp, val1);

            printf("CommandID: 90 [digitalWrite]\n\tDATA: P:%d, V:%d \n", pinTemp, val1);

            stdio_flush();
            while (getchar_timeout_us(0) != PICO_ERROR_TIMEOUT);
            gpio_xor_mask(1u << LED_PIN);
        }
```

```
        else if (rxBuf[0] == 0x91) {
            PIN1 = rxBuf[1];  // triggers the spectrum analyser
            val1 = rxBuf[2];
            PIN2 = rxBuf[3];  // triggers the LMS8001 Companion board
            val2 = rxBuf[4];
            delay_us = rxBuf[5] | (rxBuf[6] << 8);

            // GPIO remapping from Arduino to RP2040
            pin1 = pinNumToSIO_GPIO_OUT_move(PIN1);
            pin2 = pinNumToSIO_GPIO_OUT_move(PIN2);
            if (pin1 == 0xFF || pin2 == 0xFF) {
                printf("Error: Invalid Arduino Pin!\n");
                rxBuf[0] = 0;
                continue; // skip if PIN remapping is wrong
            }

            gpio_set();
            sleep_ms(100);
            pio_set();
            sleep_ms(500);// delay to keep initial pin1 and pin2 states

            // Sync start both State Machines
            pio_enable_sm_mask_in_sync(pio, (1u << sm1) | (1u << sm2));

            // print to Serial for debug
            printf("CommandID: 91 [setBoth]\n\tDATA: P1:%d V1:%d P2:%d V2:%d D:%d us\n",
pin1, val1, pin2, val2, delay_us);

            stdio_flush();
            while (getchar_timeout_us(0) != PICO_ERROR_TIMEOUT);
            gpio_xor_mask(1u << LED_PIN);
        }

        else {
            printf("Unknown header or bad length: %d\n", header);

            stdio_flush();
            while (getchar_timeout_us(0) != PICO_ERROR_TIMEOUT);
            gpio_xor_mask(1u << LED_PIN);
        }

    }
}
```

5.4 Sketch of Python Code for IQ Mode measurement

```
# Python 2.7

# Function definitions
def digitalInit(pin, val):
    valb = 1 if val == "OUT" else 0
    packet = chr(0x89) + chr(pin) + chr(valb)
    s.write(packet)

    time.sleep(0.1)
    print(s.read_all())


def digitalWrite(pin, val):
    valb = 1 if val else 0
    packet = chr(0x90) + chr(pin) + chr(valb)
    s.write(packet)

    time.sleep(0.1)
    print(s.read_all())


def setBoth(pin1, val1, pin2, val2, delay_us=10):
    val1b = 1 if val1 else 0
    val2b = 1 if val2 else 0

    # delay u 2 bajta
    lsb = int(delay_us) & 0xFF
    msb = (int(delay_us) >> 8) & 0xFF

    packet = chr(0x91) + chr(pin1) + chr(val1b) + chr(pin2) + chr(val2b) + chr(lsb) +
chr(msb)
    s.write(packet)

    time.sleep(1)
    print(s.read_all())


# Open port
s = serial.Serial('/dev/ttyACM2', 115200, timeout=1)
time.sleep(2) # Wait for Pico to stabilize

# CRITICAL: Clear any old data from the buffer
s.flushInput()
s.flushOutput()
```

```
# SET GPIO 09, 10, 10 as output
gpio09 = 9     # SA trigger pin
gpio10 = 10    # GPIO0 LMS8001 Companion trigger pin, delayed trigger pin
gpio11 = 11    # GPIO1 LMS8001 Companion pin

digitalInit(gpio09, "OUT")
digitalInit(gpio10, "OUT")
digitalInit(gpio11, "OUT")

triggerPIN = gpio09
delayedTriggerPIN = gpio10

digitalWrite(gpio09, 0)     # digitalWrite(gpio09, 1)
digitalWrite(gpio10, 0)     # digitalWrite(gpio10, 1)
digitalWrite(gpio11, 0)     # digitalWrite(gpio11, 0)
time.sleep(0.5)


print ("set trigger to 0")          # print ("set trigger to 1")
digitalWrite(triggerPIN, 0)         # digitalWrite(triggerPIN, 1)
digitalWrite(delayedTriggerPIN, 0)  # digitalWrite(delayedTriggerPIN, 1)
print ("set trigger to 0 done!")


######################
# Instruments initialization functions...
######################


print ("set trigger to 1")  # print ("set trigger to 0")
delayUS = 30
setBoth(triggerPIN, 1, delayedTriggerPIN, 1, delayUS) # time critical section
# setBoth(triggerPIN, 0, delayedTriggerPIN, 0, delayUS)
print (" triggering finished")


######################
# Instrument capturing IQ samples...
######################


# returning to initial state
digitalWrite(triggerPIN, 0)          # digitalWrite(triggerPIN, 1)
digitalWrite(delayedTriggerPIN, 0)   # digitalWrite(delayedTriggerPIN, 1)

print ("Measuring finished.")
```