

RKX-EVK-001 and Accelerometer Evaluation Board

RoKiX Development Kit User's Guide

The *RoKiX Development Kit* is an easy-to-use hardware platform that allows evaluation of Kionix and ROHM products. The development kit is based on the Cypress *CY8CKIT-059LP Prototyping Kit* featuring an integrated SoC based on ARM® Cortex®-M3 CPU with powerful analog and digital peripherals. The *RoKiX Development Kit* comes with highly configurable *RoKiX Adapter Board A3* that provides easy to use hardware interface between the MCU and the variety of Digital and Analog Kionix and ROHM devices in a plug-and-play fashion. Finally, the *RoKiX Windows GUI*, a powerful Windows-based desktop application, provides an intuitive graphical user interface to demonstrating high level device offerings and features such as visual display of real-time device, data, ability to record device data, and device register editor to name a few.

Definitions

<i>RoKiX IoT Platform Client SW</i>	Complete offering of SW for device evaluation purposes consisting of <i>RoKiX Windows GUI</i> , <i>RoKiX Python CLI</i> , <i>RoKiX Android App</i> .
<i>RoKiX Development Kit</i>	Complete offering of the software, hardware, and firmware used for device evaluation purposes consisting of <i>RoKiX Windows GUI</i> , <i>RoKiX Python CLI</i> , <i>RoKiX Adapter Board A3</i> , and <i>CY8CKIT-059 Host Adapter Board</i>
<i>RoKiX Adapter Board A3</i>	Board specifically designed to easily interface with <i>RoKiX Evaluation Board</i> and numerous development platforms
<i>RoKiX Digital Evaluation Board</i>	Evaluation board with a digital accelerometer
<i>RoKiX Firmware</i>	Proprietary firmware running on microcontroller-based host adapters
<i>RoKiX Windows GUI</i>	<i>RoKiX</i> device evaluation software with graphical user interface (GUI) running in Windows OS
<i>RoKiX Python CLI</i>	<i>RoKiX</i> device evaluation software with OS independent Python based command line interface for quickly testing device low level features

Acronyms

RoKiX	Proprietary brand name made of combining <u>ROHM</u> and <u>Kionix</u> brands
GUI	Graphical User Interface
CLI	Command Line Interface
PSoC	Programmable SoC (System on Chip)

Table of Contents

DEFINITIONS	1
ACRONYMS	1
TABLE OF CONTENTS	2
1. DEVELOPMENT KIT OVERVIEW	5
1.1. <i>RoKiX DEVELOPMENT KIT CONTENT</i>	5
1.2. <i>SYSTEM LEVEL BLOCK DIAGRAM</i>	6
1.3. <i>RoKiX ADAPTER BOARD A3</i>	7
1.3.1. <i>RoKiX Adapter Board A3 Detailed Diagram</i>	7
1.3.2. <i>Input / Output Power Configuration</i>	8
1.3.2.1. <i>VDD_SENSOR Select</i>	8
1.3.2.2. <i>IO_VDD Sensor Select and VR2 Voltage Regulator</i>	9
1.3.2.3. <i>External Power Supply Connection</i>	10
1.3.3. <i>VDD_SENSOR Current Measurements</i>	10
1.3.4. <i>IO_VDD Voltage Level Shifters</i>	11
2. INTERFACE WITH EVALUATION BOARDS	12
2.1. <i>PHYSICAL INTERFACING WITH RoKiX DIGITAL EVALUATION BOARD</i>	12
3. INTERFACE WITH HOST PLATFORMS	13
3.1. <i>CYPRESS CY8CKIT-059 PROTOTYPING KIT</i>	14
3.1.1. <i>Overview</i>	14
3.1.1.1. <i>Featuring PSoC 5LP</i>	15
3.1.1.2. <i>Design for Flexibility</i>	15
3.1.1.3. <i>Low-Cost Programmer</i>	15
3.1.2. <i>Firmware Pinout</i>	16
3.1.3. <i>USB Driver</i>	16
3.1.3.1. <i>USB Driver Installation Procedure</i>	17
3.1.4. <i>Firmware</i>	22
3.1.4.1. <i>Firmware Update Procedure</i>	22
4. GETTING STARTED WITH ROKIX WINDOWS GUI	25
4.1. <i>INTRODUCTION</i>	25
4.2. <i>SETUP</i>	25
4.2.1. <i>Installation</i>	25
4.2.2. <i>Configuration</i>	25
4.3. <i>USER INTERFACE – MENU BAR</i>	27
4.3.1. <i>File – Menu</i>	27
4.3.2. <i>Data – Menu</i>	27
4.3.2.1. <i>Streaming</i>	27
4.3.2.2. <i>Logging</i>	27
4.3.3. <i>Connection – Menu</i>	27
4.3.4. <i>Registers - Menu</i>	28

4.3.4.1.	Load	28
4.3.4.2.	Register dump	28
4.3.5.	Settings – Menu	29
4.3.5.1.	Auto Connect USB	29
4.3.5.2.	Auto config and register download	29
4.3.5.3.	Automatic streaming	29
4.3.5.4.	COM port	29
4.3.5.5.	Reset connection	29
4.3.5.6.	Paired BLE devices	29
4.3.6.	Stream - Menu	30
4.3.7.	Board – Menu	30
4.3.8.	View – Menu	31
4.3.8.1.	Sub-channel view & Digital Output in sub channel view	31
4.3.8.2.	Register write events	33
4.3.8.3.	Reference line	34
4.3.8.4.	Show wake up pop up window	34
4.3.8.5.	Show all board configurations	34
4.3.8.6.	Show ODR warning pop up window	34
4.3.9.	Help – Menu	35
4.3.9.1.	About RoKiX Windows GUI	35
4.3.9.2.	About Host Adapter Board	35
4.3.9.3.	GitHub Issues Tracker	36
4.4.	USER INTERFACE - TABS	36
4.4.1.	Plotter – Tab	36
4.4.1.1.	Raw data	37
4.4.1.2.	Zooming	37
4.4.1.3.	Pausing	37
4.4.1.4.	Moving	37
4.4.1.5.	Clearing	37
4.4.1.6.	Frequency analysis	38
4.4.1.7.	Advanced Data Path (ADP)	39
4.4.2.	Angle Calibration – Tab	40
4.4.3.	Registers – Tab	45
4.4.3.1.	Stream modify mode	46
4.4.3.2.	Updating register value	47
4.4.3.3.	Register sets	49
4.4.3.4.	Register polling function	52
4.5.	USER INTERFACE - STATUS BAR	53
4.6.	USER INTERFACE - POP-UP WINDOWS	53
4.6.1.	No data pop-up window	53
4.6.2.	Streaming pop-up window	53
4.6.3.	ODR has not reached the target value pop-up window	54
4.6.4.	Wake up pop-up window	54

4.7.	SHORTCUTS.....	56
5.	GETTING STARTED WITH ROKIX PYTHON CLI.....	57
5.1.	INTRODUCTION.....	57
5.2.	INSTALLATION FOR WINDOWS OS	57
5.3.	INSTALLATION FOR LINUX AND OS X.....	58
5.4.	PYTHON SET UP.....	58
5.4.1.	Windows installation.....	58
5.4.2.	Linux installation	59
5.4.3.	OS X installation	59
5.4.4.	Optional installations	59
5.5.	CONFIGURATION.....	60
5.5.1.	Connection to RoKiX IoT Platform HW	60
5.5.2.	Generic settings	61
5.6.	FILE STRUCTURE OF THE EVALUATION KIT.....	63
5.7.	RUNNING TEST APPLICATIONS	64
5.7.1.	Data Logger	65
5.7.2.	Data Logger with Wake-up / Back-to-Sleep Engine Test.....	66
5.7.3.	Additional Test Applications.....	67
5.7.4.	Other provided tools.....	67
5.8.	CHANGING TEST APPLICATION CONFIGURATION.....	69
5.9.	REFERENCE DRIVER IMPLEMENTATION.....	70
6.	TROUBLESHOOTING AND KNOWN ISSUES.....	71
6.1.	ODR ACCURACY AND TIMESTAMPING	71
6.2.	USB COMMUNICATION TROUBLESHOOTING	71
6.3.	RoKiX WINDOWS GUI.....	71
6.4.	RoKiX PYTHON CLI TROUBLESHOOTING	72
6.5.	RoKiX DEVELOPMENT KIT COMMUNICATION ISSUES TROUBLESHOOTING	73
6.5.1.	RoKiX Windows GUI Status Bar "Status: Disconnected".....	73
6.5.2.	RoKiX Windows GUI Status Bar "Status: No data".....	74
6.5.3.	RoKiX Python CLI "Automatic search found no devices".....	75
6.5.4.	RoKiX Python CLI "No data received."	75
6.5.5.	RoKiX Python CLI "Permission Error 'Access is denied.'"	76
6.5.6.	RoKiX Python CLI "EVKIT_ERR_BUS1; sensor-bus operation failed"	76

1. Development Kit Overview

1.1. RoKiX Development Kit Content



Figure 1. RKX-EVK-001 (Left) and RoKiX Digital Evaluation Board purchased separately (Right) shown here together

The RoKiX Development Kit (RKX-EVK-001) comes standard with Cypress PSoC® 5LP Prototyping Kit plugged into the RoKiX Adapter Board A3, one micro-USB cable (3.3' / 1M), one 14-position ribbon cable (1.5' / 457.20mm), Quick Start Guide, and Precautions for use. The RoKiX Development Kit is designed to work seamlessly with the RoKiX Digital Evaluation boards, such as KX132-1211-EVK-001 and KX134-1211-EVK-001 that can be purchased separately.

1.2. System Level Block Diagram

The main components of the *RoKiX Development Kit* are the host platform (Cypress CY8CKIT-059) and the *RoKiX Adapter Board A3*. The *RoKiX Development Kit* is designed to be interfaced seamlessly with the *RoKiX Digital Evaluation Board* that can be purchased separately. The main purpose of the *RoKiX Adapter Board A3* is to provide a hardware interface between the host platform and the evaluation board. The Figure 2 shows the simplified high-level block diagram of the *RoKiX Development Kit*.

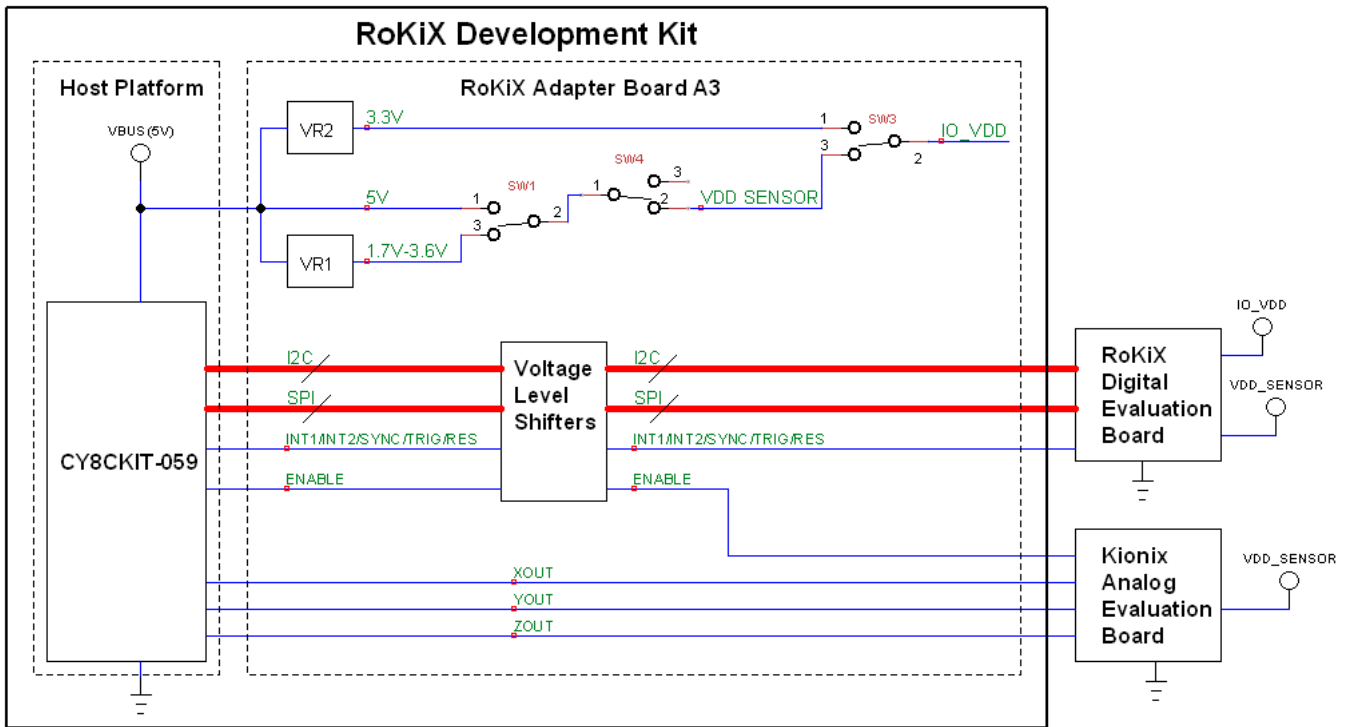


Figure 2. High Level Block Diagram of the *RoKiX Development Kit*

1.3. RoKiX Adapter Board A3

1.3.1. RoKiX Adapter Board A3 Detailed Diagram

The *RoKiX Adapter Board A3* is designed to easily interface with Kionix and ROHM sensors and numerous development platforms. By default, the board is populated to interface with Cypress CY8CKIT-059LP PSoC® prototyping platform and with Kionix standard evaluation boards featuring a 14-pin male header. However, with some hardware modifications, the board can also support additional host platforms such as Arduino™ UNO R3, and Raspberry Pi, and additional evaluation boards like [ROHM Sensor Shield Modules](#) 5-pin digital or 4-pin analog boards. The Figure 3 shows the main component of the *RoKiX Adapter Board A3*.

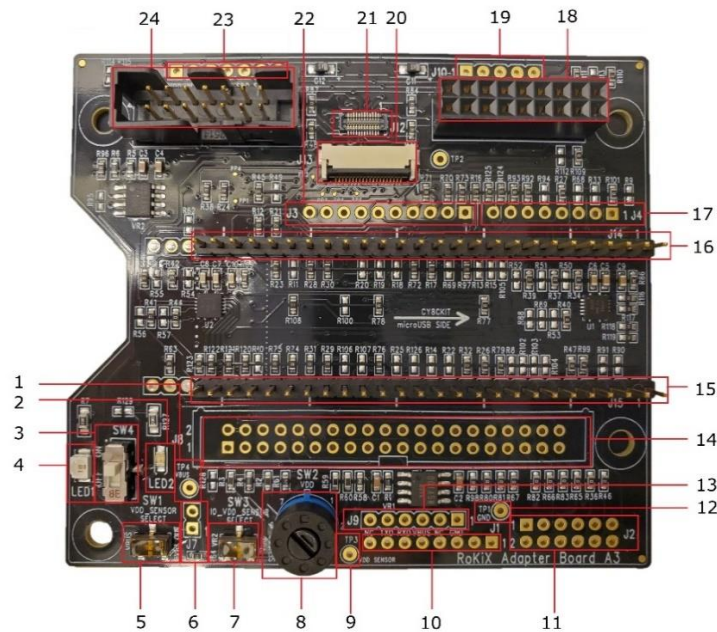


Figure 3. *RoKiX Adapter Board A3* Main Features

- | | |
|--|---|
| 1. TP4 – Test Point 4 for VBUS (Host) input voltage measurement | 13. J9 – Raspberry Pi 6-pin debug header |
| 2. LED2 – Orange LED is ON when VDD_SENSOR voltage is ON | 14. J8 – Raspberry Pi 40-pin dual-row header |
| 3. SW4 – Switch that connects VDD_SENSOR to VBUS/VR1_OUT | 15. J15 – Cypress CY8CKIT-059LP compatible header |
| 4. LED1 – Green LED is ON when VBUS (Host) Voltage is provided | 16. J14 – Cypress CY8CKIT-059LP compatible header |
| 5. SW1 – VDD_SENSOR select switch (VBUS or VR1_OUT) | 17. J4 – Arduino™ UNO R3 Compatible Digital Header (bottom mount) |
| 6. J7 / R64 – VDD_Sensor current measurement header/bypass. | 18. J6 – RoKiX Digital evaluation board compatible header |
| 7. SW3 – IO_VDD Select switch (VDD_SENSOR or VR2_OUT) | 19. J10 – ROHM Sensor Module 5-Pin Digital / 4-Pin Analog Header |
| 8. SW2 – 7-position rotary switch to configure VR1_OUT voltage:
1 = 3.3V, 2 = 3.0V, 3 = 2.8V, 4 = 2.5V, 5 = 1.8V, 6 = 1.7V, 7 = 3.6V | 20. J13 – Kionix KAETH0004R01 Board 20-pin FPC Connector |
| 9. TP3 – Test Point 3 for VDD_SENSOR voltage measurement. | 21. J12 – RoKiX Add-On Board connector |
| 10. J1 – Arduino™ UNO R3 Compatible Power Header (bottom mount) | 22. J3 – Arduino™ UNO R3 Compatible Digital Header (bottom mount) |
| 11. J2 – Arduino™ UNO R3 Compatible Analog Header (bottom mount, even-numbered pins). | 23. J11 – ROHM 7-Pin Digital Sensor (SPI) Header |
| 12. TP1 – Test Point 1 for GND reference voltage measurements | 24. J5 – RoKiX Digital evaluation board ribbon cable compatible header |

1.3.2. Input / Output Power Configuration

1.3.2.1. VDD_SENSOR Select

The *RoKiX Adapter Board A3* gives users the flexibility to test sensors at different VDD and IO_VDD input voltages as well as provides a way to interface sensors with both 5V platforms (e.g., Arduino™ UNO R3, or Cypress CY8CKIT-059) and 3.3V platforms (e.g., Raspberry Pi).

The VDD Sensor Select circuitry is show in Figure 4. When *RoKiX Adapter Board A3* is connected to a host platform, the input voltage to the board is supplied on VBUS net and the green LED (LED1) will be ON. The VBUS voltage is then supplied as an input to the Voltage Regulator (VR1) and is connected the Single Poll Double Throw (SPDT) switch SW1. The purpose of the SW1 is to select the VDD voltage to the sensor that will be connected to the *RoKiX Adapter Board A3* (VDD_SENSOR). One option for VDD_SENSOR is the actual VBUS voltage. The second option is the output voltage from the VR1 voltage regulator (VR1_OUT). The default configuration of the *RoKiX Adapter Board A3* is to select the output voltage from the VR1 voltage regulator.

NOTE: Care must be taken when switch SW1 is moved left to select VBUS voltage. For platforms like CY8CKIT-059 Prototyping Kit and Arduino™ UNO R3, the VBUS voltage can be as high as 5V. Since many Kionix's sensors including KX132-1211, are rated to 3.6V VDD max, the overvoltage and potential permanent damage may be done in cases when 5V VBUS voltage is connected to VDD_SENSOR.

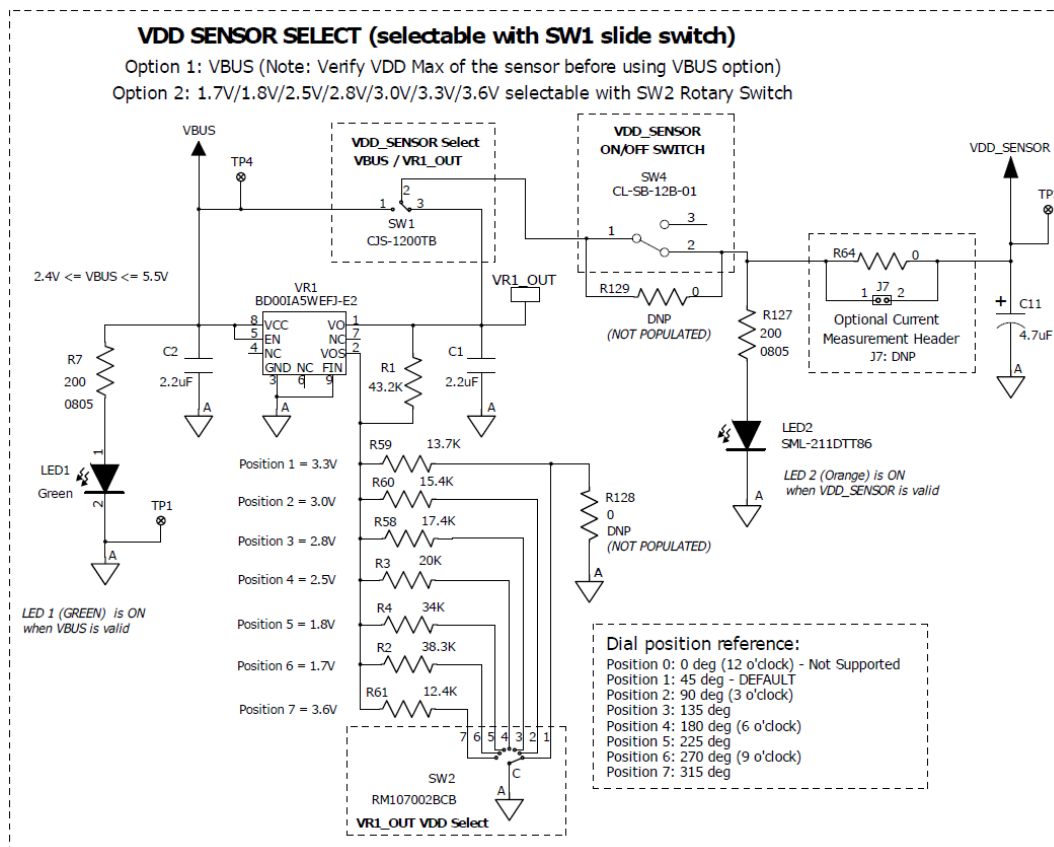


Figure 4. *RoKiX Adapter Board A3* VDD Sensor Select

The VR1 voltage regulator is a variable output Low Dropout (LDO) linear voltage regulator. The output voltage of the VR1 is selected via rotary switch SW2. The SW2 switch has 7 positions as indicated on the schematic and on the printed circuit board itself. The

switch SW2 can be rotated with a small flat screwdriver. By default, the *RoKiX Adapter Board A3* is shipped with switch SW2 in Position 1. See Table 1 for details on how to select the output voltage using SW2.

SW2 position	1	2	3	4	5	6	7
VR1_OUT	3.3V (default)	3.0V	2.8V	2.5V	1.8V	1.7V	3.6V

Table 1. Switch SW2 Voltage Select for different Switch Positions

Following the voltage path from VBUS to the switch SW1 (from left-to-right in Figure 4), next comes switch SW4 (VDD_SENSOR ON/OFF switch). The purpose of the switch SW4 is to disconnect the VDD_SENSOR from the input voltage (VBUS). This can be useful when evaluation board needs to be unplugged and re-plugged again. By default, the switch SW4 is on the ON position (UP position when looked from above). When *RoKiX Adapter Board A3* is connected to a host platform, and the switch SW4 is turned ON the orange LED (LED2) will be ON. If the switch SW1 is connected to the output of the VR1 voltage regulator (VR1_OUT), the brightness of the orange LED will be proportional to the VR1_OUT voltage output. At default position of the switch SW2 (3.3V) the LED light will be bright and when the output voltage is selected to 1.8V or 1.7V using the switch SW2, the LED2 light will be dim. If orange LED (LED2) is completely OFF and green led (LED1) is ON, please ensure the switch SW2 is not turned to an intermediate position and is at one of the 7 positions shown in the Table 1.

1.3.2.2. IO_VDD Sensor Select and VR2 Voltage Regulator

The *RoKiX Adapter Board A3* provides a flexibility for selecting the IO_VDD source for the connected sensors independent of the VDD_SENSOR if desired. As shown in Figure 5, the selection is made using the Single Pole Double Throw (SPDT) switch SW3. The default option for IO_VDD is to be connected to VDD_SENSOR rail. In this case, the IO_VDD voltage will follow the VDD_SENSOR voltage. The alternative selection of the SW3 switch would connect IO_VDD rail to the output of the VR2 voltage regulator.

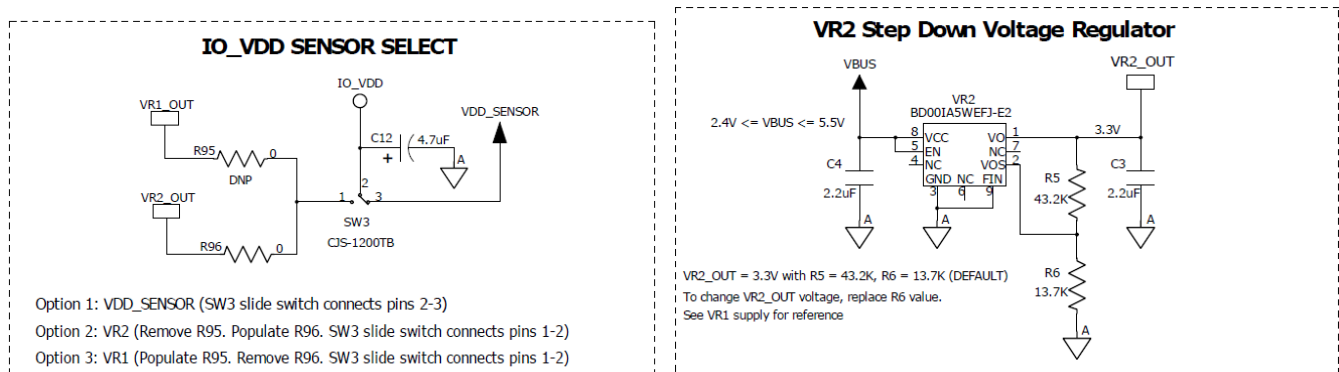


Figure 5. IO_VDD Sensor Select and VR2 Voltage Regulator

The VR2 voltage regulator is also variable output Low Dropout (LDO) linear voltage regulator. By default, it is configured to output 3.3V using the preset values of R5 and R6 feedback resistors. However, the user can modify the output of the VR2 voltage regulator if desired by replacing the R6 resistor (13.7k) with another value (Table 2).

VR2_OUT	3.3V (default)	3.0V	2.8V	2.5V	1.8V	1.7V	3.6V
R6	13.7k	15.4k	17.4k	20k	34k	38.3k	12.4k
R5	43.2k	43.2k	43.2k	43.2k	43.2k	43.2k	43.2k

Table 2. Voltage Regulator (VR2) Output Option

1.3.2.3. External Power Supply Connection

In some cases, it may be required to provide an external voltage source for the VDD_SENSOR. To accomplish this, please remove R64 zero-ohm resistor. The positive terminal of the external power supply can then be attached to the test point TP3 and negative terminal can be attached to any GND location on the *RoKiX Adapter Board A3*. One such convenient location is a test point TP1 show in Figure 6.

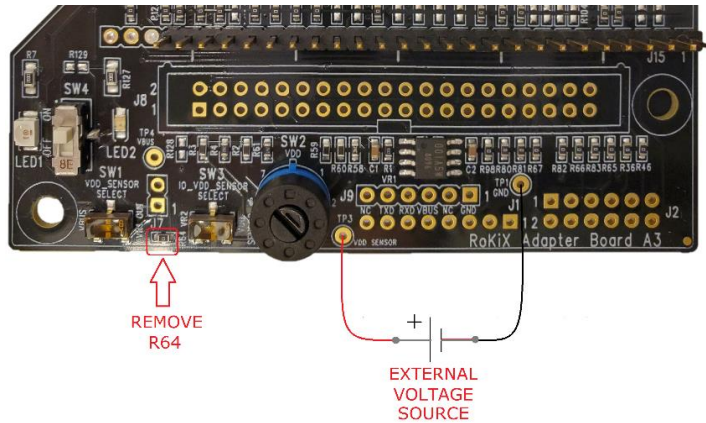


Figure 6. External Voltage Source Recommended Connection

1.3.3. VDD_SENSOR Current Measurements

The *RoKiX Adapter Board A3* provides a convenient way to measure the current supplied on VDD_SENSOR power rail for testing and evaluation purposes. In order to measure the VDD_SENSOR current, it is recommended to remove R64 zero-ohm resistor and to connect a current meter across the J2 header that can be optionally populated for such a test.

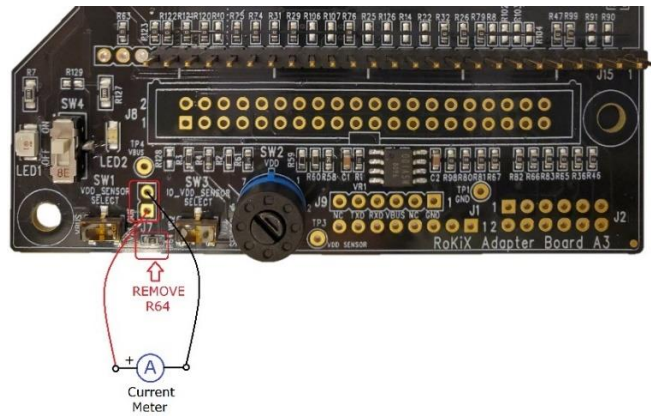


Figure 7. VDD_SENSOR Current Measurement Recommended Connections

1.3.4. IO_VDD Voltage Level Shifters

The *RoKiX Adapter Board A3* comes standard with a pair of voltage level shifters (U1, U2) that are designated to shift the voltage levels of all digital I/O pins from the voltage level supported by the host platform (VBUS) and I/O Voltage provided to a sensor (IO_VDD) and vice-versa (Figure 8). This allows seamless interface between such platforms like Cypress CY8CKIT-059LP and Arduino™ UNO R3, where the I/O voltage can be as high as 5V with many of Kionix sensors that are limited to IO_VDD voltage of 3.6V or lower. Please note the following information regarding the voltage level shifters:

- ✓ The acceptable input voltage range on the host side (B-side) is 2.3V to 5.5V
- ✓ The acceptable input voltage range on the sensor side (A-side) is 1.65V to 3.6V
- ✓ There is an internal 10k pull-up resistor on each side (A and B) of the level shifter.
- ✓ Level shifters have been verified to support I²C communication (up to 1000kHz) and SPI communication (up to 10MHz) with Kionix sensors. For I²C communication, it is recommended to have additional pull-up resistors on SDA and SCL lines for faster transient switching. In many cases, there will be pull-up resistors on the evaluation board that come with the sensor. However, in other cases, it is recommended to populate 2.7k resistor at locations R88 (SDA) and R89 (SCL) on the *RoKiX Adapter Board A3*. Once connected, the effective resistance would be 2.1k on each signal level ($2.7k \parallel 10k = 2.1k$), which is a sufficient value for all Kionix sensors at all VDD voltages.
- ✓ It is possible to bypass the onboard level shifters if needed. This can be accomplished from removing the zero-ohm resistors on A and B sides of the level shifter and connecting *level shifter bypass* resistors (R50-R57).

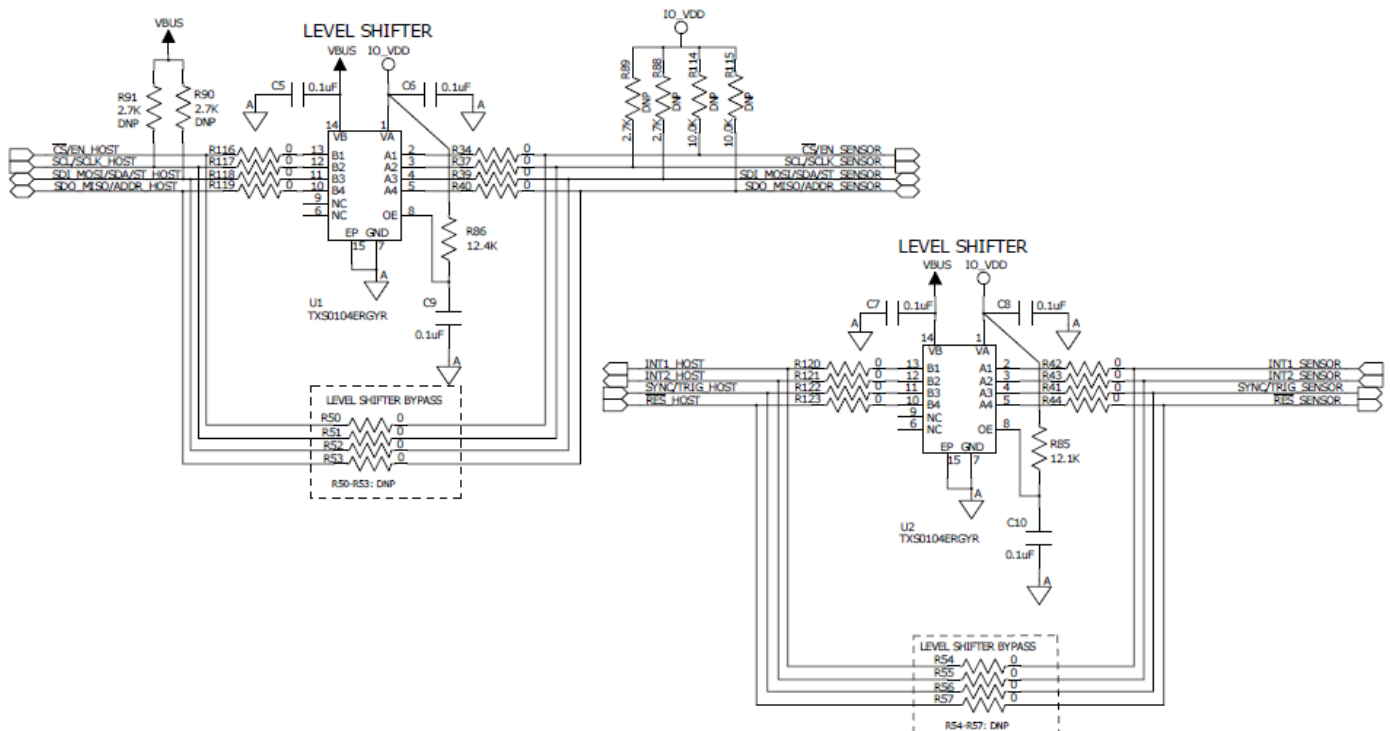


Figure 8. Voltage Level Shifters for I/O Signals

2. Interface with Evaluation Boards

2.1. Physical Interfacing with RoKiX Digital Evaluation Board

The *RoKiX Adapter Board A3* comes with pair of headers that provides an easy way to connect to the standard RoKiX Digital evaluation boards that come with 14-pin male header. One header is J5 14-pin male header, and another header is J6 18-pin female header (Figure 9).

NOTE: The 18-pin female header J6 is mechanically and electrically compatible with 14-pin male header found on RoKiX Digital evaluation board and simplify alignment of both connectors relatively to each other. The pins 1, 2 & 17, 18 on J6 are not connected electrically.



Figure 9. RoKiX Digital Evaluation Board Interface Headers

Thus, there are two ways to conveniently connect the standard RoKiX Digital evaluation board to the *RoKiX Adapter Board A3* – one is using the 14-pin ribbon cable plugged into J5 14-pin male header and another is by plugging the evaluation board directly into J6 18-pin female header of the (Figure 10)

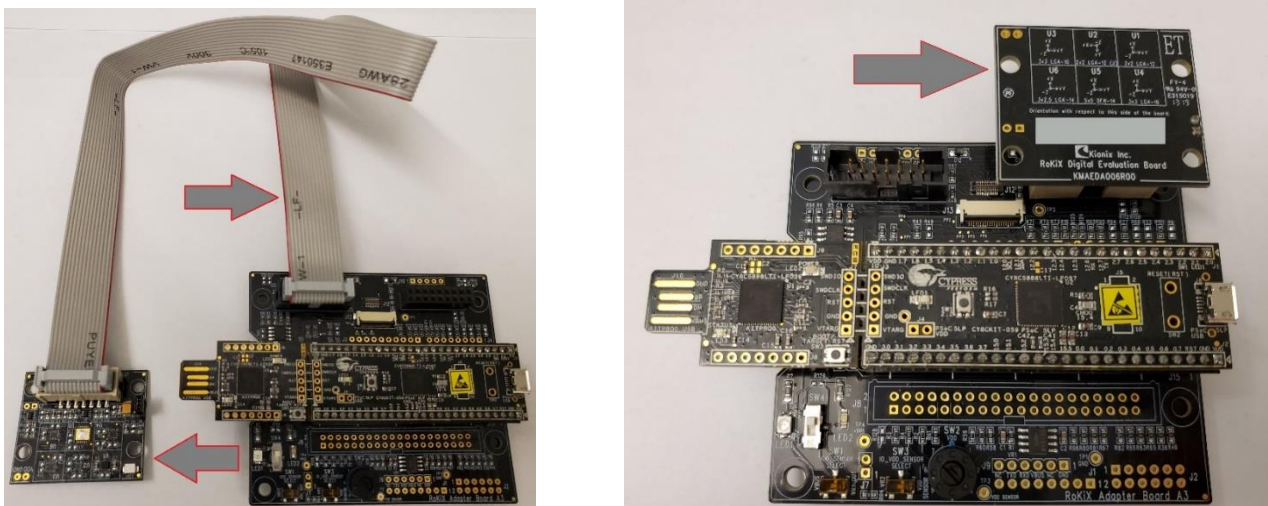




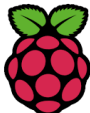
Figure 10. Interface with RoKiX Digital Evaluation Board

NOTE: For evaluation of analog sensors, the *RoKiX Adapter Board A3* is designed to interface with RoKiX Analog Evaluation Boards (black solder mask). The legacy Kionix analog evaluation boards (green solder mask), cannot be used due to a different routing of XOUT and YOUT analog signals on J5 & J6 headers.

3. Interface with host platforms

The *RoKiX Adapter Board A3* is designed to provide an easy hardware interface between Kionix / ROHM devices and numerous development platforms. Table 3 shows the list of host platforms that can be directly interfaced with the *RoKiX Adapter Board A3* via compatible headers. Note, that in order to interfaces with any given host platform, the proper hardware modifications may be required, including populating headers, and populating / removing certain zero-ohm resistors. By default, the *RoKiX Adapter Board A3* is factory populated to support Cypress CY8CKIT-059 PSoC® 5LP Prototyping kit of the box.

Table 3. List of Compatible Host Platforms

Vendor	Platform Name	RoKiX Adapter Board A3 HW Interface	Populated on the board?
	CY8CKIT-059 (PSoC 5LP)	J14, J15 (26 – pin headers)	Yes
		Various 0-ohm resistors*	Yes
	CY8CKIT-049 (PSoC 4)	J14, J15 (22 – pin headers)	No
		Various 0-ohm resistors*	No
	CY8CKIT-044 (PSoC 4M)	J14, J15 (29 – pin headers)	No
		Various 0-ohm resistors*	No
	CY8CKIT-042-BLE (PSoC4 BLE)	J1, J2, J3, J4 Arduino Compatible Headers	No
		Various 0-ohm resistors*	Yes
	CY8CKIT-042 (PSoC 4) CY8CKIT-062-BLE (PSoC 6 BLE)	J1, J2, J3, J4 Arduino Compatible Headers	No
		Various 0-ohm resistors*	Yes
CY8CKIT-043 (PSoC 4M)	J1, J2, J3, J4 Arduino Compatible Headers	No	
	Various 0-ohm resistors*	No	
	Arduino UNO R3	J1, J2, J3, J4 Arduino Compatible Headers	No
		Various 0-ohm resistors*	Yes
	Pi 1 Model A+ / B+ Pi 2 Model B Pi 3 Model B / B+	J8 40-pin Header	No

*See KMAAAA000R05_SCH schematic file for details.

3.1. Cypress CY8CKIT-059 Prototyping Kit

3.1.1. Overview

As was previously described, the *RoKiX Development Kit* uses *Cypress CY8CKIT-059 Prototyping Kit* as the target host adapter platform due to the numerous advantages it offers including high performance, mixture of onboard digital and analog peripherals, support for Full Speed USB 2.0 connectivity, easy to use IDE with free license, and the low cost.

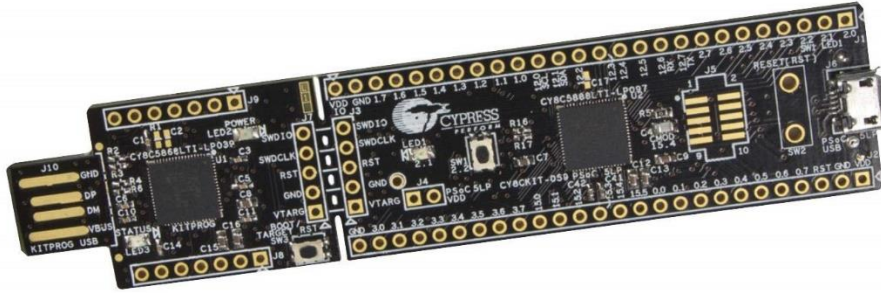


Figure 11. Cypress CY8CKIT-059 Prototyping Kit

When *Cypress CY8CKIT-059 Prototyping Kit* is shipped as part of the *RoKiX Development Kit*, it comes pre-loaded with Kionix's custom firmware, two 26-pin female headers soldered at locations J1 & J2, and is plugged into the *RoKiX Adapter Board A3* compatible male header J14 & J15 to provide the plug-and-play functionality out of the box (Figure 12).

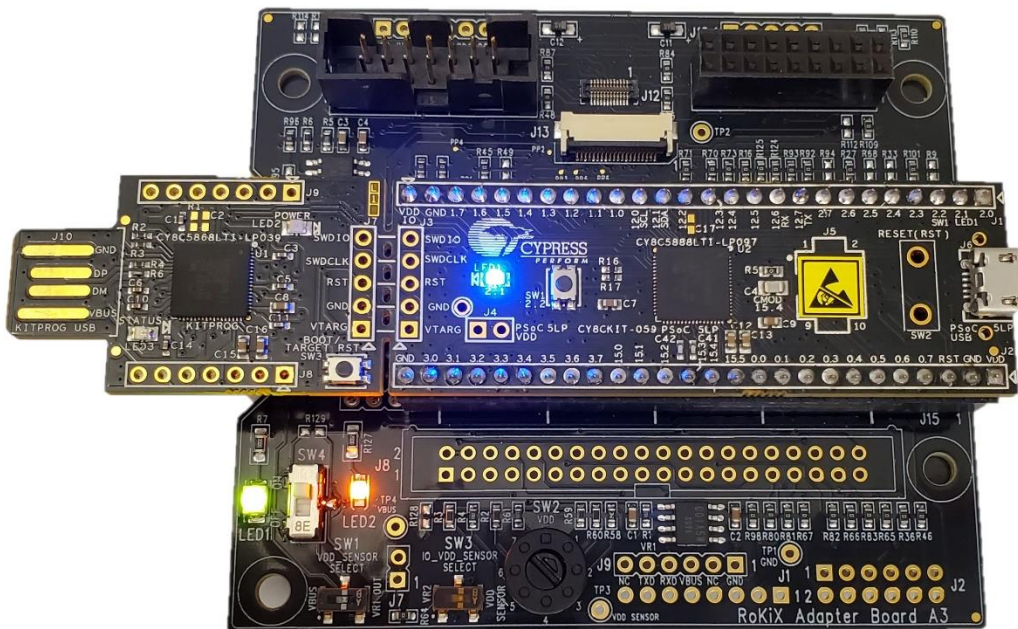


Figure 12. CY8CKIT-059 Plugged Directly in RoKiX Adapter Board A3

NOTE: The following content (3.1.1.1 - 3.1.1.3) is provided directly from the manufacturer's website ([LINK](#))

3.1.1.1. Featuring PSoC 5LP

"The CY8CKIT-059 PSoC 5LP Prototyping Kit features the [CY8C5888LTI-LP097](#) device from the PSoC 5LP family. PSoC 5LP is the industry's most integrated programmable SoC, combining high-precision and programmable analog and digital peripherals with an ARM® Cortex®-M3 CPU in a single chip. Process sensor signals with the 24-bit hardware DFB coprocessor, offload traditional CPU tasks to the CPLD-based Universal Digital Blocks and increase system performance with the peripheral-to-peripheral DMA controller. Integrate high-precision custom 20-bit Analog Front Ends with the Programmable Analog Blocks including opamps, PGAs, filters, comparators, SAR and Delta-Sigma ADCs and the industry's best CapSense touch-sensing solution."

3.1.1.2. Design for Flexibility

"The kit provides access to all the PSoC 5LP device I/Os in a breadboard-compatible format. It features a micro-USB header for creating prototypes with Full Speed USB 2.0 connectivity. The kit is also designed with a convenient snappable form-factor, allowing users to separate the USB connector with the KitProg Programmer and Debugger from the target board to use them independently. Once done with the prototype, you're still left with a handy SWD programmer!"

3.1.1.3. Low-Cost Programmer

"The kit includes Cypress's KitProg Programmer and Debugger. KitProg can program and debug the target PSoC 5LP device via SWD when using PSoC Creator or PSoC Programmer. It supports bridging over USB-UART and USB-I2C interfaces and also provides access to Micrium µC/Probe to read and write memory on the target device. When snapped away, this tiny USB board can be used as a KitProg programmer and debugger with any PSoC 3, PSoC 4 or PSoC 5LP device. The KitProg firmware is provided as a bootloader image that can be upgraded to develop custom applications for it."

3.1.2. Firmware Pinout

The interface between the Cypress PSoC microcontroller mounted on the *CY8CKIT-059 Prototyping Kit (RoKiX Adapter Board A3)* headers J14, J15) and the Kionix's sensors mounted on the evaluation board and either plugged directly to J6 (18-pin receptacle header) or via ribbon cable plugged into J5 (14-pin male header) on the *RoKiX Adapter Board A3* is shown in Table 4.

Function in Firmware	14-pin Male Header Pin J5	18-pin Receptacle Header Pin J6	A3 Board PSoC Headers Pin J14, J15	PSoC 5LP I/O Port	A3 Board Zero Ohm
Not used in firmware	2	4	J14-1	P2.0	R101
SPI (nCS) / Enable (Analog)	2	4	J14-22	P1.5	R21
X_OUT	3	5	J15-21	P3.4	R108
Y_OUT	4	6	J15-20	P3.5	R31
SPI (SCLK)	5	7	J14-11	P12.5	R16
I2C (SCL)	5	7	J15-11	P0.0	R32
SPI (MOSI/SDI)	7	9	J14-23	P1.6	R12
I2C (SDA)	7	9	J15-10	P0.1	R26
SPI (MISO/SDO) / ADDR	9	11	J14-24	P1.7	R23
SYNC/TRIG	10	12	J14-4	P2.3	R99
INT1	11	13	J14-13	P12.3	R17
INT2	12	14	J14-5	P2.4	R27
nRES	13	15	J14-6	P2.5	R125
Z_OUT	14	16	J15-19	P3.6	R29

Table 4. Physical Mapping of I/O Signals to the Cypress PSoC 5LP MCU

3.1.3. USB Driver

Before connecting the *RoKiX Adapter Board A3* to the computer, it is highly recommended to install the *RoKiX IoT Platform* software first using the installer file available for download at ROHM Semiconductor website:

- <https://www.rohm.com/support/accelerometer-evk-support>

If *RoKiX IoT Platform* software installer is used, a separate USB driver installation for Cypress *CY8CKIT-059 Prototyping Kit* is not required. Also, the Windows 10 operating system should automatically use the correct USB driver. However, earlier Windows versions are not able to automatically find the CDC ACM driver and the user will need to install the signed release `inf` file as describes in section 3.1.3.1.

3.1.3.1. USB Driver Installation Procedure

1. Following the installation of the *RoKiX IoT Platform* software, locate the folder *cdc_acm_driver* on the computer in the following location:

- \Documents\RoKiX\RoKiX-Firmware\Windows-dependencies\RoKiX-USB-driver\cdc_acm_driver

and verify that the presence of the “following two files in the above-mentioned directory”

- rokix_cdc_acm.cat
- rokix_cdc_acm.inf

2. Connect the Cypress CY8CKIT-059 Prototypic Kit to a computer using the provided microUSB cable (Figure 13).

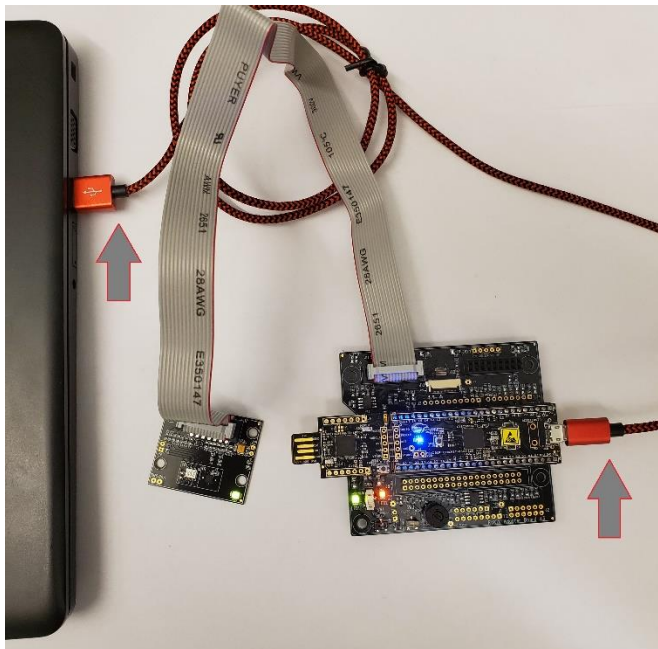


Figure 13. CY8CKIT-059 connected to PC (connection to the evaluation board is optional in this step)

- Open the Device Manager where you should find the "Evaluation Kit (Cypress)" (Figure 14)

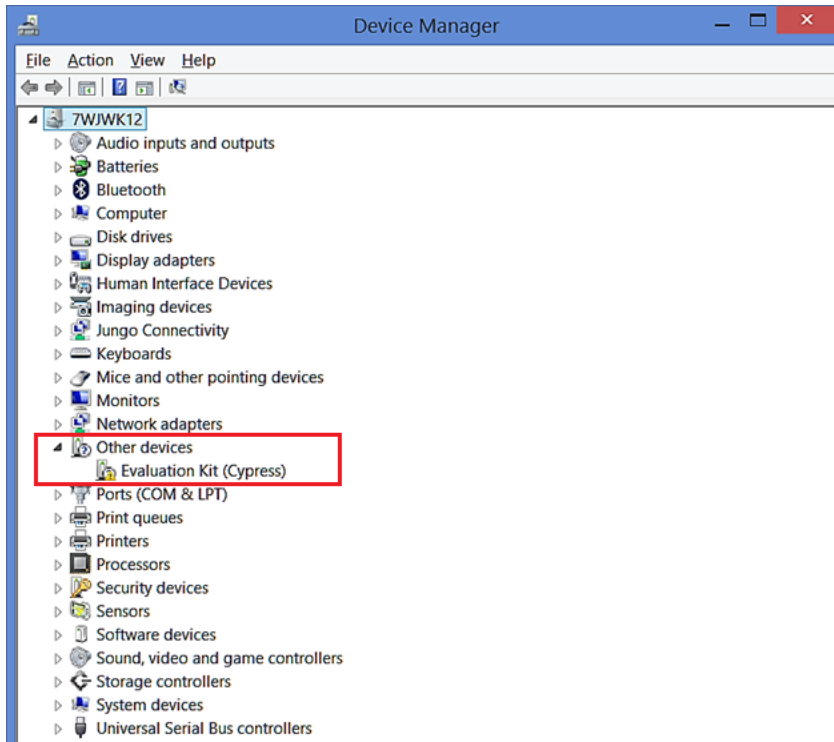


Figure 14. Device Manager View

- Right-click the "Evaluation Kit (Cypress)" item, and choose "Update Driver Software ...". A new window should open. Select "Browse my computer for driver software" (Figure 15)

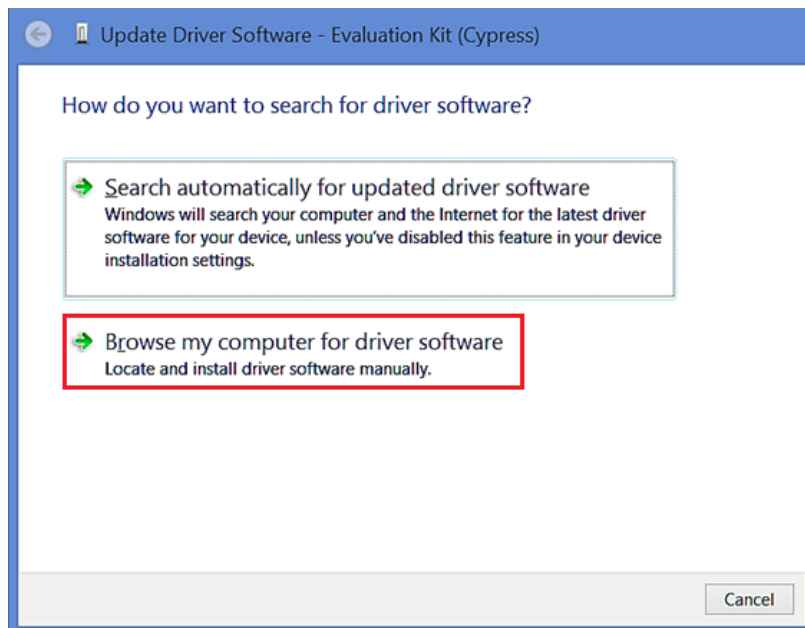


Figure 15. Update Driver Software - 1

5. Select "Let me pick from a list of device drivers on my computer": (Figure 16)

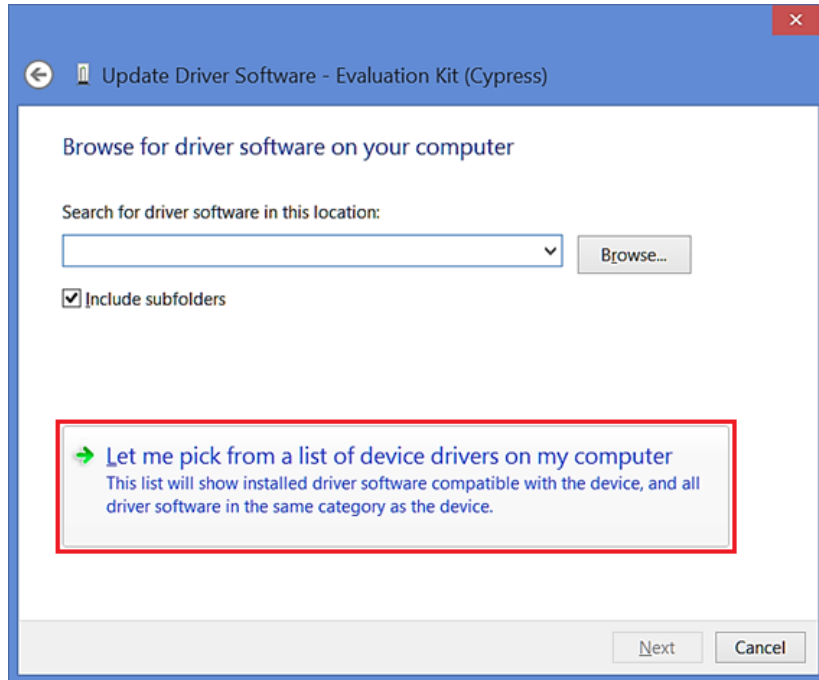


Figure 16. Update Driver Software - 2

6. Choose "Next" (the selection in the list does not matter) (Figure 17):

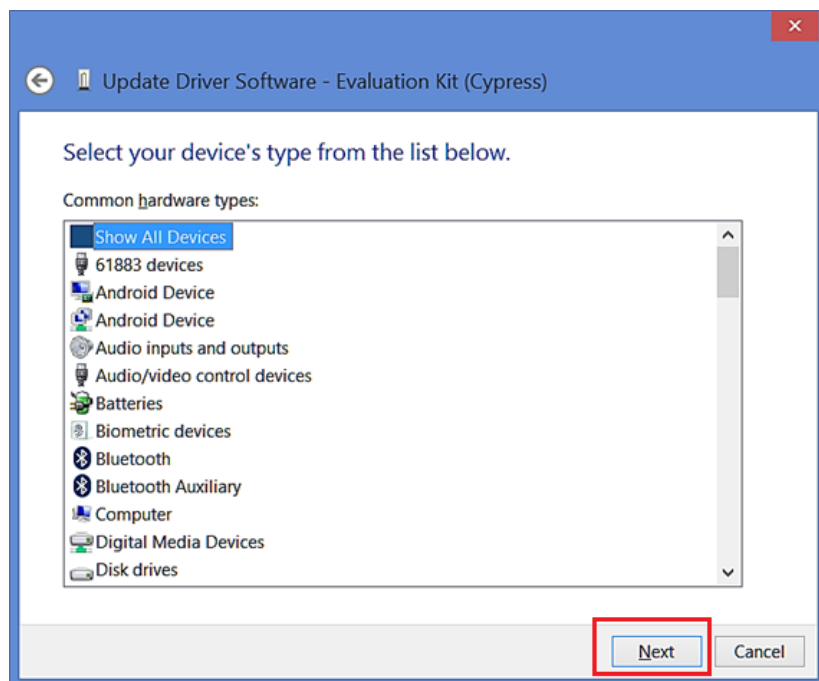


Figure 17. Update Driver Software - 3

7. Choose "Have Disk" (Figure 18)

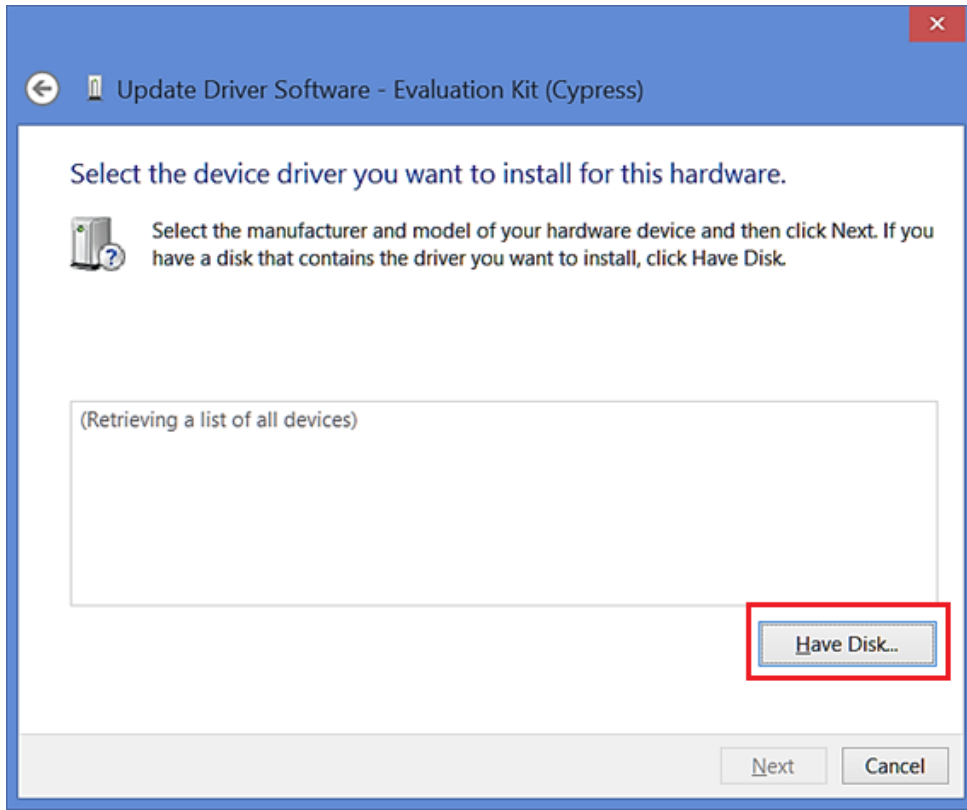


Figure 18. Update Driver Software - 4

8. Then choose "Browse" (Figure 19) and find the downloaded inf file and then select "OK". You will be returned to an older pop-up window where you should select "Next" (Figure 20)

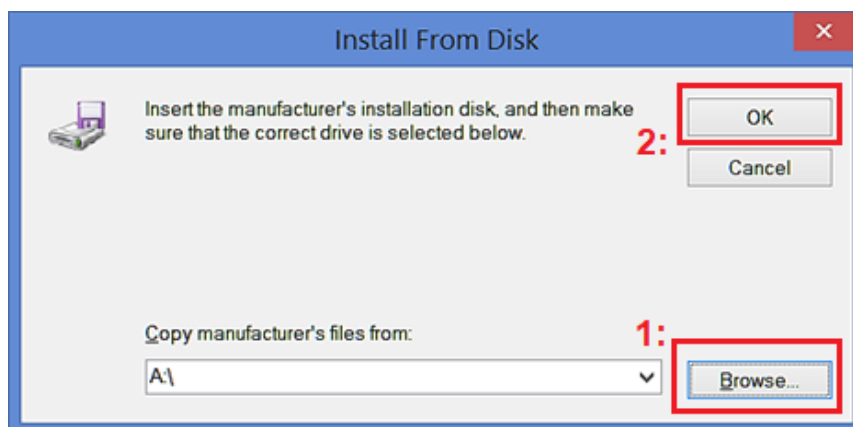


Figure 19. Update Driver Software - 5

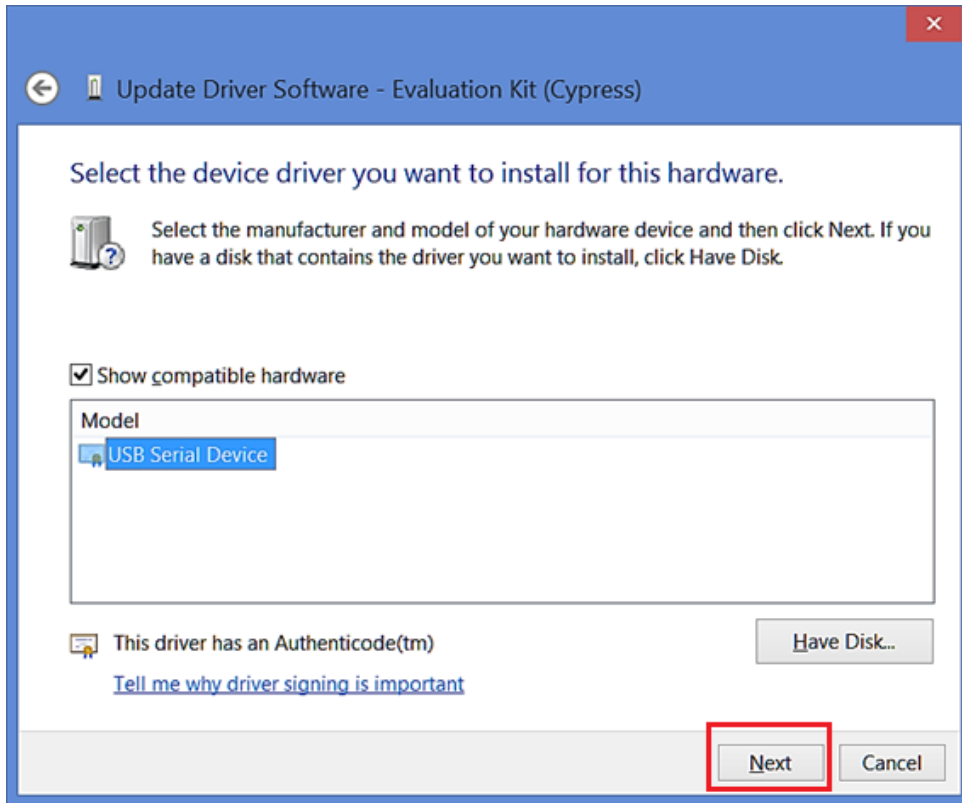


Figure 20. Update Driver Software – 6

9. Windows will prompt you to install the driver, please select "Install" (Figure 21).

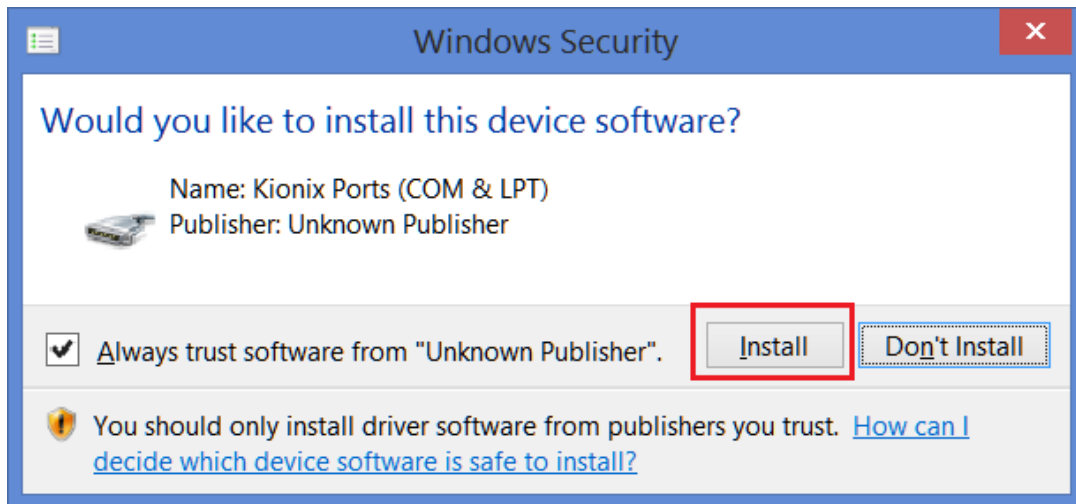


Figure 21. Update Driver Software – 7

10. Finally, please wait until the driver installation is complete.

3.1.4. Firmware

When Cypress *CY8CKIT-059 Prototyping Kit* is supplied as part of the *RoKiX Development Kit*, it will come pre-loaded with the *RoKiX Firmware* to interface with *RoKiX Windows GUI*. To upload the latest *RoKiX Firmware* to the *CY8CKIT-059 Prototyping Kit* please follow the procedure outlined in the section 3.1.4.1.

3.1.4.1. Firmware Update Procedure

1. Download and Install the PSoC Programmer (Windows) from the Cypress's website:
<https://www.cypress.com/products/psoc-programming-solutions>
2. Following the installation of the *RoKiX IoT Platform* software, locate the folder *Cypress-PSoC* on the computer in the following location:
 - \Documents\RoKiX\RoKiX-Firmware\Cypress-PSoC
 and verify that the presence of a *RoKiX Firmware* file with an extension *.hex*.
3. Connect the *CY8CKIT-059 Prototyping Kit* into the USB port of the PC directly or with a *USB extension cable A-Male to A-Female* as shown in Figure 22 (note that firmware flashing is always done via USB-A PCB connector, not micro-USB connector on the other side of the board).

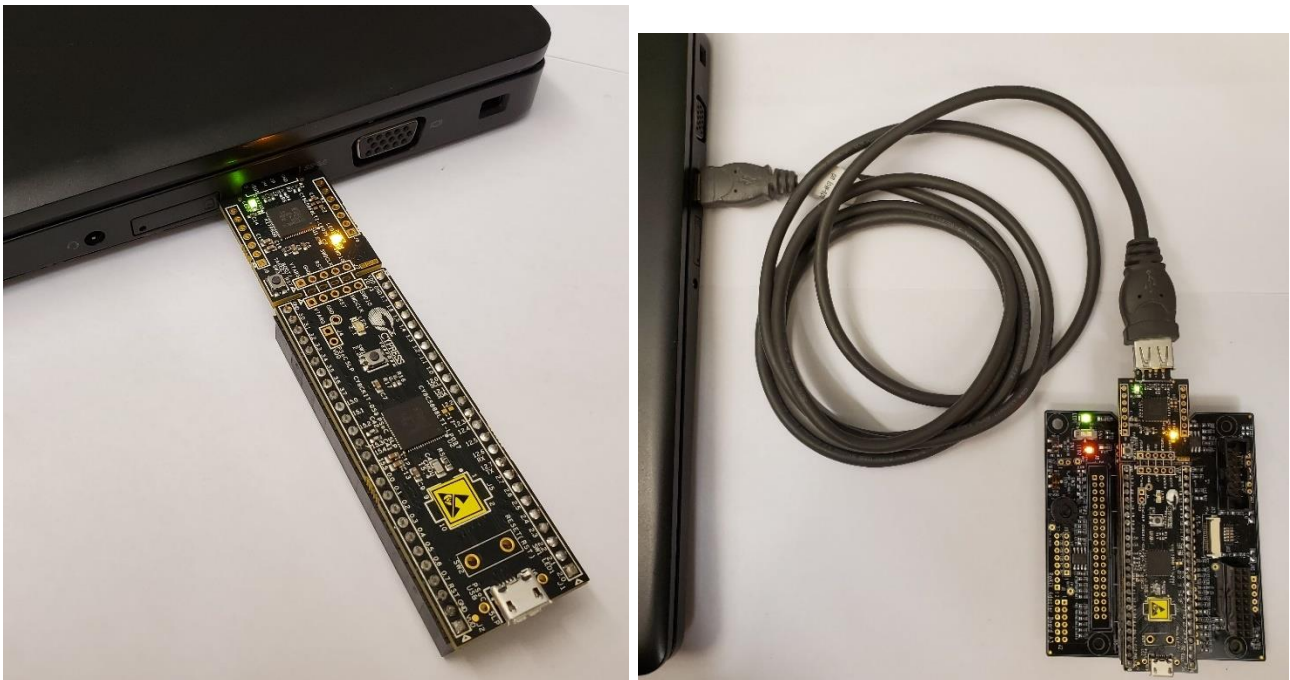


Figure 22. *CY8CKIT-059 Prototyping Kit* Connected to the PC

4. Open the *PSoC Programmer* application on your computer. Once opened, verify that Powered and Connected status messages are displayed in the status bar – the bottom right side of the window (Figure 23). If not, please verify the *CY8CKIT-059 Prototyping Kit* is properly plugged into the USB port.

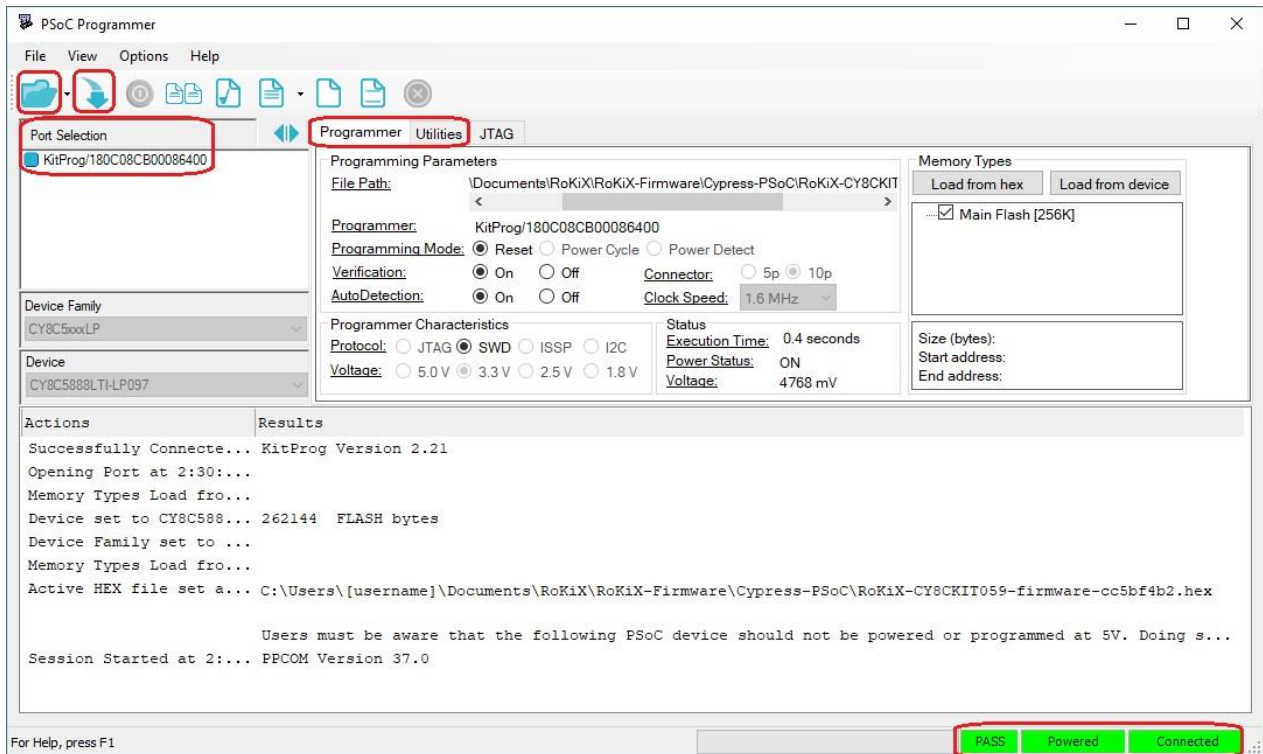


Figure 23. PSoC Programmer GUI

If you received a Warning message “This Programmer is currently out of date”, or “The communication firmware on the kit does not match what is installed with the release of the PSoC Programmer”, click OK button to navigate to *Utilities* tab in the *PSoC Programmer*, and click the *Upgrade Firmware* button (Figure 24). When firmware upgrade is completed, go back to *Programmer* tab, and proceed to the next step.

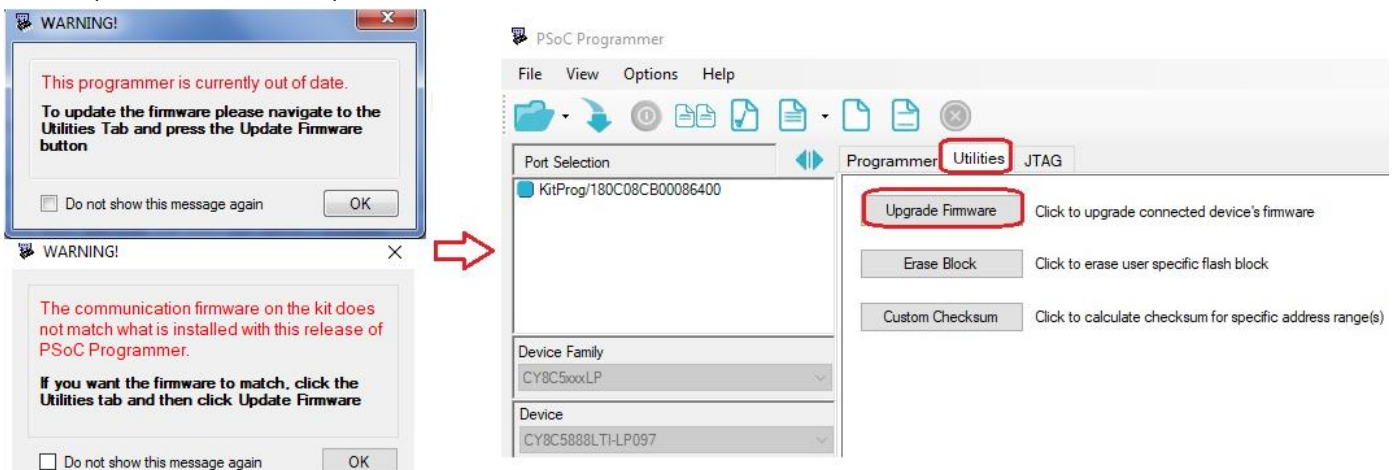


Figure 24. Programmer Firmware Update

5. Next, select the firmware *hex* file mentioned in the step 2 by either pressing the Open Folder button or through the menu (File>File Load) or by pressing the F4 key (Figure 23).
6. Next, flash the firmware on the *CY8CKIT-059 Prototyping Kit* by pressing the Down Arrow button or through the menu (File>Program) or by pressing the F5 key. (Figure 23)

If programming was successful, the *Programming Succeeded* message would be shown in the Results window (Figure 25).

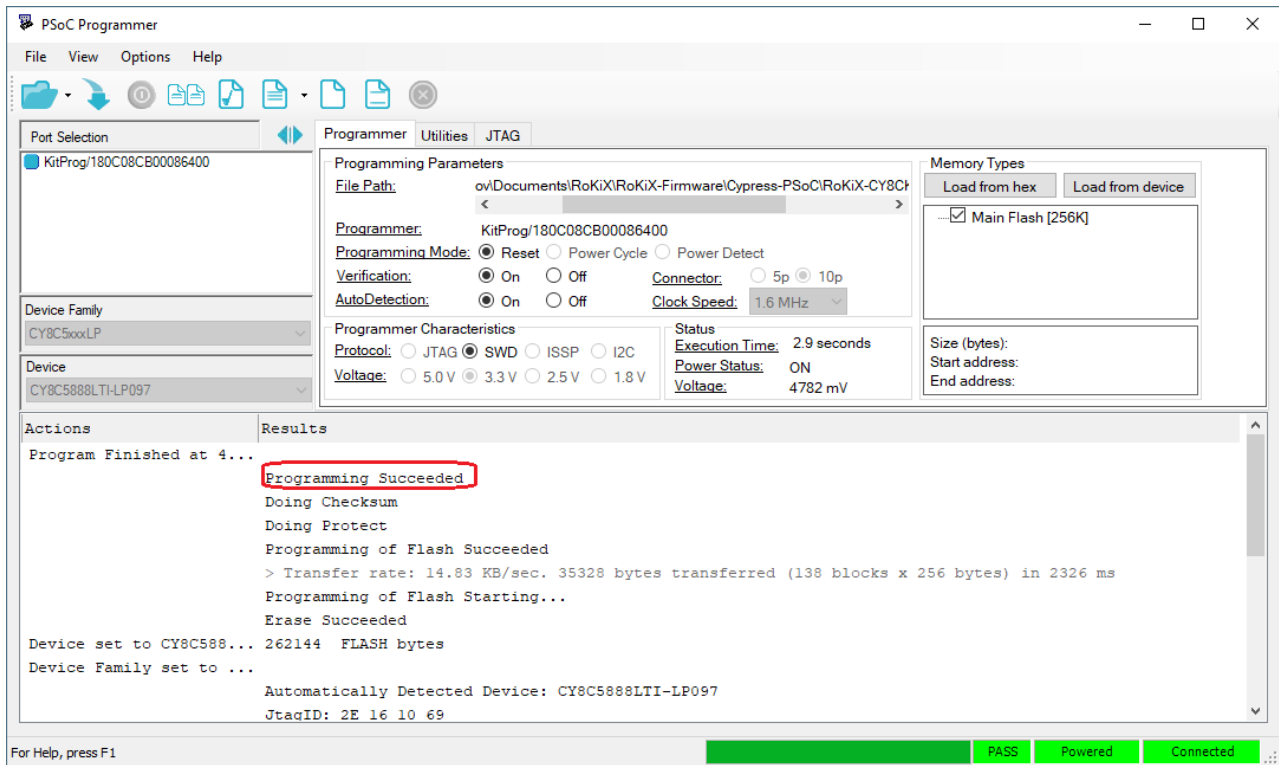


Figure 25. Firmware Upgrade Success Message

7. You are now ready to use *CY8CKIT-059 Prototyping Kit* with the RoKiX Windows GUI!

4. Getting Started with RoKiX Windows GUI

4.1. Introduction

The RoKiX Windows GUI, which is a part of the RoKiX IoT Platform Client Software, provides an easy to use graphical user interface demonstrating high level device offerings and features. Some of the features include:

- A visual display of real-time device data
- Ability to record device data onto a file
- Device registry editor

RoKiX Windows GUI is compatible with Windows OS versions 7, 8 and 10.

4.2. Setup

4.2.1. Installation

If RoKiX IoT Platform Software has not been installed already, it can be installed by downloading the latest installer file from the ROHM Semiconductor website:

- <https://www.rohm.com/support/accelerometer-evk-support>

Before running the RoKiX Windows GUI software, necessary USB serial drivers must be installed (if Windows does not install these drivers automatically). See section 3.1.3 for details.

4.2.2. Configuration

Please follow the following basic procedure to start using the RoKiX Windows GUI:

- ✓ Attach the KX132-1211 RoKiX Digital Evaluation Board to the RoKiX Adapter Board either directly with the provided ribbon cable (Figure 10).
- ✓ Connect the RoKiX Development kit to the PC with the provided micro-USB cable (Figure 13).
- ✓ Launch RoKiX Windows GUI application.

After the installation, the shortcuts to the RoKiX Windows GUI and to the RoKiX Development Kit User's Guide can be found on the desktop, in the Windows Start menu under RoKiX folder, and in the installation directory:

- \Documents\RoKiX\

- ✓ If Configuration update pop-up window is shown, click Yes to download the latest configurations from the server (Figure 26).

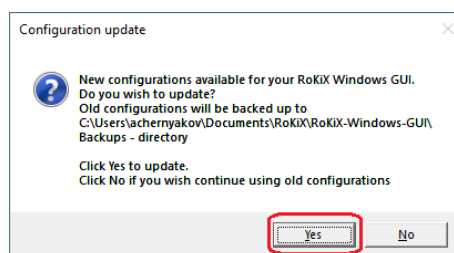


Figure 26. Configuration Updated Popup Window

✓ Select the board configuration from the Board menu:

1. **CY8CKIT-059 / RoKiX adapter A3 / I2C**

✓ Select the desired configuration stream for the corresponding accelerometer sensor from the Stream menu: e.g.:

2. KX132-1211 / Accel data 50Hz ±2g high performance

NOTE: I²C interface can be used for all digital sensors but the highest Output Data Rate (ODR) will be limited to ~3400Hz.

NOTE: For sensors that support SPI interface (e.g. KX132-1211, KX134-1211), select CY8CKIT-059 / RoKiX adapter A3 / SPI board configuration that supports ODR up to 25600Hz.

✓ If the “Please enable streaming to activate Plotter movement!” – Pop-up window appears on the screen, enable data streaming with “Streaming” button.

The plotter should now display real time output for X, Y, and Z axes of KX132-1211 sensor according to its movement (Figure 27).

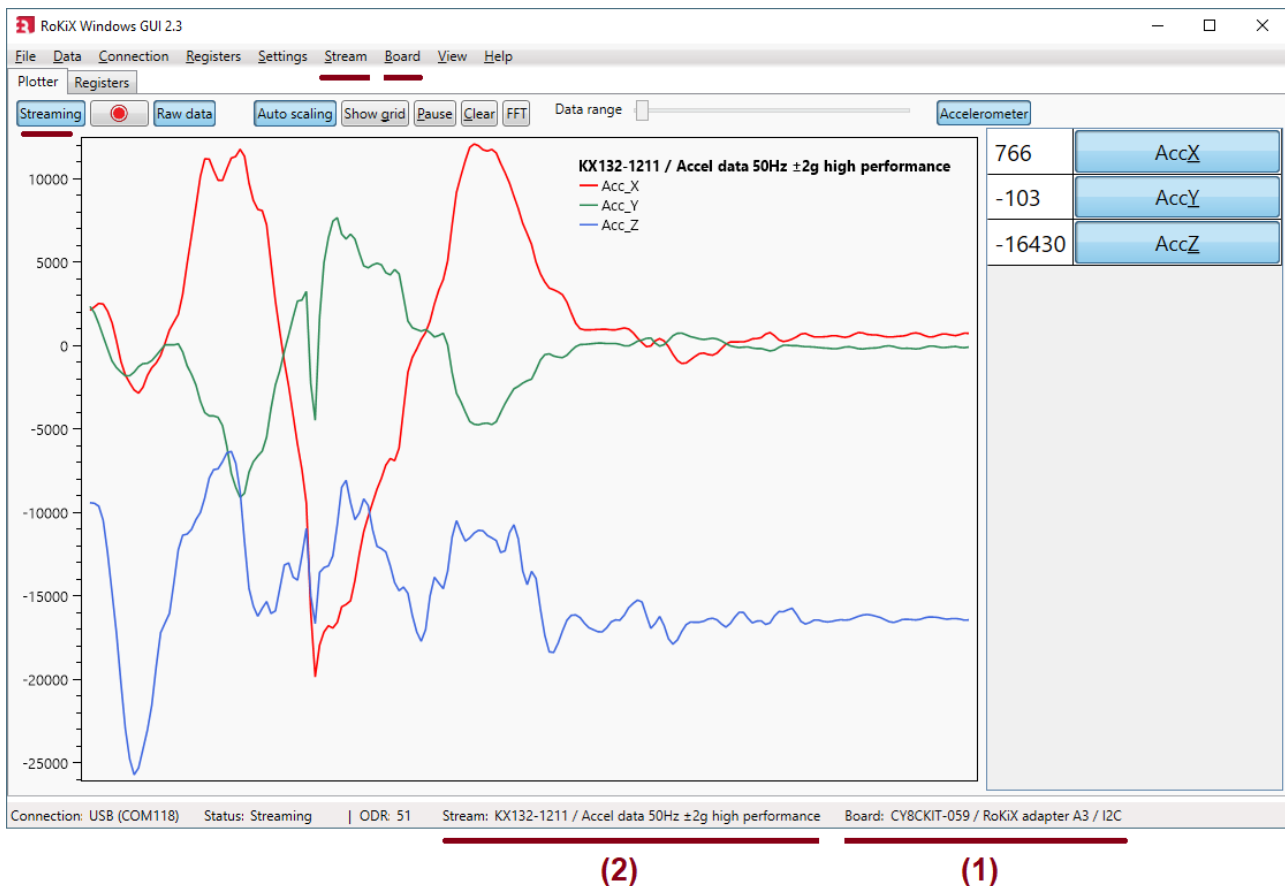
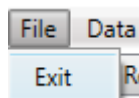


Figure 27. Plotter view with KX132-1211 motion displayed

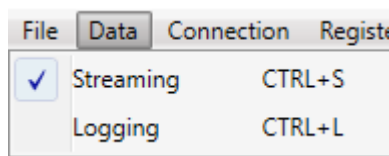
4.3. User Interface – Menu bar

4.3.1. File – Menu



The File menu contains only the option to exit the application. Selecting “Exit” will exit from the application.

4.3.2. Data – Menu



The Data menu contains the options related to acquiring the data.

4.3.2.1. Streaming

The streaming menu is used for enabling / disabling device data streaming. Shortcut: CTRL + S

NOTE: Data stream enabling / disabling may take a while, so please be patient.

4.3.2.2. Logging

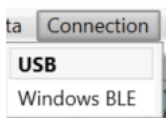
The logging menu is used for enabling / disabling device data logging. The status bar will show the log file name.

Logging to log_file.txt

Shortcut: CTRL + L

4.3.3. Connection – Menu

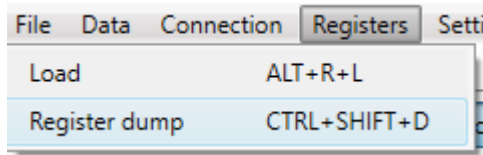
The *RoKiX Windows GUI* connects to the *RoKiX Development Kit* via USB COM port. The Bluetooth connection (Windows BLE) is reserved for other host platforms supported by the *RoKiX IoT Platform*. The *RoKiX Windows GUI* uses USB COM connection by default. When auto-connect is enabled, the USB connection is established automatically when the *RoKiX Development Kit* is connected.



NOTE: Changing connection may take a while, please be patient.

NOTE: If you are having connection problems, “CTRL + R” can be used to refresh current connection.

4.3.4. Registers - Menu



4.3.4.1. Load

Selecting "Load" will allow the user to load the device's register definition file from the "Registers" – tab (refer to section 4.4.3 for details).

4.3.4.2. Register dump

Selecting "Register dump" allows users to save the current value of all the register of the selected sensor to the text file (Figure 28).

```

Register dump for KX132-1211

0x00 MAN_ID           0x4B 0b01001011
0x01 PART_ID          0x3D 0b00111101
0x02 XADP_L           0x00 0b00000000
0x03 XADP_H           0x00 0b00000000
0x04 YADP_L           0x00 0b00000000
0x05 YADP_H           0x00 0b00000000
0x06 ZADP_L           0x00 0b00000000
0x07 ZADP_H           0x00 0b00000000
0x08 XOUT_L           0x43 0b01000011
0x09 XOUT_H           0xFF 0b11111111
0x0A YOUT_L           0x30 0b00110000
0x0B YOUT_H           0x00 0b00000000
0x0C ZOUT_L           0x1B 0b00011011
0x0D ZOUT_H           0xC0 0b11000000
    
```

Figure 28. Partial Snapshot of the KX132-1211 Register Dump

NOTE: The default location to save the Register Dump file is:

\\Documents\RoKiX\RoKiX-Windows-GUI\log_files

NOTE: Before performing the "Register dump", the corresponding Register XML file needs to be loaded first (see Section 4.3.4.1), if you receive a "Notification" dialog box (Figure 29).

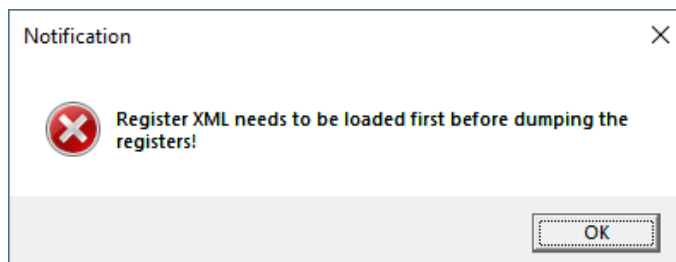
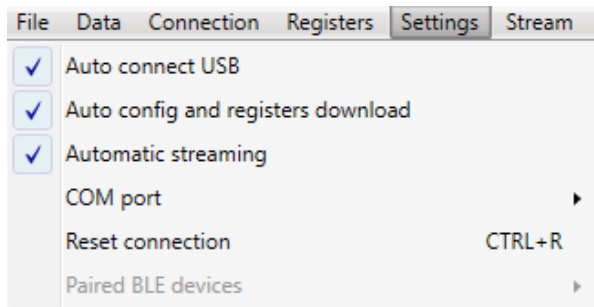


Figure 29. XML File Loading Notification Dialog Box

4.3.5. Settings – Menu



4.3.5.1. Auto Connect USB

When “Auto connect” is enabled, the *RoKiX Windows GUI* will automatically select the USB COM port for the connected device and connect to it.

4.3.5.2. Auto config and register download

When “Auto config download” is enabled, the *RoKiX Windows GUI* will automatically check and download the latest board and stream configurations. The user will be notified when there are new configurations available for download.

4.3.5.3. Automatic streaming

When “Automatic streaming” is enabled, the *RoKiX Windows GUI* will automatically start data streaming when the device stream is changed.

4.3.5.4. COM port

When there are multiple devices connected or there is some problem with the USB COM port selection, the COM port can be selected from the dropdown list. Before doing this the “Auto-connect” feature must be disabled.

4.3.5.5. Reset connection

If you are having connection problems, “Reset connection” can be used for refreshing the current connection. It also initializes the current data stream again.

Shortcut: CTRL + R

4.3.5.6. Paired BLE devices

The Bluetooth communication is available by RoKiX IoT Platform but is not supported by the RoKiX Development kit.

4.3.6. Stream - Menu

The stream menu is used for selecting the device data stream which the application uses to receive data. The list of streams is dynamic and will change according to the chosen board configurations (section 4.3.7). For example, if the *CY8CKIT-059 / RoKiX Adapter A3 / SPI* board configuration is selected from the Board menu, the KXTJ3 sensor will not be shown in the Stream menu because it does not support SPI interface. However, when *CY8CKIT-059 / RoKiX Adapter A3 / I2C* board configuration is selected, the KXTJ3 sensor will be shown (Figure 30).

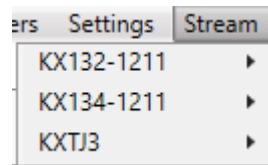


Figure 30. RoKiX Windows GUI stream menu.

NOTE: The RoKiX Windows GUI will store the last used stream configuration and it will be loaded in the next application startup.

4.3.7. Board – Menu

The RoKiX Windows GUI support multiple host adapters (Arduino, *CY8CKIT-059 Prototyping Kit*, nRF52840-DK,) each in turn supports one or two communication interface protocols (I²C, SPI). The correct board configuration must be selected based on the host adapter used and the communication interface protocol supported by the sensor that is being evaluated. Note that the Board menu lists either all supported board configurations for all supported Host Adapters (Figure 31 – left) or only the relevant ones that are supported by the Host Adapter currently plugged in, e.g. *CY8CKIT-059 / RoKiX Adapter A3 / I2C* that comes standard with RoKiX Development Kit (Figure 31 – right). The selection is done via View menu (section 4.3.8.4).

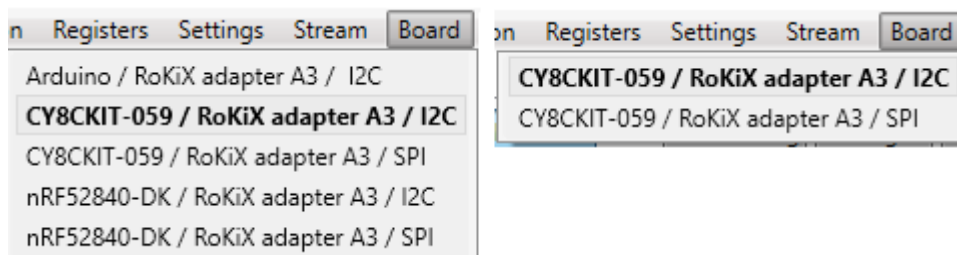
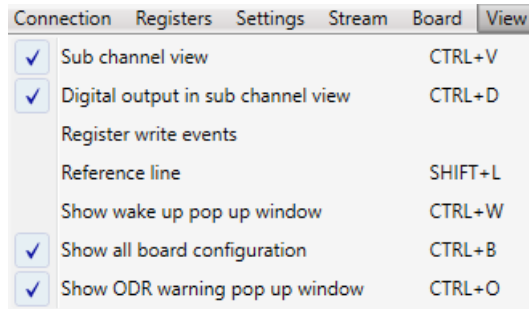


Figure 31. RoKiX Windows GUI Board menu – Full List (Left) and only for connected platform (right)

The default board configuration is “CY8CKIT-059 / RoKiX Adapter A3 / I2C”.

4.3.8. View – Menu

The View menu item provides different features that can be shown or hidden in the *RoKiX Windows GUI*.



4.3.8.1. Sub-channel view & Digital Output in sub channel view

The “Sub channel view” is enabled by default and also via View menu or CTRL+V keyboard shortcut, the plotter shows an additional side panel on the right which can be used to show/hide the sub channel view. It also has a column for the digital output of each sub channel which can be enabled with “Digital output in sub channel view” submenu item or CTRL+D keyboard shortcut.

NOTE: Available sub channel views are always related to the used device stream and the data channels inside of it.

NOTE: When you have a very high ODR (12.8 kHz or 25.6 kHz), the digital output can slow down the plotter’s performance. (Figure 32)



Figure 32. Plotter view with Sub Channel View and Digital Output Enabled

It is possible to select which channels are enabled by pressing on the channel name. For example, for a default stream such as *Accelerometer data 50Hz ±2g high performance* of the KX132-1211, in order to monitor the accelerometer's Z axis only, press once on AccX and AccY to disables these channels. The only remaining channel would be AccZ as shown in Figure 33.

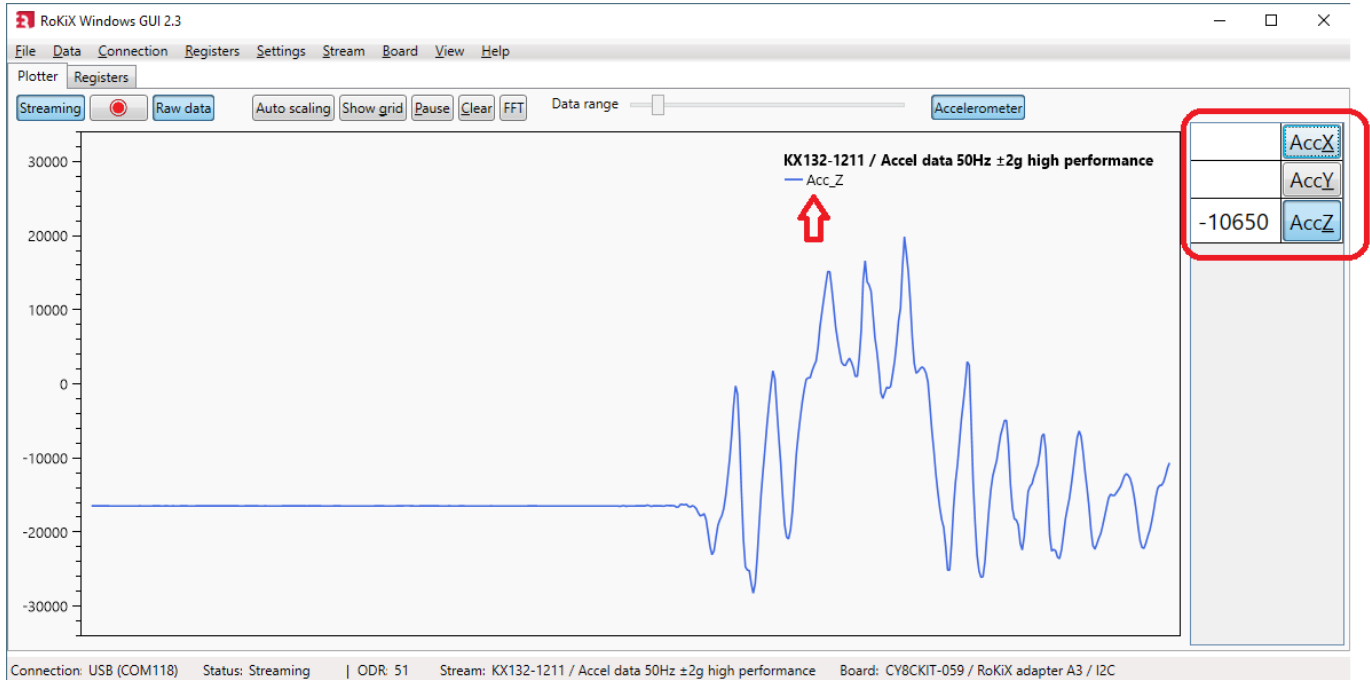


Figure 33. Plotter view *with only* AccZ sub-channel activated.

4.3.8.2. Register write events

When *Register write events* function is enabled from View menu. When selected, the register write events will be show in the output window located below the plot window. (Figure 34).

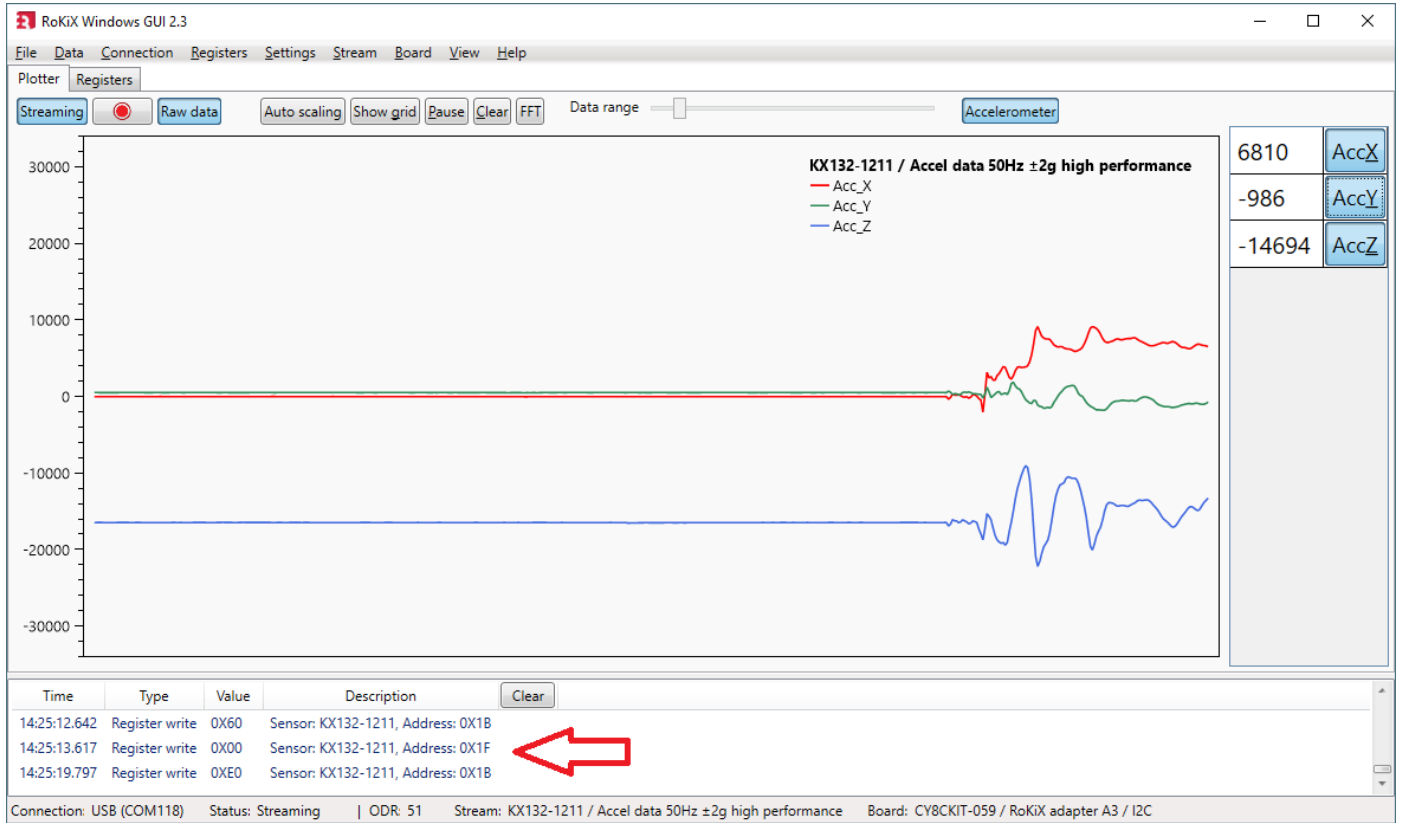


Figure 34. Register Write Events output window

4.3.8.3. Reference line

When the “Reference line” is enabled, the plotter shows an additional horizontal line that can help to compare the real time signal value against the referenced value (Figure 35). The line can be dragged up/down the plotter view by pressing and holding the left button mouse. The present value of the Reference line position is also shown in the Status bar (lower right corner of the window). To achieve a higher resolution / granularity when setting the reference line position, use the mouse scroll wheel to zoom into and out of the plotter window.

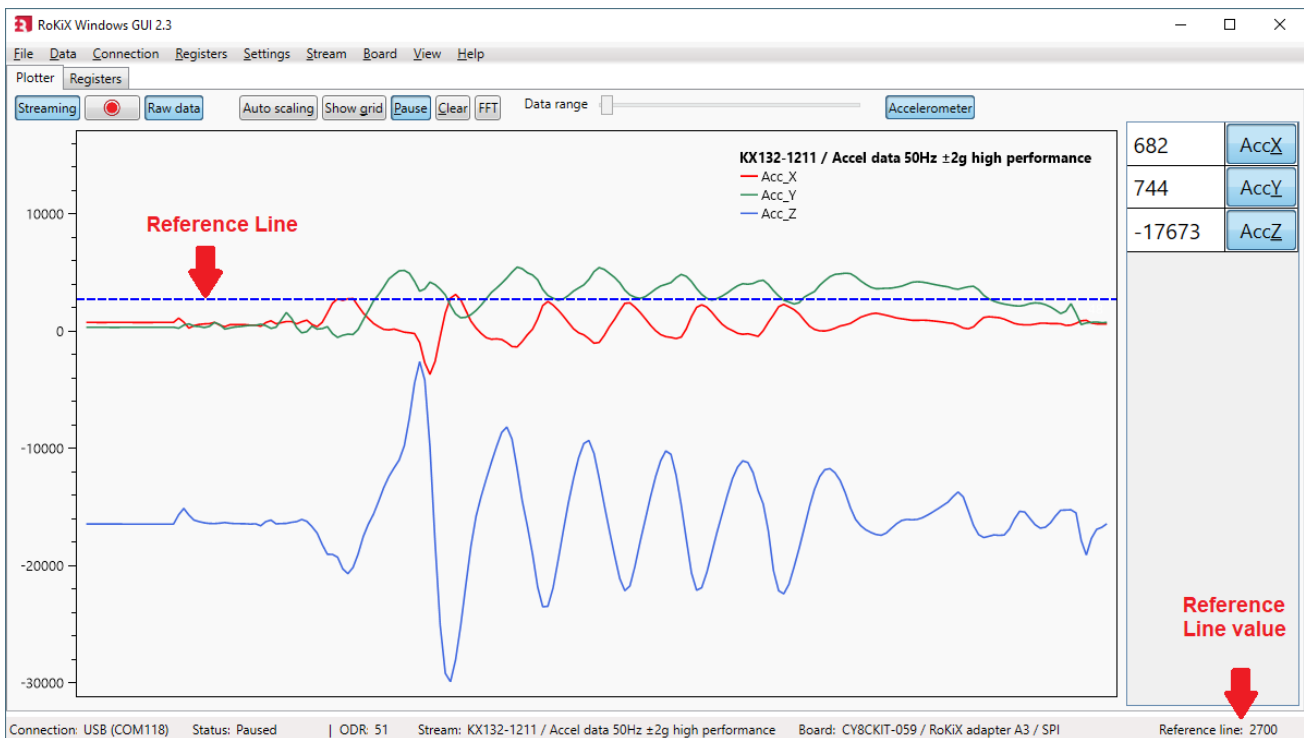


Figure 35. Plotter view with reference line enabled.

4.3.8.4. Show wake up pop up window

The “Show wake up pop up window” sub menu item is visible only for selected wake-up / back-to sleep detection streams. When the wake-up event is detected, the pop up window is displayed in the plotter view (0). The item is also enable using CTRL + W shortcut.

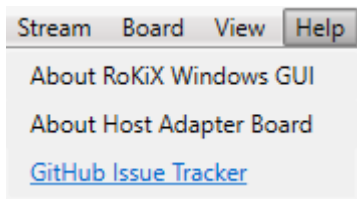
4.3.8.5. Show all board configurations

The “Show all board configurations” sub menu item controls whether Board menu item lists all supported board configurations for all supported Host Adapters or only the relevant ones that are supported by the Host Adapter currently plugged in (4.3.7).

4.3.8.6. Show ODR warning pop up window

The ODR warning pop up window is shown anytime when the real time Output Data Rate (ODR) as measured by the *RoKiX Windows GUI* is significantly different from the nominal ODR set in the Stream (4.6.4).

4.3.9. Help – Menu



4.3.9.1. About RoKiX Windows GUI

The About RoKiX Windows GUI help menu shows the current *RoKiX Windows GUI*'s version and which Git commit it is from and when it has been built. It also shows the link to GitHub repository where the latest version can be downloaded from (Figure 36).

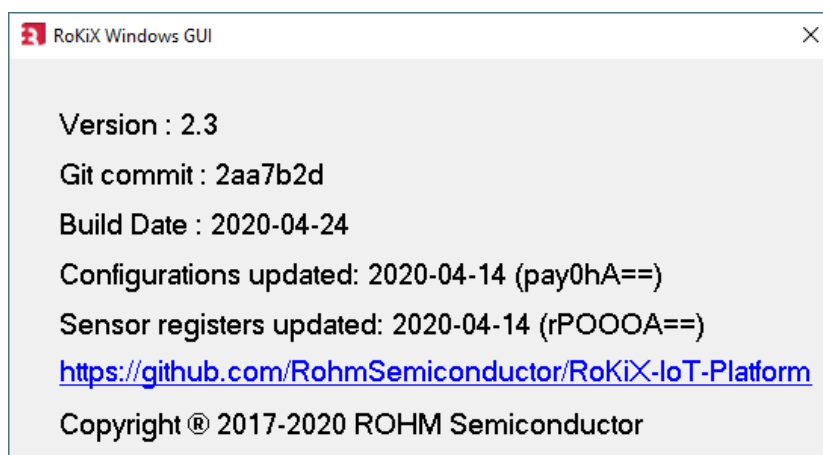


Figure 36. About *RoKiX Windows GUI* menu

4.3.9.2. About Host Adapter Board

The About Host Adapter Board help menu is enabled when a host adapter board is connected. The provided information includes its firmware information (RoKiX Protocol Version, RoKiX Firmware Version), as well as the hardware information such as Board ID and Board Unique ID (UID) when available (Figure 37).

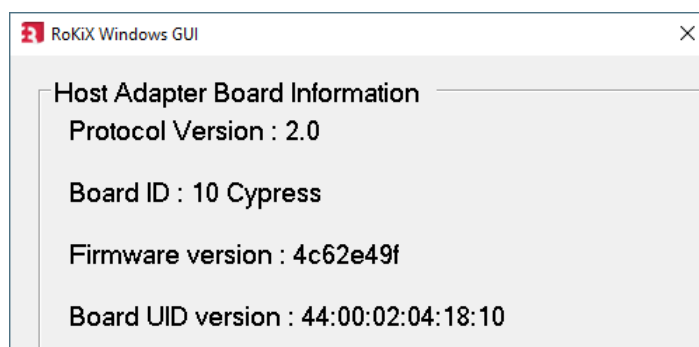


Figure 37. About Host Adapter Board Information menu for *CY8CKIT-059 Prototyping Kit*

4.3.9.3. GitHub Issues Tracker

A link to the *GitHub Issues* shows information on ToDos items, feature requests, and bugs on the Windows GUI software as well as other RoKiX firmware and software.

4.4. User Interface - Tabs

The functionalities of the *RoKiX Windows GUI* are divided between separate tabs.

4.4.1. Plotter – Tab

The Plotter in Figure 27 shows device data from the current stream. The Plotter has its own Streaming and Raw data buttons in order to change them quickly. (Figure 38).



Figure 38. RoKiX Window GUI menu bar view.

Data logging can be enabled/disabled easily with the button with the red circle icon.

NOTE: When logging is enabled the red circle icon starts to blink.

When Auto scaling is enabled, plotter will auto scale the minimum and maximum values in the y-axis according to the device data.

Show grid – enables data grid lines. The shortcut "G" can also be used for this.

NOTE: This may slow down the plotter's performance.

Pause – pauses the plotter. The shortcut "P" can also be used for this.

Clear – clears all data points from the plotter. The shortcut "C" can also be used for this.

FFT – turns on the Fast Fourier Transform (FFT) functionality of the plotter. More information can be found in section 4.4.1.6.

Data range – this slider bar adjusts the amount of data points shown in the plotter.

NOTE: At high data rates, the slider area of Data Range may flash red, indicating it is unable to draw all the samples received. The data will be averaged in order to fit into a screen.

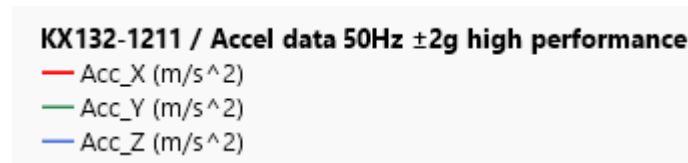
The Plotter has dynamic buttons in order to show/hide data channels within the used device stream. For example, for KX132-1211 sensor, for all data streams that start with ADP (Advanced Data Path), there are two stream buttons Advanced Data Path and Accelerometer that can be toggled to show / hide ADP and Raw Output Data streams respectively (Figure 39).



Figure 39. RoKiX Windows GUI Stream Selection

4.4.1.1. Raw data

When raw data is disabled, SI units for data streams are visible:



NOTE: If the full scale range of the device has been changed in the Stream modify mode (4.4.3.1), e.g. g-range of the accelerometer, then the Raw values displayed in the plotter view and in the Digital Output view channel will be incorrect.

4.4.1.2. Zooming

You can zoom in and out using the mouse scroll button or right mouse button + CTRL.

NOTE: When zooming and “Auto scaling” is enabled, the Plotter will no longer perform auto scaling. In order to re-enable “Auto scaling” after zooming, Auto scaling button must be enabled again.

4.4.1.3. Pausing

You can pause with the “Pause” – button or with the shortcut “P”. The Plotter can also be paused to a certain a position using the left mouse button.

4.4.1.4. Moving

The position of the Data axis (y-axis) can be moved up and down using the right mouse button.

4.4.1.5. Clearing

You can clear all visible data points from the plotter with the “Clear” – button.

4.4.1.6. Frequency analysis

The Plotter also has an FFT (Fast Fourier Transform) functionality to show frequency data. The sub-channel view can be used to show only the wanted sub-channel frequency graph(s) and further narrowed down to any axis of choice if desired. In Figure 40 two input frequencies 300 Hz and 600 Hz are applied to the KX132-1211 sensor configured with a 200Hz - 400Hz band pass filter, and 1600Hz Output Data Rate. The plotter is configured to show Advanced Data Path only stream for Z-axis (ADPZ).

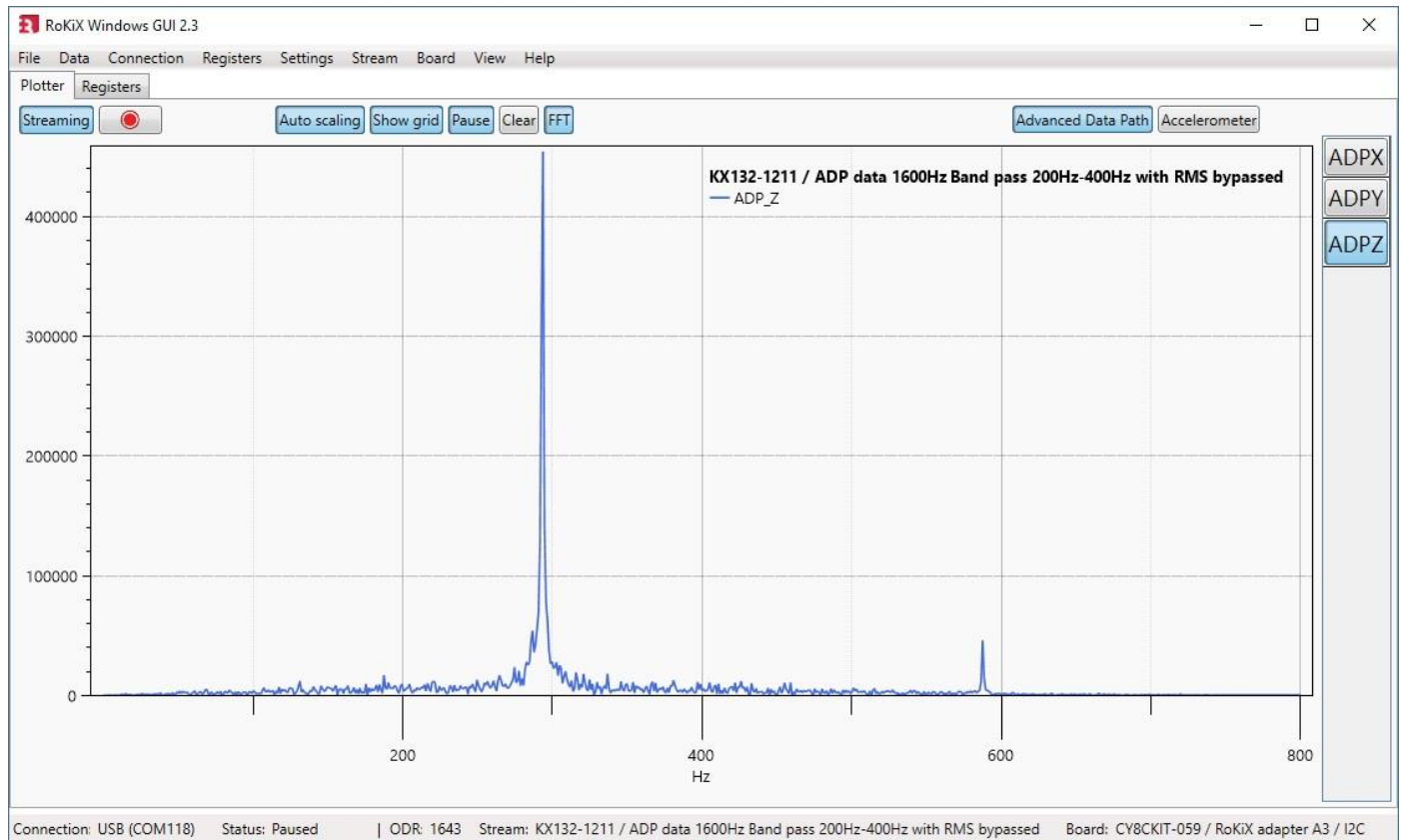


Figure 40. Plotter view with FFT functionality

NOTE: The x-axis range for the plot starts from 0 Hz and ends in ODR/2 Hz . It is automatically adjusted if the ODR is modified.

4.4.1.7. Advanced Data Path (ADP)

Advanced Data Path (ADP) is a special ASIC level functionality available only for KX132-1211 / KX134-1211 accelerometers, which consists of three blocks including a configurable second order low-pass filter, first order low-pass or high-pass filter, and an RMS calculation engine. A user can configure each stage of the ADP individually or they can be bypassed with the corresponding register settings using the register editor. Output of the ADP engine can be monitored with the Plotter by viewing Advanced Data Path channel.

To review the ADP functionality further, consider an example shown in Figure 41. In this example, the selected stream for KX132-1211 sensors is ADP data & WUF/BTS Detection 1600Hz Band pass 200Hz-400Hz with RMS. This stream, which is available for SPI communication board configuration only, has the Advanced Data Path configured to band-pass filter topology with signal bandwidth set from 200Hz to 400Hz. The output is then routed to the RMS engine and then to the Wake-Up/Back-to-Sleep engines. The wake-up threshold is set to 48 counts as indicated by the reference blue dotted line, and the Wake-up counter is set to 5 counts. When raw data (4.4.1.1) is disabled the selected data stream will scale ADP output directly to values (unit is WUF/BTS Threshold). Reference line is also moved to the wake up threshold level (48) for easier comparison of the threshold value and ADP output value (4.3.8.3). Also Wake up pop-up window is enabled (0). The pop-up window will appear when wake-up event is active e.g. when signal within band pass frequency range is present.

If there is a need to change the threshold values then proper threshold values can be seen from the plotter view and then written to sensor using register editor (see section 4.4.3.1 for details). A separate on-line tool for defining ADP settings is provided at <http://rohm-data-logging.appspot.com/adp/>.

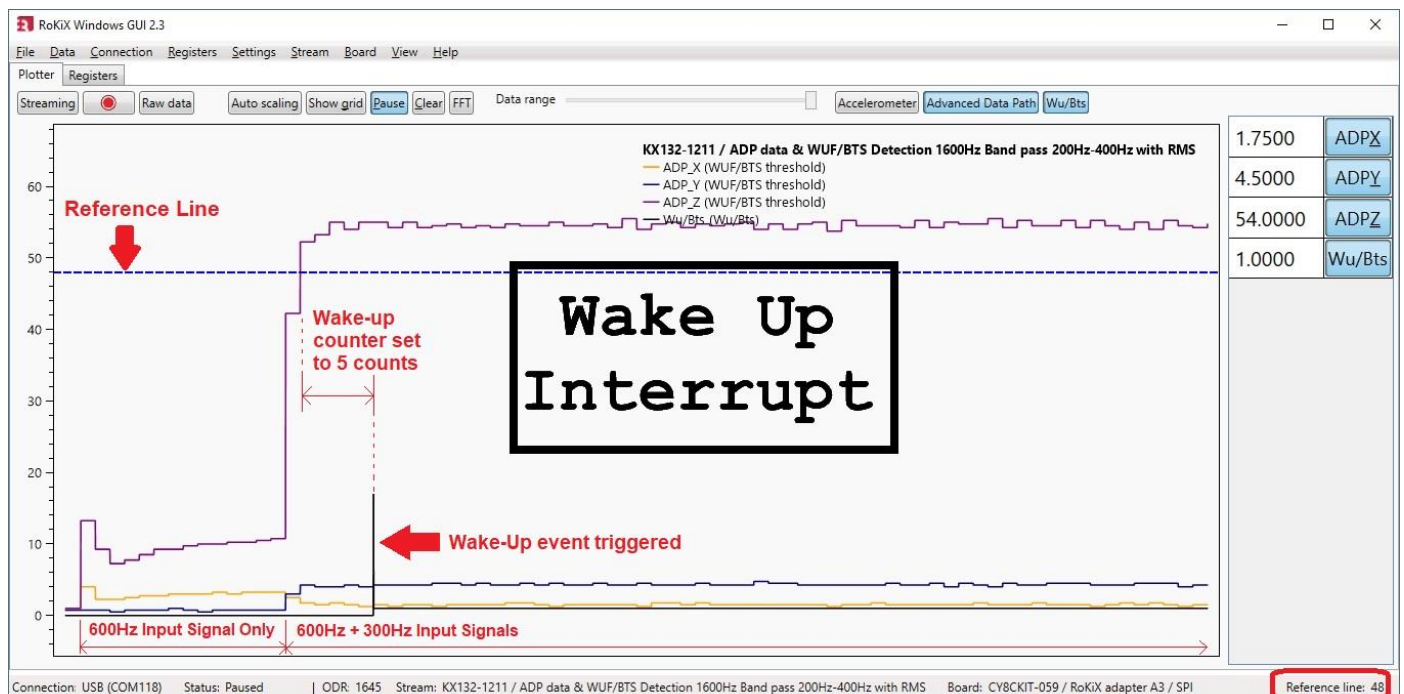


Figure 41. The plotter view with ADP & WUF data stream.

4.4.2. Angle Calibration – Tab

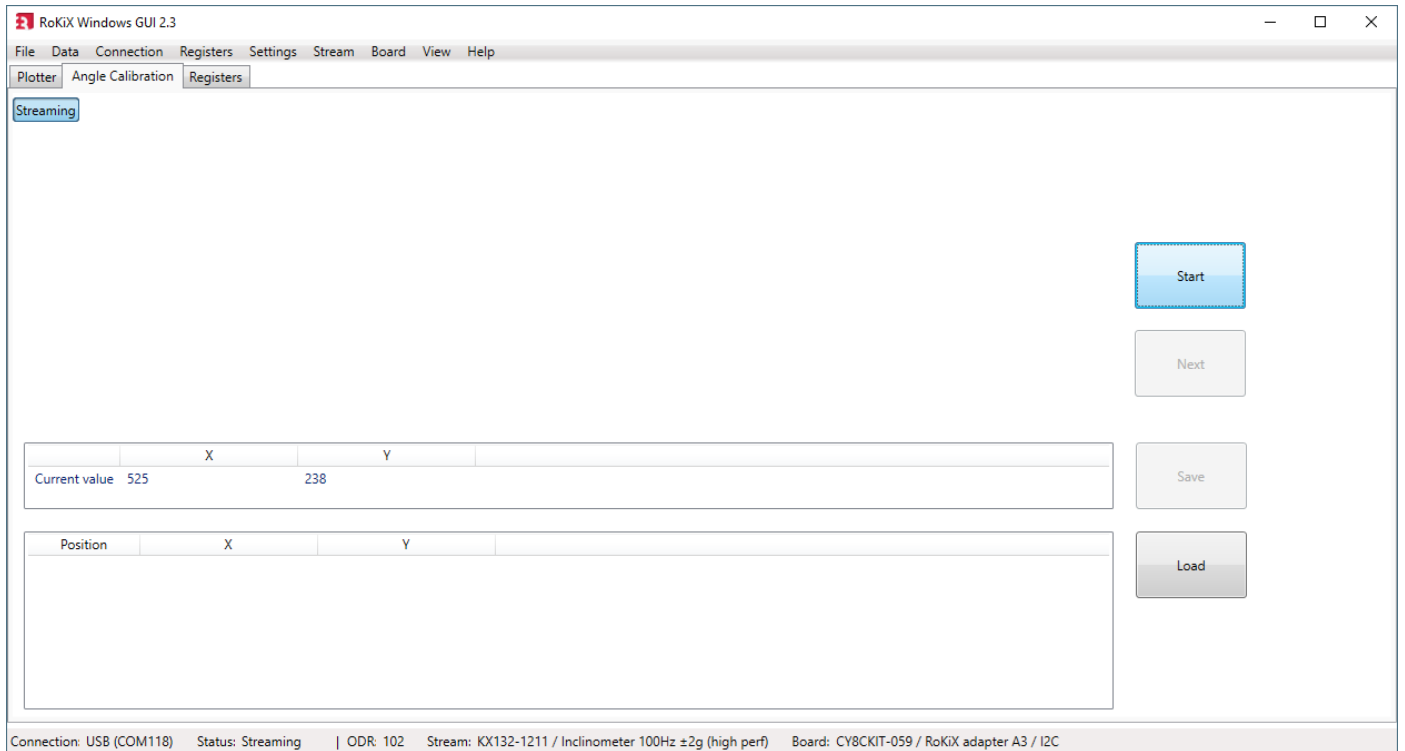


Figure 42. Angle Calibration Tab

The Angle Calibration tab is visible when selected stream name includes Inclinator name in it. For example, for KX132-1211 there is a stream called Inclinator 100Hz ±2g (high perf) as shown in Figure 42. The Angle Calibration tab allows the user to calibrate the sensor for evaluation purposes with *RoKiX Windows GUI*. Note that calculated parameters are not saved to the sensor itself. The user can start a new calibration by clicking the start button while streaming.

NOTE: If *RoKiX Windows GUI* detects that calibration has been done on the same calibration position twice in a row, the Calibration Error window will be shown and user will be required to restart the calibration procedure from the 1st calibration procedure (Figure 43).

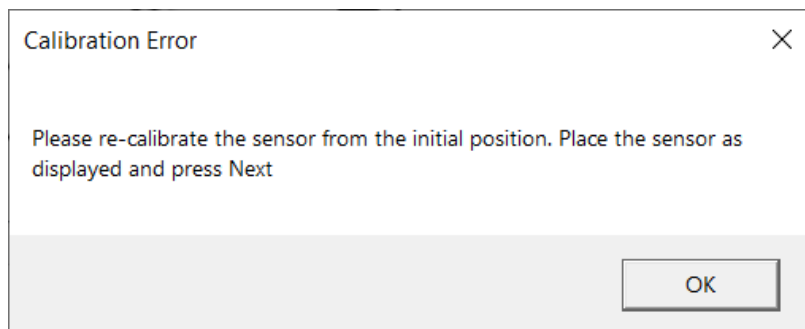


Figure 43. Angle Calibration Error Message

1. Put the device in the 1st calibration position as displayed in the orientation diagram (Figure 44).
 - The current values (in counts) for X and Y axes are displayed in the table right under the orientation image.

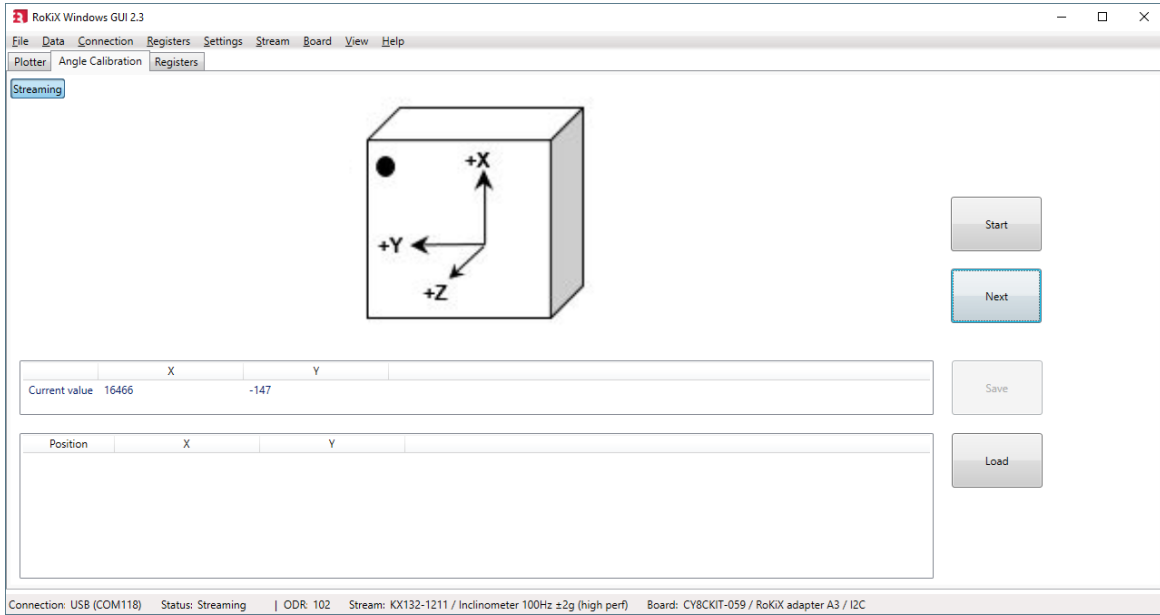


Figure 44. Angle Calibration. Position - 1

2. Click Next and hold the device still until the diagram changes to the 2nd position (Figure 45)
 - The last current value for position 1 is now stored and shown in the 2nd table.

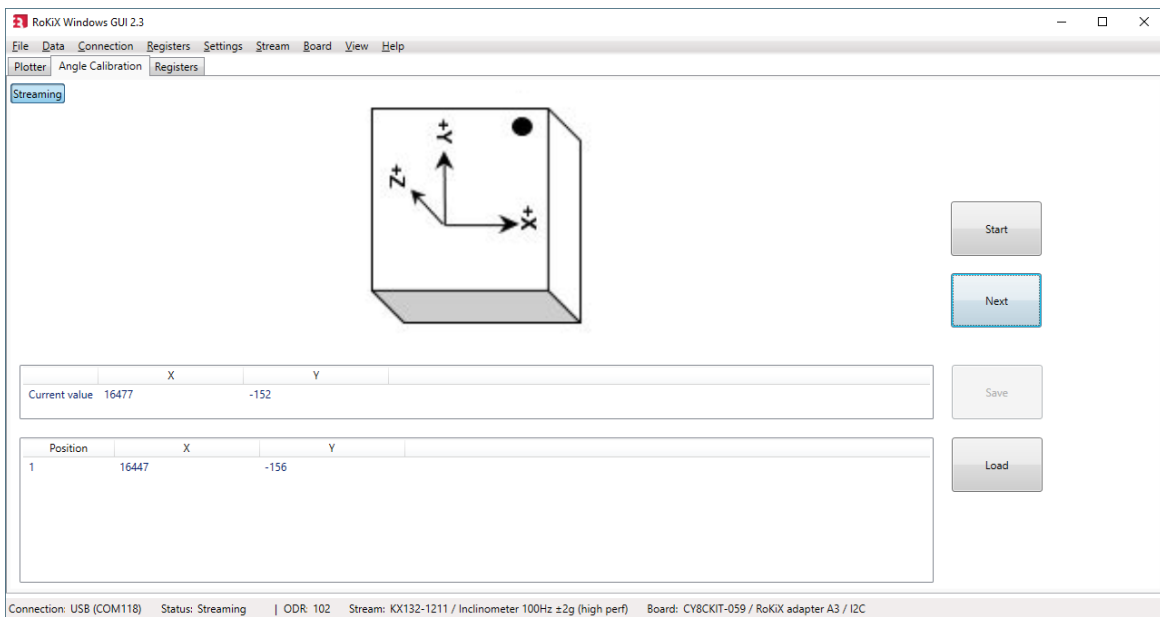


Figure 45. Angle Calibration. Position - 2

- Click Next and hold the device still until the diagram changes to the 3rd position (Figure 46)

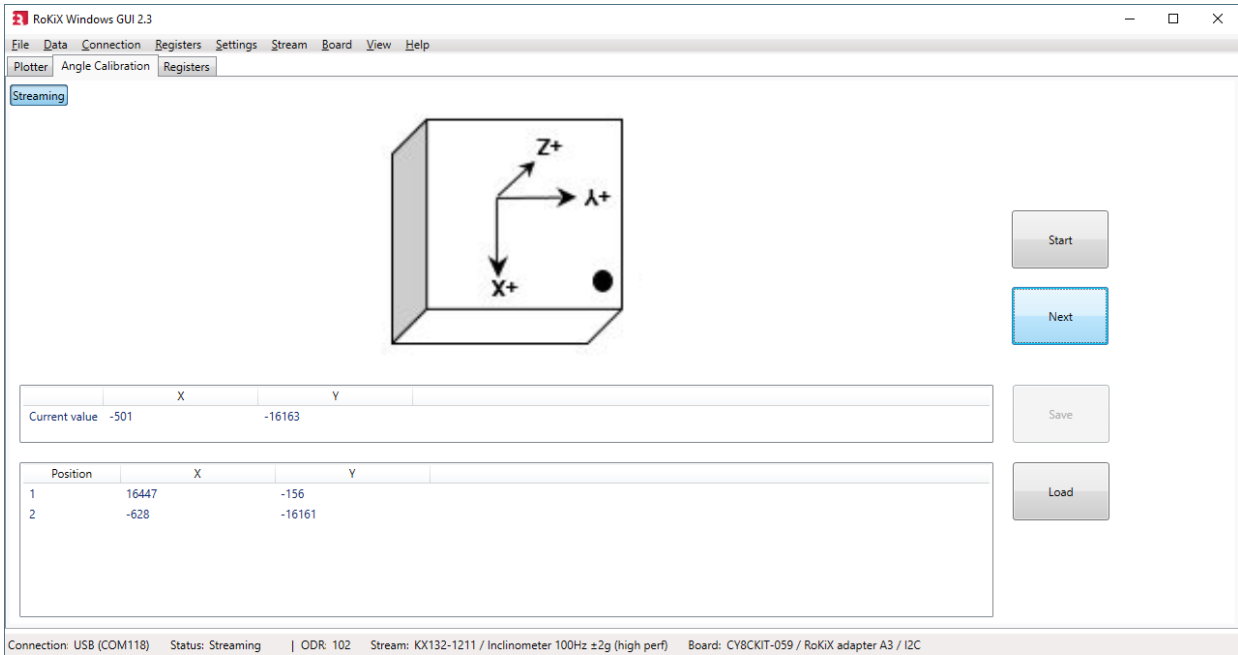


Figure 46. Angle Calibration. Position - 3

- Click Next and hold the device still until the diagram changes to the 4th position (Figure 47)

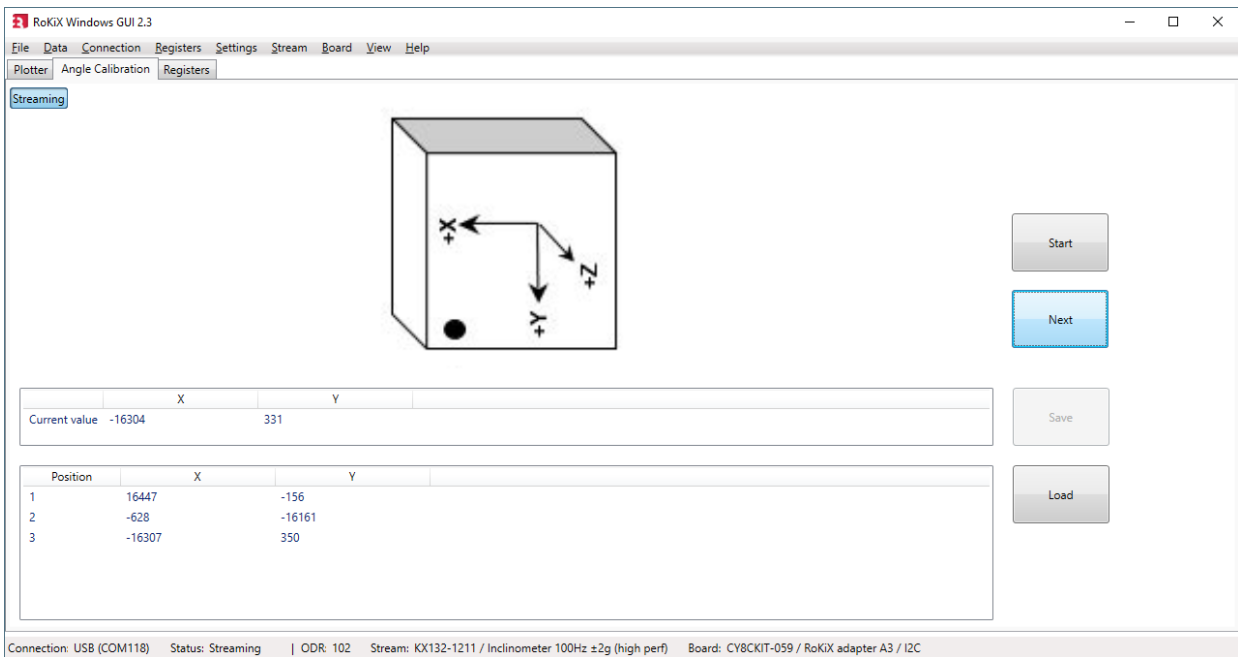


Figure 47. Angle Calibration. Position - 4

After the 4th calibration position is finished, the orientation figure returns to the 1st position. The InclInometer calibrated status will be shown in the right corner of the status bar (Figure 48), and a small desktop notification message will be shown for a few seconds to indicate that calibration procedure is done (Figure 49).

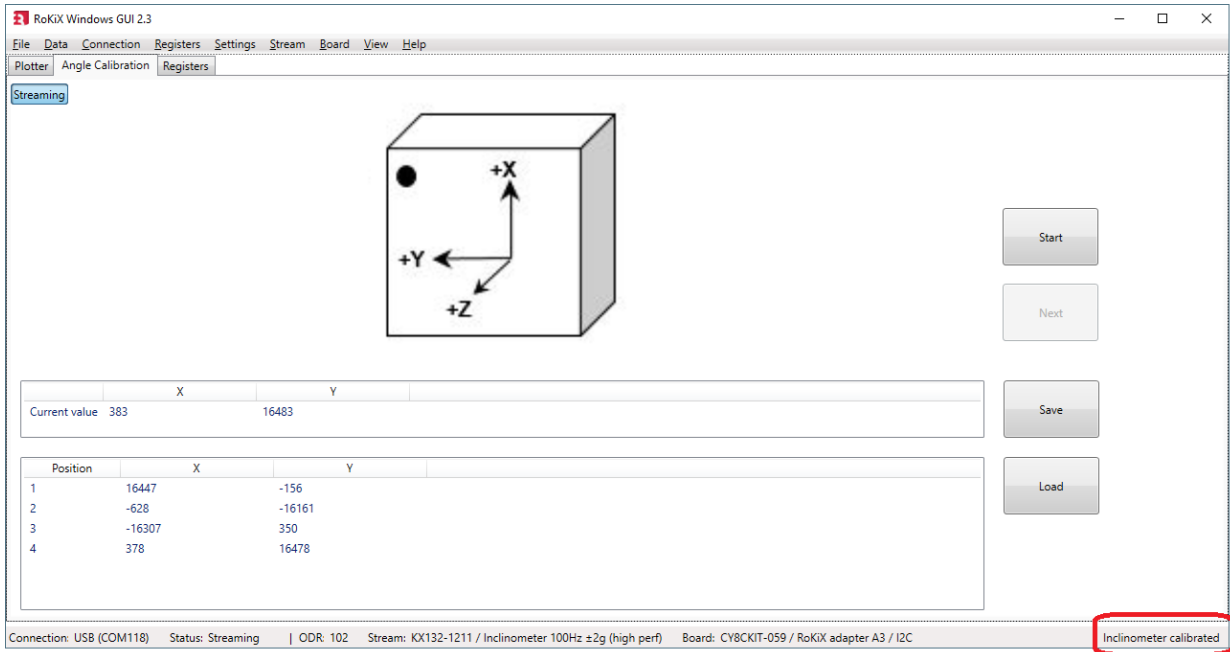


Figure 48. Angle Calibration Completed. Back to Position 1.

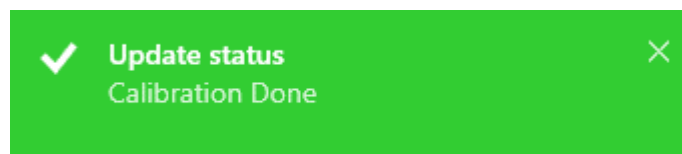


Figure 49. Calibration Done notification window

At this point, if the user is satisfied with the calibration parameters, they can be saved to an individual .json file that could be loaded next time when the *RoKiX Windows GUI* restarts. These calibration parameters will now be used for angle calculations.

The plotter view now shows the Accelerometer (X, Y, Z) acceleration data stream as well as the angle stream. The default units for angle values are radians. To display the angle information in degrees, click on Raw data button located above the plotter view (Figure 50).

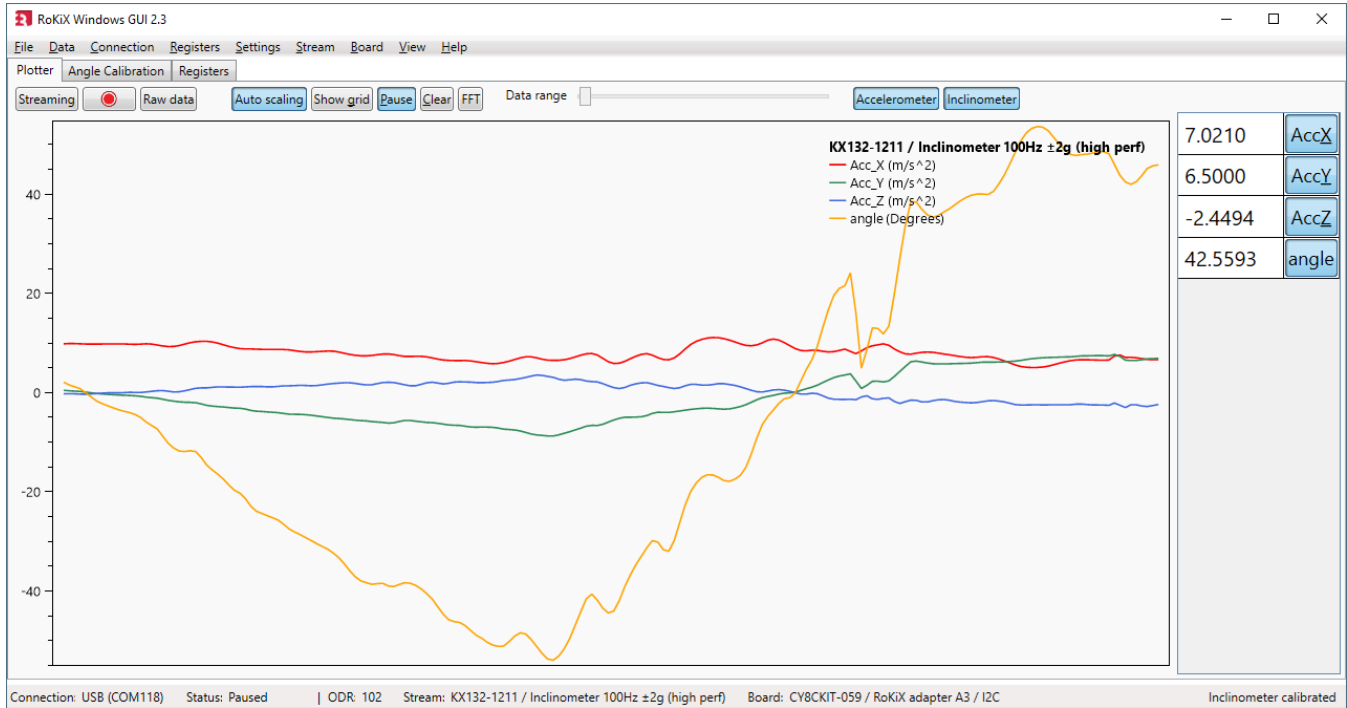


Figure 50. Plotter view with KX132-1211 inclinometer stream.

4.4.3. Registers – Tab

The register editor tab can be used for reading and writing device register values (Figure 51). The first step to use the register tab is to load the correct device register XML file, which would normally be just the name of the device, e.g. KX132-1211.xml. All the registers xml files can be found in the following directory:

\\Documents\RoKiX\RoKiX-Windows-GUI\SensorRegisters

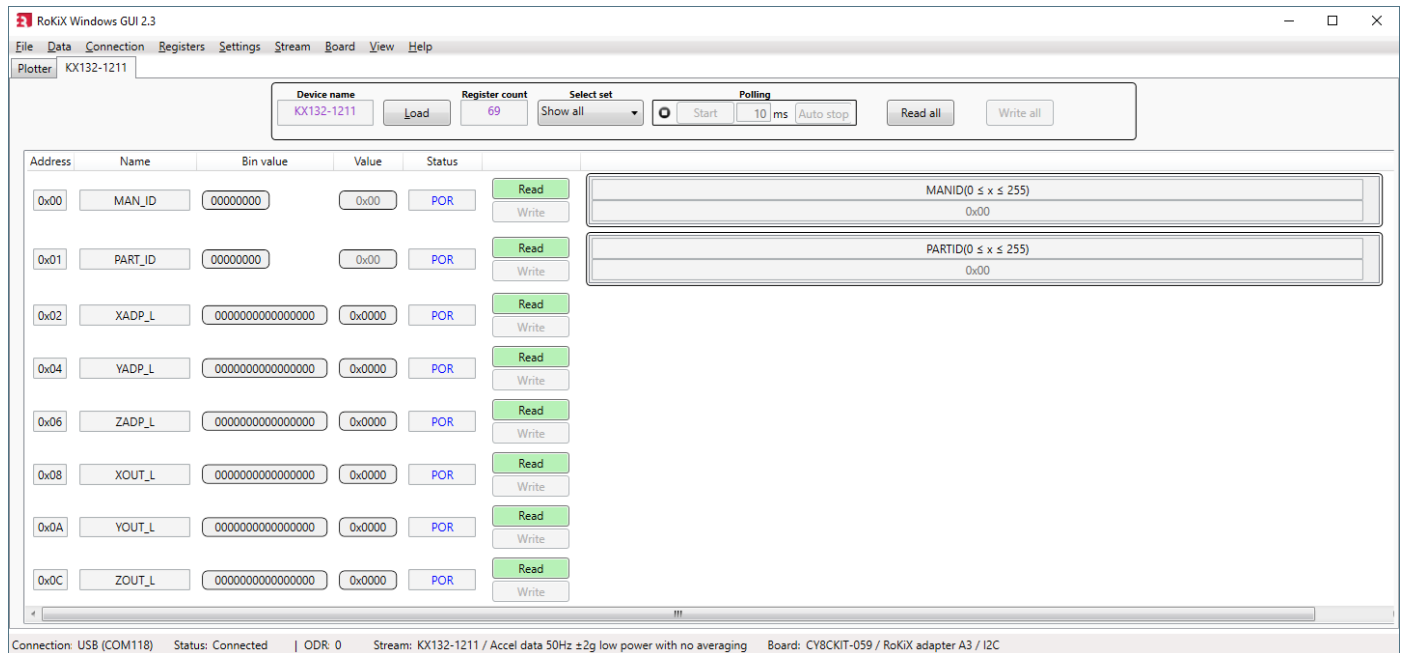


Figure 51. RoKiX *Windows* GUI register editor tab

NOTE: When the device register XML file is loaded, the name of the tab will change to one of the following:

- <device name>. For example: KX132-1211. This means the Register editor is entered while the Streaming is not active.
- <device name> - NOT FOUND. For example: KXTJ3 - NOT FOUND. This means the selected device is not supported by the currently selected Board configuration from the Board menu (4.3.7).
- <device name> - MODIFYING STREAM. For example: KX132-1211 - MODIFYING STREAM. This means the Register editor was entered while the Streaming was enabled (4.4.3.1).

NOTE: When a new device in the register tab is opened, the register's content shows the POR values of the registers as defined in the device register XML file. In order to see the current registers values, press the Read All button to read all the registers at once or the Read button, to read the value of an individual register.

4.4.3.1. Stream modify mode

When the Register editor tab is selected while Streaming is enabled, the register editor will enter into a data stream modify mode as indicated by the name of the Registers tab (e.g. KX132-1211 - MODIFYING STREAM) (Figure 52). This mode offers the user a way to change the registers which will have an effect on the data stream itself. When desired register changes have been made and the user switched the Plotter tab, Streaming will be re-enabled with the modified register values. For example, this makes it easy to change the ODR or data range.

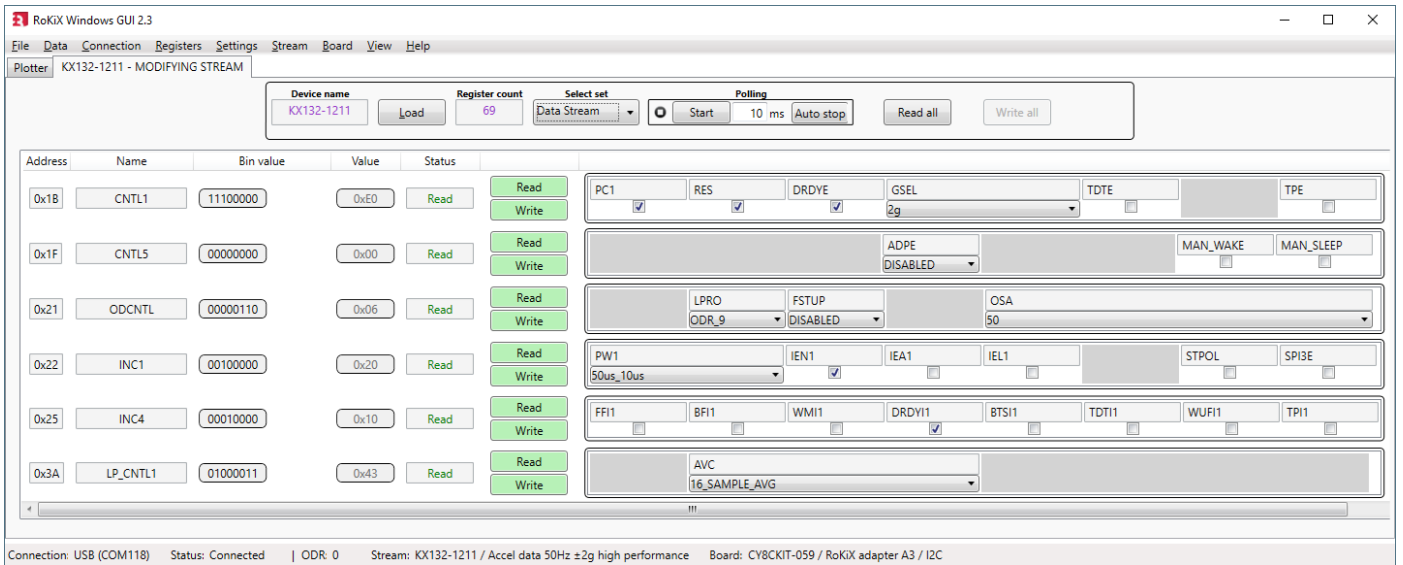


Figure 52. Register editor view in the Stream Modify Mode

NOTE: Streaming and logging is automatically paused once the Register editor tab is selected.

NOTE: Some devices require that the power control bit (PC1) be set to 0 before changing the register values. Otherwise register value changes are not applied. After registers are edited, the PC1 bit must be set back to 1 to enable the device again. When switching to the Plotter view, do not press the Streaming button again. When Streaming button is disabled and enabled again, default registers values as set in the stream configuration will be written to the device!

NOTE: When a full-scale range of the device is changed (e.g., g-range of the KX132-1211 accelerometer), the SI units (m/s²) will not be shown correctly in the plotter view and the Digital Output sub-channel view (4.4.1.1). The counts value is not affected.

NOTE: When registers have been changed, "(MODIFIED)" text will be shown after the stream name in the Plotter View area (Figure 53).

NOTE: When ODR value has been changed through the register editor, the "ODR has not reached the target value" pop-up message may be shown because the new real-time ODR is significantly different from the original ODR value defined in the stream (Figure 53). The ODR pop-up message can be disabled through the View menu (4.3.8.6).

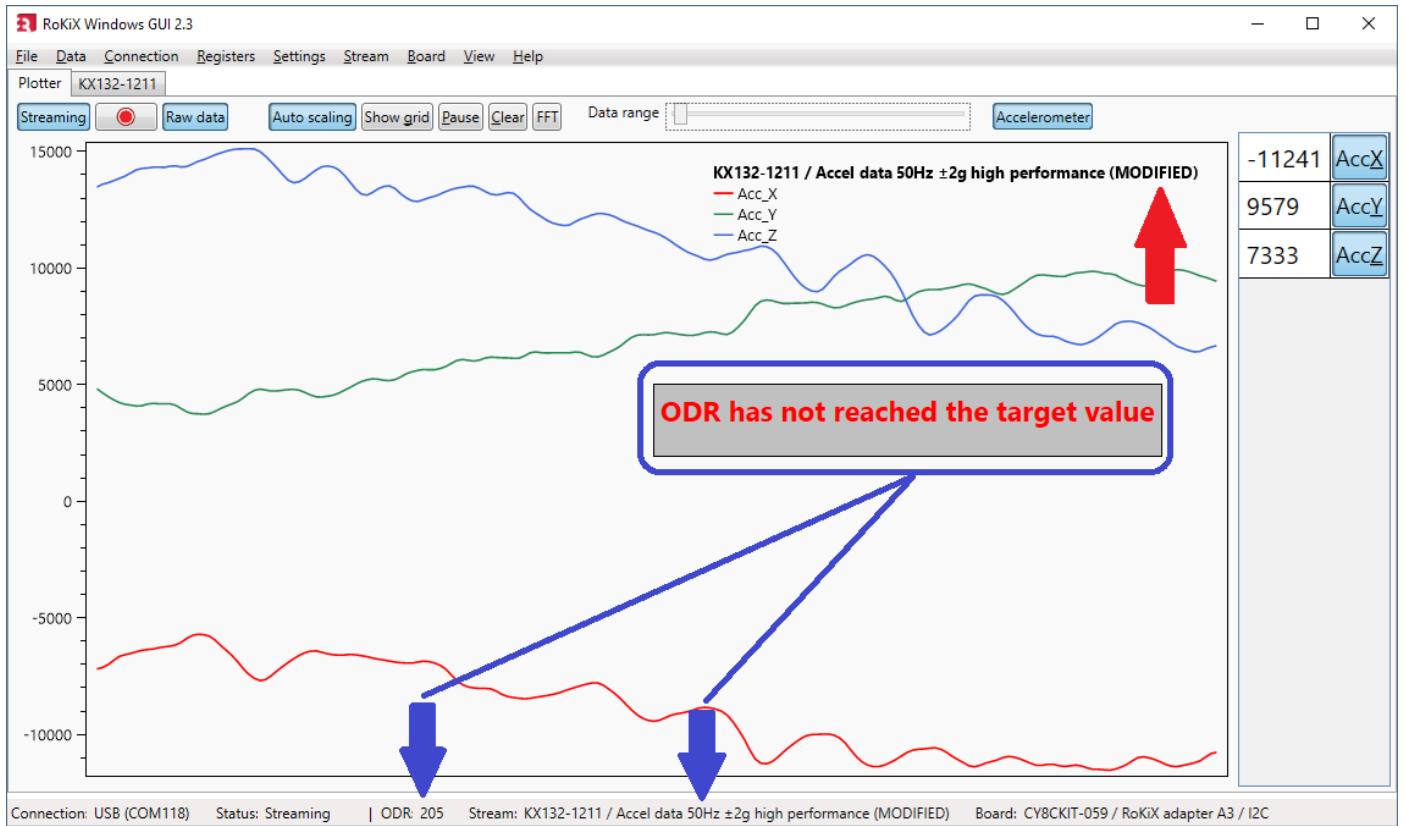


Figure 53. The Plotter View After Modifying the Register Values

4.4.3.2. Updating register value

The graphical user interface provides an intuitive way to update the values of certain bits and entire registers.

Single Bit Function

When individual bit defines a certain function, the value of the bit can be changed by checking / unchecking the check box (e.g., PC1 bit of the KX132-1211 that sets the part in operational or standby mode). When selection is made, click on Write button to store the setting (Figure 54).

Multiple Bits Function

Some bits are grouped together if they make up one setting and have a predefined function for each combination of bits. For example, the Full-Scale range of the KX132-1211 accelerometer is defined by the value of GSEL1 and GSEL0 bits. These two bits are grouped together in the register editor as GSEL and user can select the actual function, instead of modifying the individual bits. When selection is made, click on Write button to store the setting.

Reserved Bits

If the bit is grayed out, it means it is reserved and thus the register editor does not provide a way to modify it to avoid an unexpected behavior. For example, bit 1 in CNTL1 register is reserved (Figure 54).

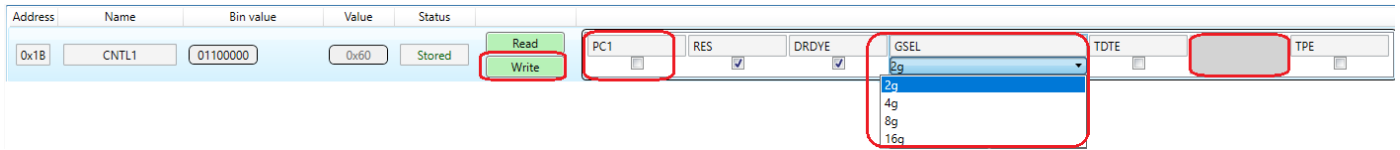


Figure 54. CNTL1 register of KX132-1211

Writing to an 8-bit register

Certain functions of the device can span across the entire 8-bit register and their value is between 0 and 255 (Figure 55). Other functions can take less 8-bit. For example, the FTDH value is a 5-bit value and is between 0-31 (Figure 56). In order to change the value, user can simply enter the value either as a decimal or as a hex value (the hex format is 0xFF) and press the enter. Once value is written, the "value" field of the register gets updated with the new value. It is possible to read the value back. Also, if "Register writes events" function is enabled in the View menu (4.3.8.2), the write information will be shown in the events window.

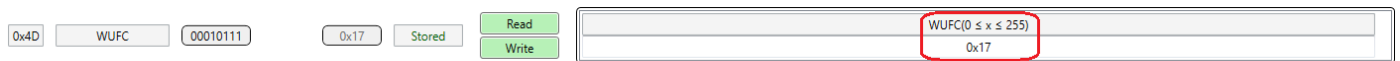


Figure 55. WUFC 8-bit value (0-255)



Figure 56. FTDH 5-bit value (0-31)

Changing device settings that spans over two registers

Some functions of the device may span more than 8-bits and are thus split between 2 or more registers. For example, the WUFTH function of the KX132-1211 is an 11-bit value (0 to 2047). The lower 8 bits are stored in the WUFTH register (0x49), and the upper 3 bits in the BSWUFTH register (0x4A) (Figure 57).



Figure 57. WUFTH 11-bit value (0-2047) spanned across two registers

To simplify register update, the user can write the entire value to the lowest byte register (e.g., a value from 0 to 2047 to WUFTH_L register), and the register editor will update both registers accordingly. For example, to set the WUFTH value to 950 (0x03B6), write 950 to the WUFTH register 0x49, and the register WUFTH (0x49) will store the value 0xB6, while the register BSWUFTH (0x4A) will store the value 0x03.

NOTE: The lowest byte register will always show the full value in the *bit field*. However, the 8-bit *register value field* will show the actual 8-bit value of the register.

NOTE: When write is done to the lowest byte register, the 8-bit *register value field* of the lowest byte register will be updated to show the new value automatically. However, the 8-bit *register value field* of the upper register(s) that contain the remaining bit(s), will not be updated automatically to display the new value until the Read button is pressed. Also note, if Register write events is enabled (4.3.8.2), only the write to the lowest register is shown in the event window.

Changing device settings that occupy two or more registers

While limited, there are some device settings that not only span over few registers, but also use those registers exclusively for this setting. In those cases, the Register editor hides all but the lowest byte and gives the user the ability to update the entire value from a single write to the lowest byte register. Consider the ADP_F1_BA settings of the KX132-1211 accelerometer. This setting is a 23-bit value (0-8388607) and spans across registers ADP_CNTL4 (0x67) – ADP_CNTL6 (0x69) (Figure 58).

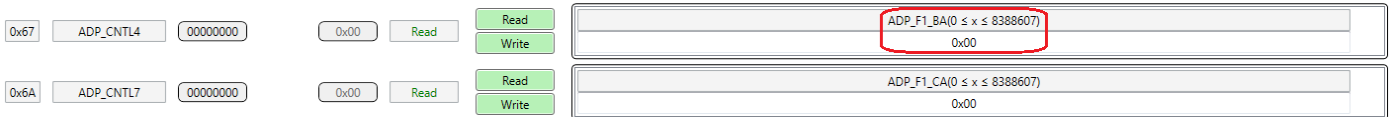


Figure 58. 23-bit value ADP_F1_BA is updated from a single 8-bit register

The user has the ability to update the entire 23-bit value through write to the ADP_CNTL4 register that stores the lowest 8 bits of the 23-bit value.

NOTE: The lowest byte register will always show the full value in the *bit field*. However, the 8-bit *register value field* will show the actual 8-bit value of the register.

4.4.3.3. Register sets

The list of registers shown in the Register tab is defined by the register set files. Each set file lists the register addresses of the registers in the set to be later displayed. The *RoKiX Windows GUI* comes with several pre-defined registers set that can be selected from the "Select set" drop down box (Figure 59). The default register set available for all devices is "Show all" that lists all the register supported by the current device. All other register sets are defined in the corresponding register set files.

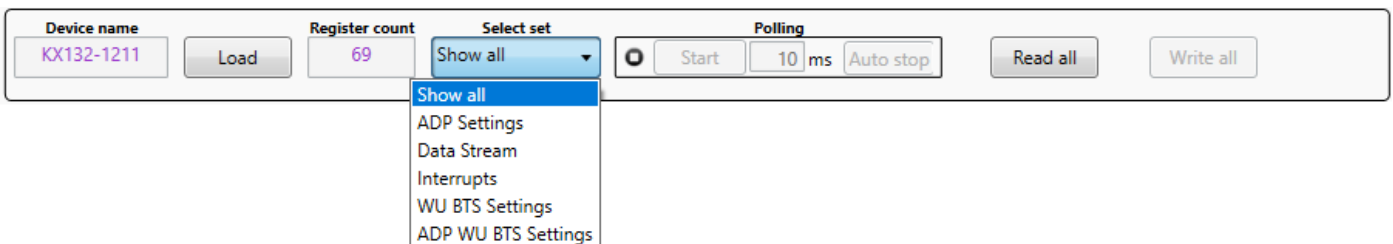


Figure 59. Select set Drop Down Menu

The register sets are very useful feature of the *RoKiX Windows GUI* as it allows users to visually see only the registers of interest grouped together. Consider the Data Stream register set shown in Figure 60. From this set, it is possible to select all the basic configuration of the accelerometer such standby/run mode, power mode, ODR, full scale range, and set some basic interrupt.

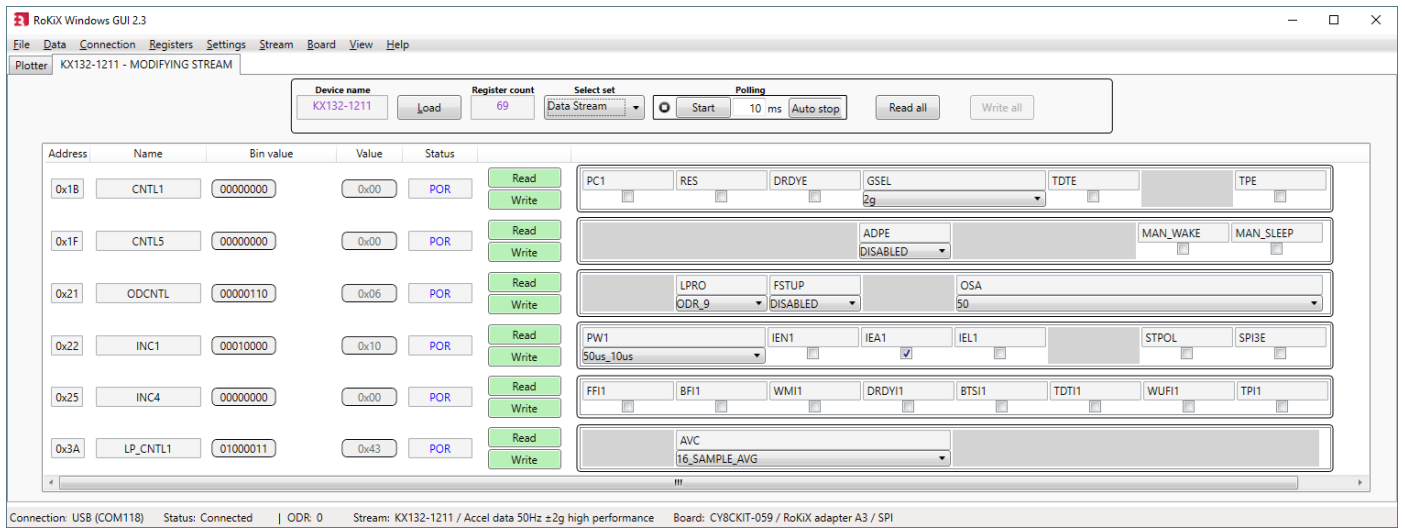


Figure 60. Data Stream Register Set

The register set files are text files.

Register Set File Location:

The *default* register set files are stored together with the stream configuration files for each board configuration. For example, for RoKiX Development Kit, the list of register sets can be found in the following two folders (one for I²C and one for SPI serial interface protocol):

`\Documents\RoKiX\RoKiX-Windows-GUI\Configuration\stream_config\board_cy8ckit059_i2c_evkv3`

`\Documents\RoKiX\RoKiX-Windows-GUI\Configuration\stream_config\board_cy8ckit059_spi_evkv3`

The custom register set files can be created and stored in the dedicated user folder where they will be loaded together with the default register sets:

`\Documents\RoKiX\RoKiX-Windows-GUI\SensorSet`

Register Set File Naming Convention:

The register set file name follows the following naming scheme:

`<device name>_<set name>_set.txt`

For example, the Data Stream for KX132-1211 sensor is called:

`KX132-1211_Data Stream_set.txt`

Register Set File Structure Convention:

The register set definition follows the following scheme :

```
<Set Name> : Reg1, Reg2, Reg3
```

NOTE: the set name can contain a space. The set name should be then followed by a colon ":" sign and HEX addresses of the registers each preceding by "0x" in front of it. The registers should be separated by commas ",", and spaces are not allowed. Any register preceded by a space would not be shown in the register view. The registers can be listed in any order and will be sorted automatically in the register viewer from lowest to highest regardless of the order in the register set file. The example of the Data Stream register set is shown in Figure 61.

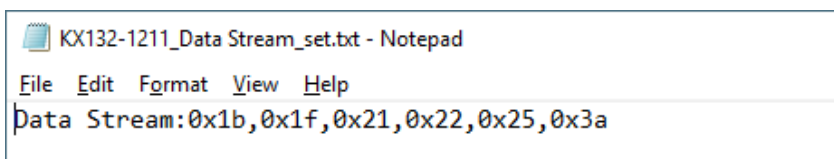


Figure 61. KX132-1211 Data Stream Register Set

NOTE: The *RoKiX Windows GUI* loads all the register sets on the start up. If changes were made to the content of the register set files or new register set files were created, the changes would be visible next time the program is loaded.

4.4.3.4. Register polling function

Register polling (i.e., reads) is a simple way to monitor the values of the registers defined in the Register sets (4.4.3.3). The Polling feature is enabled with a Start button. A delay between successive register reads can be set as well (the default delay is 10 ms). The polling can be stopped with a Stop button. Also, the polling feature has an “auto stop” – checkbox, which stops the register polling as soon as the value of any of the registers in the register set changes. The register polling continues when the Start button is pressed again. The below example shows how to monitor Wake-Up / Back-to-Sleep detection:

- Select sensor KX132-1211 by pressing the “Load” button.
- Select set “WU BTS Settings” register set from the “Select set” pull down menu.
- Press the “Read all” button.
- Uncheck the PC1 and DRDYE bit checkboxes in the register CNTL1 (0x1B) and press Write.
- Set WUFE and BTSE bit values to ENABLED in CNTL4 (0x1E) register and press Write.
- Check MAN_SLEEP bit box in CNTL5 (0x1F) register and press Write.
- Write 5 to the BTSC (0x4C) register to set the back-to-sleep counter to 100 msec.
- Write 5 to the WUFC (0x4D) register to set the wake-up counter to 100 msec.
- Check the Power Control bit (PC1) in CNTL1 (0x1B) register to enable the sensor and press Write.
- Select “Interrupts” register set from the “Select set” pull down menu
- Press the “Start” button from “Polling”

Shake the device and monitor the status of Wake-Up Function Status (WUFS) and Back-to-Sleep Status (BTS) bits in INS3 (0x18) register as well as the WAKE bit in the STATUS_REG (0x19) register. When the device is shaken above 0.5g threshold, the “Wake” bit is asserted to indicate the wake state. Once the device is left in place and acceleration drops below 0.5g, the WAKE bit will be de-asserted to indicate the Sleep mode (note, if “Auto stop” was enabled, the polling will stop on the WAKE event and needs to be restarted to see the back-to-sleep event). The WUFS and BTS bits values are changing for a brief period when the corresponding interrupt has fired and stay on until they are auto cleared through INT_REL register auto read (Figure 62).

NOTE: Streaming and logging are automatically paused when register polling is started.

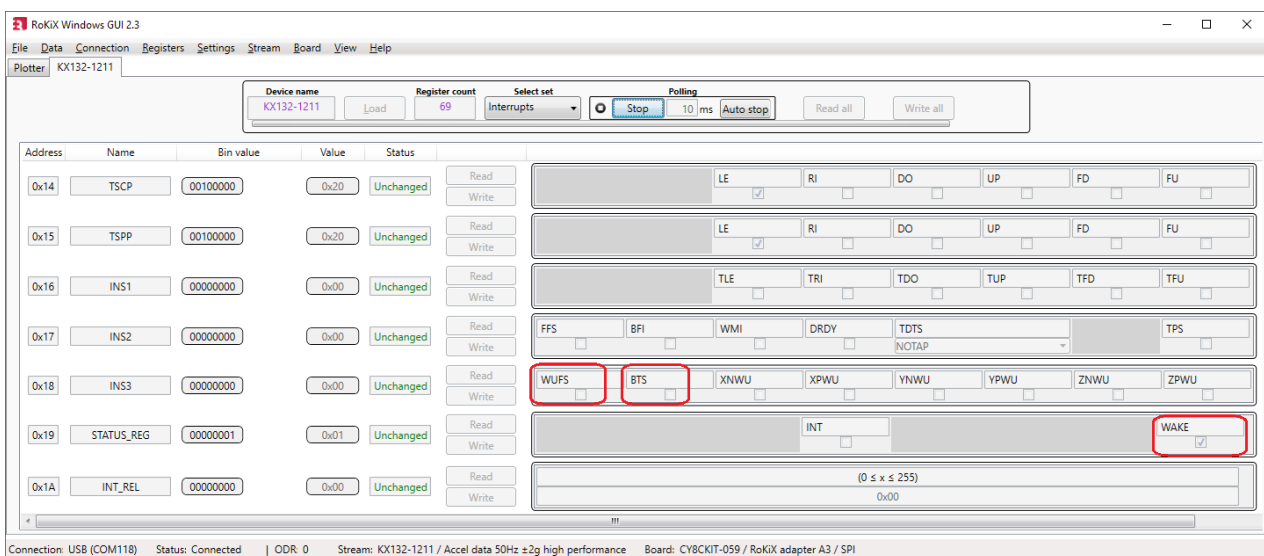


Figure 62. Wake-Up / Back-to-Sleep Interrupt Detection Register Set

4.5. User Interface - Status bar

Connection: USB (COM118) Status: Streaming | ODR: 51 Stream: KX132-1211 / Accel data 50Hz ±2g high performance Board: CY8CKIT-059 / RoKiX adapter A3 / I2C

The status bar shows the following information:

- Current connection interface (USB / Bluetooth) (4.3.3)
- COM port (4.3.5.4)
- The status of the communication (Streaming, Connected, Disconnected, No Data,)
- Real-time Output Data Rate (ODR) information
- Stream Configuration (0)
- Board Configuration (4.3.7)
- Reference line value (4.3.8.3)
- Angle calibration status for select streams (0)

NOTE: Bluetooth communication is not supported by the *RoKiX Development Kit*.

NOTE: It is normal to see a slight variation in the ODR value. Data is received at varying intervals and the ODR is calculated when *RoKiX Windows GUI* receives the data from the used connection layer.

4.6. User Interface - Pop-up windows

The application makes use of pop-up windows to notify the user about important actions. This section provides detailed information about pop-up windows.

4.6.1. No data pop-up window

“No data received” pop-up is showed when streaming has been started, but no data is received. The problem could be invalid board configuration selection or some connection problem (see section 6.5.2 for details).

No data received! Please check your board configuration and device functionality.

4.6.2. Streaming pop-up window

Streaming pop-up window is shown in the Plotter view to notify the user about data stream enabling. Streaming can be enabled with the specific “Streaming” – button, from Data/Streaming – menu or with the shortcut “CTRL + S”. It also possible to enable Automatic streaming (4.3.5.3)

Please enable streaming to activate Plotter movement!

4.6.3. ODR has not reached the target value pop-up window

ODR has not reached the target value

The ODR warning pop up window is shown anytime when the real time Output Data Rate (ODR) as measured by the *RoKiX Windows GUI* is significantly different from the nominal ODR set in the Stream (Figure 63). One way this can happen when the selected ODR is greater than 3200Hz and the interface protocol is I²C and not SPI. It can also occur when the USB cable is either damaged or of a low quality. It can also occur when the ODR value has been modified through the Stream Modify Mode (4.4.3.1).

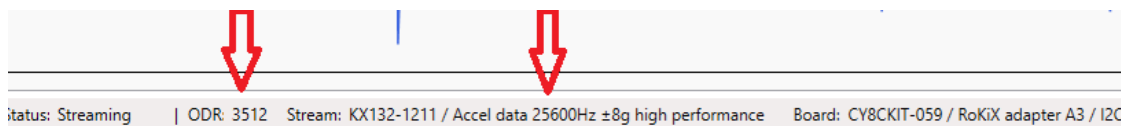


Figure 63. Real-time ODR is significantly different than the nominal ODR

4.6.4. Wake up pop-up window



Figure 64. Wake-Up Interrupt pop-up window

The “Wake up pop up window” (Figure 64) is shown for selected wake-up / back-to sleep event detection streams when the “Show wake up pop up window” sub menu item is enabled (4.3.8.4). When the wake-up event is detected, the pop-up window is displayed in the plotter (Figure 65).

NOTE: It is recommended to use SI units in Plot in order to better visualize the Wake-up Direction line changes.

NOTE: The Wake-Up Interrupt pop up window that is displayed can be replaced with any custom image. The new image file must be called `wakeup.png`, have an approximate size of 380 x 190 pixels, and be placed in the RoKiX Windows GUI Resource folder:

`\Documents\RoKiX\RoKiX-Windows-GUI\Resources`

instead of the original file.

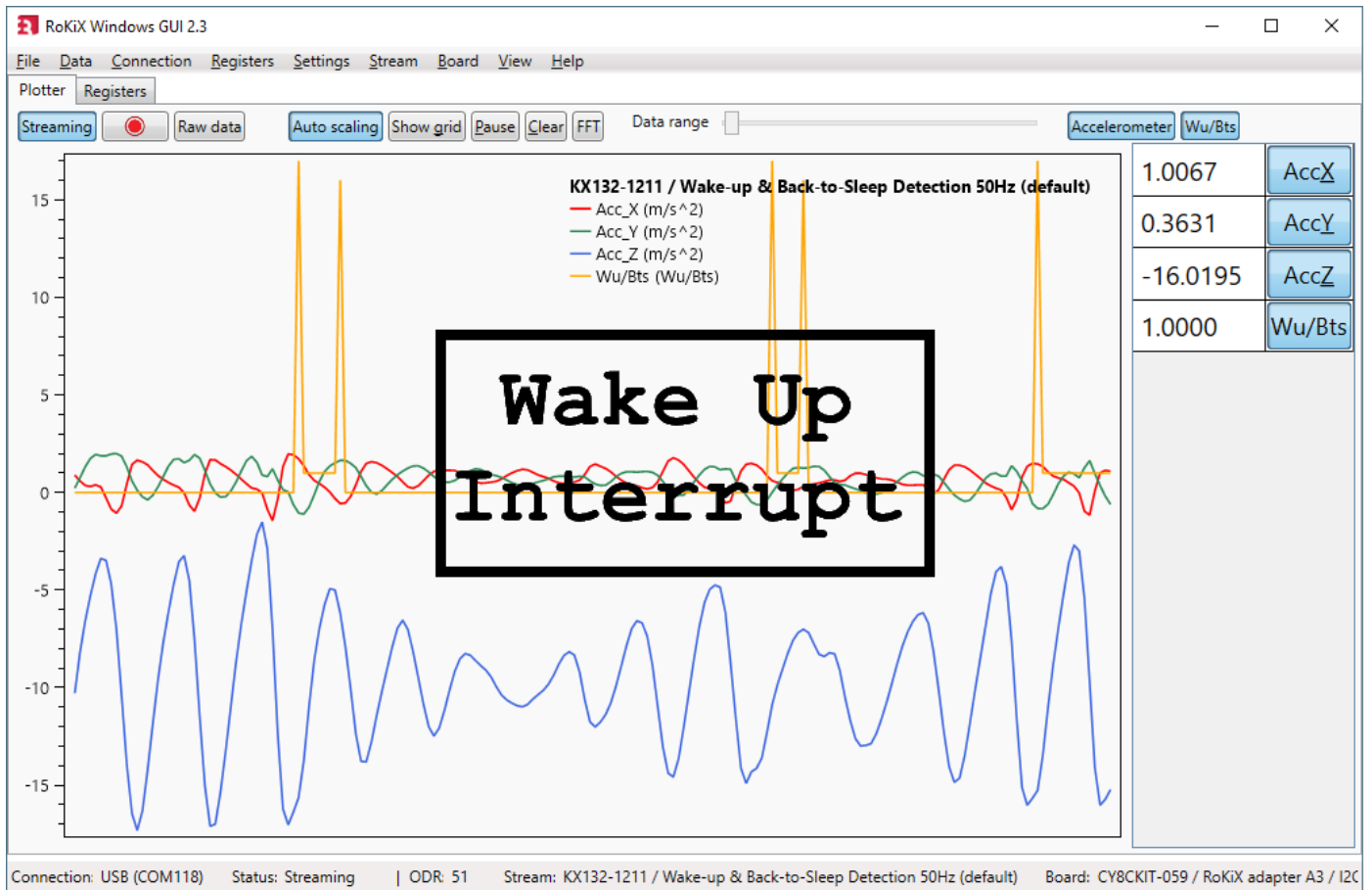


Figure 65. The plotter view with wake up triggered.

4.7. Shortcuts

The RoKiX Windows GUI has many keyboard shortcuts:

CTRL + L	Enable/disable logging
CTRL + S	Enable/disable streaming
ALT + R + L	Load sensor register definition file
CTRL + R	Reset used connection and data streaming (disconnect and connect when having connection problem). Re-enable the Streaming when connection is established.
CTRL + E	Show events view
CTRL + V	Show subchannel view
CTRL + D	Show digital output in sub channel view (works only if subchannel view is enabled)
CTRL + C	Clears the current points in plotter view
G	Shows the grid in the plotter
P	Pause plotter
CTRL + W	Hide/show wake up interrupt pop up window
SHIFT + L	Hide/show reference line

5. Getting Started with *RoKiX Python CLI*

5.1. Introduction

The *RoKiX Python CLI*, which is also a part of the *RoKiX IoT Platform Client Software*, offers versatile access to the device register level functionality. Simple and minimalistic demonstrator applications are provided for testing a variety of device features. These will help to test and evaluate device functionality and provide needed information on how to implement device driver SW.

5.2. Installation for Windows OS

The *RoKiX IoT Platform SW* Windows installer includes the *RoKiX Python CLI* and it will be installed to the following directory: `\Documents\RoKiX\RoKiX-Python-CLI`. Additionally, the installer creates start menu items for the *RoKiX Python CLI* (Figure 66).

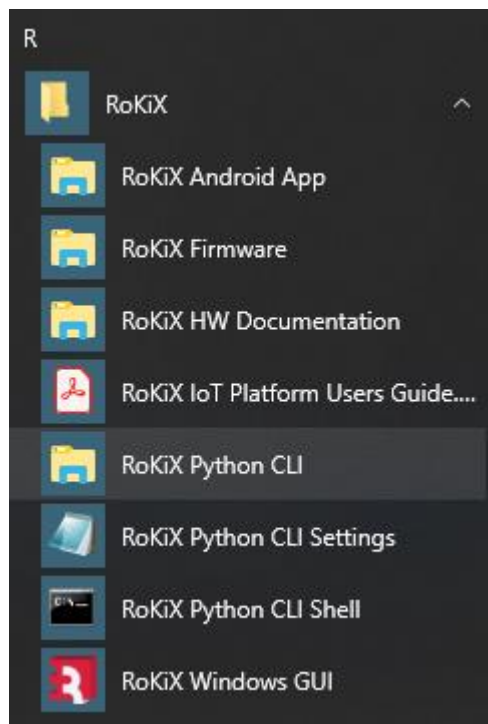


Figure 66. RoKiX Start Menu

Section 5.4 describes the installation processes of the necessary Python environment. The following URL will always point to the latest version of the *RoKiX Python CLI*:

<https://www.rohm.com/support/accelerometer-evk-support>

5.3. Installation for Linux and OS X

On the Linux system, the latest version of the *RoKiX Python CLI* can be downloaded with the following URL:

<https://www.rohm.com/support/accelerometer-evk-support>

Currently, Bluetooth usage with the *RoKiX Python CLI* is not supported when running on Apple OS X. Connecting via USB is the default.

See section 5.4 below for installation of the Python environment.

5.4. Python Set Up

A [Python](#) interpreter is needed to run the *RoKiX Python CLI*. Currently, Python versions above 3.0 are recommended for use (e.g., 3.7.1). As usual, a separate Python environment is recommended to bring into use, the instructions below will guide you through the required installation procedure.

5.4.1. Windows installation

For your Windows installation, please download the Anaconda distribution of Python:

<https://www.anaconda.com/download/>

and follow the instructions:

<http://docs.anaconda.com/anaconda/install/windows/>

After the installation is complete, please create a new virtual environment:

```
conda create -n rokix_env python
```

Take the `rokix_env` environment into use,

```
activate rokix_env
```

Navigate to the *RoKiX Python CLI* folder (replace [username] with an actual username used for Windows login)

```
cd C:\Users\[username]\Documents\RoKiX\RoKiX-Python-CLI
```

With Windows OS - if automatic USB serial port detection is used in the *RoKiX Python CLI* (by default this feature is active in `rokix_settings.cfg`, refer to section 5.5.1), the following modules (dependencies) are also needed:

```
python -m pip install -r requirements_windows.txt
```

Please verify the USB driver is properly installed before proceeding further (3.1.3).

5.4.2. Linux installation

On Ubuntu the Python virtual environment is installed as follows:

Create a virtual environment

```
python3 -m venv rokix_env
```

A new virtual environment is activated with the command

```
source rokix_env/bin/activate
```

and the required Python dependencies are installed with

```
python -m pip install -r requirements_linux.txt
```

On some Linux system's (e.g. Ubuntu 18.04) serial ports may not be accessible for the user by default. If you encounter the problem "Permission denied: '/dev/ttyACM0'" while running data loggers, please refer to section 6.4 for details.

If automatic USB serial port detection is not used, the port settings to the `rokix_settings.cfg` file are used as an example:

```
serial_port = /dev/ttyUSB0
```

Ports of the connected devices can be obtained with the following command

```
ls -la /dev/serial/by-id
```

5.4.3. OS X installation

By default, Apple OS X is missing the Python package manager "pip". It is necessary to install the required python packages. The easiest way to install pip is by opening the terminal and using the command

```
sudo easy_install pip
```

After pip is installed, please follow the instructions on the previous section for Linux install.

5.4.4. Optional installations

For both Windows OS and Linux, dependencies required by `plot.py` are installed with

```
python -m pip install -r requirements_plot.txt
```

In addition, dependencies related to the cloud-based services are obtained by running

```
python -m pip install -r requirements_cloud.txt
```

5.5. Configuration

The *RoKiX Python CLI* settings are in the `rokix_settings.cfg` file located in the root directory of *RoKiX Python CLI*. The settings file uses windows `.ini` file syntax. Semicolon “;” at the beginning of the line indicates a comment.

There are settings for both:

- *RoKiX Python CLI* framework and
- Test applications included in the *RoKiX Python CLI*

5.5.1. Connection to *RoKiX IoT Platform HW*

The most important framework level setting is board selection. It defines the board configuration (host adapter) and communication protocol interface used (e.g. I²C, SPI). The definition can be found in the section `[board]` of the `rokix_settings.cfg` file.

NOTE: The board selection is done by uncommenting the required board configuration, while ensuring the rest of the board configurations are commented out. By default, CY8CKIT-059 Prototyping Kit with I²C interface protocol is selected. Below is an example of `[board]` section of the `rokix_settings.cfg` file.

```
[board]
; Board selection

; Cypress CY8KIT059 development board with a RoKiX Adapter Board A3.
board=rokix_board_cy8ckit059_i2c_a3.json
;board=rokix_board_cy8ckit059_spi_a3.json

; Arduino Uno R3 board with a RoKiX Adapter Board A3.
;board=rokix_board_arduino_uno_i2c_a3.json

; Arduino Uno R3 board with Rohm sensor shields
;board=rokix_board_arduino_uno_evk-00x.json

; Nordic Semiconductor nRF52840-DK board with a RoKiX Adapter Board A3.
;board=rokix_board_nrf52840dk_i2c_a3.json
;board=rokix_board_nrf52840dk_spi_a3.json

; Aardvark I2C / SPI host adapter
;board=rokix_board_aardvark_i2c_usb.json
;board=rokix_board_aardvark_spi_usb.json
```

Key `bus2` settings defines the connection type for the interface 2. Default is USB. All other connection types should be commented out.

```
[bus2]
; bus2 selection for board
; NOTE: bus2=USB setting works for both USB_SERIAL and USB_AARDVARK if only one of those is
connected
bus2=USB
;bus2=BLE
;bus2=BLE_PYGATT
;bus2=USB_SERIAL
;bus2=USB_AARDVARK
```

The corresponding COM port can be written in the `rokix_settings.cfg` file:

```
[bus2]
; USB serial COM port number or 'auto' for autodetection
serial_port = COM10
```

The default connection is set to automatic port search:

```
[bus2]
serial_port = auto
```

On Linux the latest connected USB device can also be seen with the command `dmesg`.

5.5.2. Generic settings

The `[generic]` section contains connection independent settings. These include debug logging settings and interrupt related settings.

The debug logging level is defined with `logging_level` command and the default level is `INFO`:

```
logging_level = INFO
```

Other alternatives are: `DEBUG`, `INFO`, `WARNING`, `ERROR` and `CRITICAL`.

The Data Ready (`drdy`) function mode for `*_data_logger.py` applications is defined with `drdy_function_mode` command and the default option is `ADAPTER_GPIO1_INT` to use `INT1` external interrupt pin.

```
drdy_function_mode = ADAPTER_GPIO1_INT
```

Other alternatives are: `ADAPTER_GPIO2_INT`, `REG_POLL`, and `TIMER_POLL`.

If Data Ready function mode is set to `TIMER_POLL`, the polling interval can be set using command `drdy_timer_interval`, and the default value is set to 0.04 seconds.

```
drdy_timer_interval = 0.04
```

When the `REG_POLL` option is used the data-ready bit is monitored from the status register to trigger device data reading. This will introduce more traffic on the bus and thus reduce ODR. This will also verify that all data is read from the device. Physical interrupt lines are not used in this option. Table 5 summarizes the available modes to acquire data with the *RoKiX Python CLI* with recommended setting highlighted in green.

Setting for <code>stream_mode</code>	Setting for <code>drdy_function_mode</code>	Physical interrupt lines	Measurement mode with data ready operation	Notes
TRUE	TIMER_POLL	Connections are not used.	Asynchronous reading	Firmware polls a data register with interval. Use <code>drdy_timer_interval</code> to configure the interval period.
TRUE	ADAPTER_GPIO1_INT ADAPTER_GPIO2_INT	Use device's interrupt lines.	Synchronous reading	Uses GPIO line 1 or 2 for the firmware interrupt. The fastest option and suitable for logging with higher ODR's. Default set to <code>ADAPTER_GPIO1_INT</code>
FALSE	REG_POLL	Connections are not used.	Synchronous reading	A client monitors the DRDY bit, increases traffic in the bus. The slowest mode.
FALSE	TIMER_POLL	Connections are not used.	Asynchronous reading	A client polls a data register with interval. Use <code>drdy_timer_interval</code> to configure the interval period.
FALSE	ADAPTER_GPIO1_INT ADAPTER_GPIO2_INT	Use device's interrupt lines.	Synchronous reading	Needs to configure GPIO numbers. A client polls either GPIO

Table 5. Data acquisition modes within the *RoKiX Python CLI*.

5.6. File structure of the evaluation kit

The overall structure of the *RoKiX Python CLI* is shown below.

```

RoKiX-Python-CLI/
__init__.py
get_configs.py
LICENSE.txt
plot.py
README.md
requirements_cloud.txt
requirements_linux.txt
requirements_linux_plot.txt
requirements_plot.txt
requirements_windows.txt
rokix_settings.cfg
stream_logger.py
├──cfg
├──examples
├──kx_lib
├──kx132
├──kx134
├──kxtj3
└──license

```

Each device has its own directory containing. For example, for KX132-1211 is in KX132¥ directory.

- Register definitions (for example `kx132_registers.py`)
- Reference driver implementation (for example `kx132_driver.py`)
- Data logger application for reading device data (for example `kx132_data_logger.py`)
- ASIC specific test applications (for example `kx132_data_wu_bts_logger.py`)

`kx_lib\` directory contains *RoKiX Python CLI* middleware.

5.7. Running test applications

The *RoKiX Python CLI* comes with number of test applications for different sensors that can help developers to quickly start testing many of the functions supported by a given sensor. Since test applications are device specific, they are located in their corresponding folders. For example, the test applications for KX132 sensors are located in `\Documents\RoKiX\RoKiX-Python-CLI\kx132\`. Additional test applications can be found in the examples folder in `\Documents\RoKiX\RoKiX-Python-CLI\examples\`. To access the list of the available argument that can be added on the command line for additional functionality, type `--help` at the end of the command:

```
cd kx132
python kx132_data_logger.py --help
```

```

usage: kx132_data_logger.py [-h] [-b BOARD] [--bus2 {USB,BLE_PYGATT,USB_SERIAL,USB_AARDVARK,BLE}]
                          [--serial_port SERIAL_PORT] [--ble_mac BLE_MAC]
                          [--drdy_function_mode {ADAPTER_GPIO1_INT,ADAPTER_GPIO2_INT,REG_POLL,TIMER_POLL}]
                          [--drdy_timer_interval DRDY_TIMER_INTERVAL]
                          [--other_function_mode {ADAPTER_GPIO1_INT,ADAPTER_GPIO2_INT,REG_POLL,TIMER_POLL}]
                          [--other_timer_interval OTHER_TIMER_INTERVAL] [-s STREAM_MODE]
                          [--max_timeout_count MAX_TIMEOUT_COUNT] [-o ODR] [-l LOOP] [--filename FILENAME]

Example: kx132_data_logger.py -s

optional arguments:
  -h, --help            show this help message and exit
  -b BOARD, --board BOARD
                        Board configuration file; Examples:rokix_board_rokix_sensor_node_i2c.json,
                        rokix_board_rokix_sensor_node_i2c_addon.json; Default:"rokix_board_cy8ckit059_spi_a3.json"
  --bus2 {USB,BLE_PYGATT,USB_SERIAL,USB_AARDVARK,BLE}
                        Bus2 connection; Default:"USB"
  --serial_port SERIAL_PORT
                        Serial COM port to use; Default:"auto"
  --ble_mac BLE_MAC
                        Mac address for bluetooth connection. Full address (or first bytes with windows);
                        Default:"None"
  --drdy_function_mode {ADAPTER_GPIO1_INT,ADAPTER_GPIO2_INT,REG_POLL,TIMER_POLL}
                        This setting defines how data ready function works in *_data_logger.py application;
                        Default:"ADAPTER_GPIO1_INT"
  --drdy_timer_interval DRDY_TIMER_INTERVAL
                        Dataready poll interval in seconds if drdy_function_mode is TIMER_POLL; Default:"0.04"
  --other_function_mode {ADAPTER_GPIO1_INT,ADAPTER_GPIO2_INT,REG_POLL,TIMER_POLL}
                        This setting defines how asic feature event works in applications where the features is
                        enabled; Default:"ADAPTER_GPIO2_INT"
  --other_timer_interval OTHER_TIMER_INTERVAL
                        Other function such as wakeup status poll interval in seconds if other_function_mode is
                        TIMER_POLL; Default:"0.04"
  -s STREAM_MODE, --stream_mode STREAM_MODE
                        Streaming mode on/off, 1/0, True/False; Default:"True"
  --max_timeout_count MAX_TIMEOUT_COUNT
                        ; Default:"1"
  -o ODR, --odr ODR
                        Output data rate; Default:"25.0"
  -l LOOP, --loop LOOP
                        Number of data samples; Default:"None"
  --filename FILENAME
                        File name with or without extension. If no extension is given .csv is used; Default:"None"

```

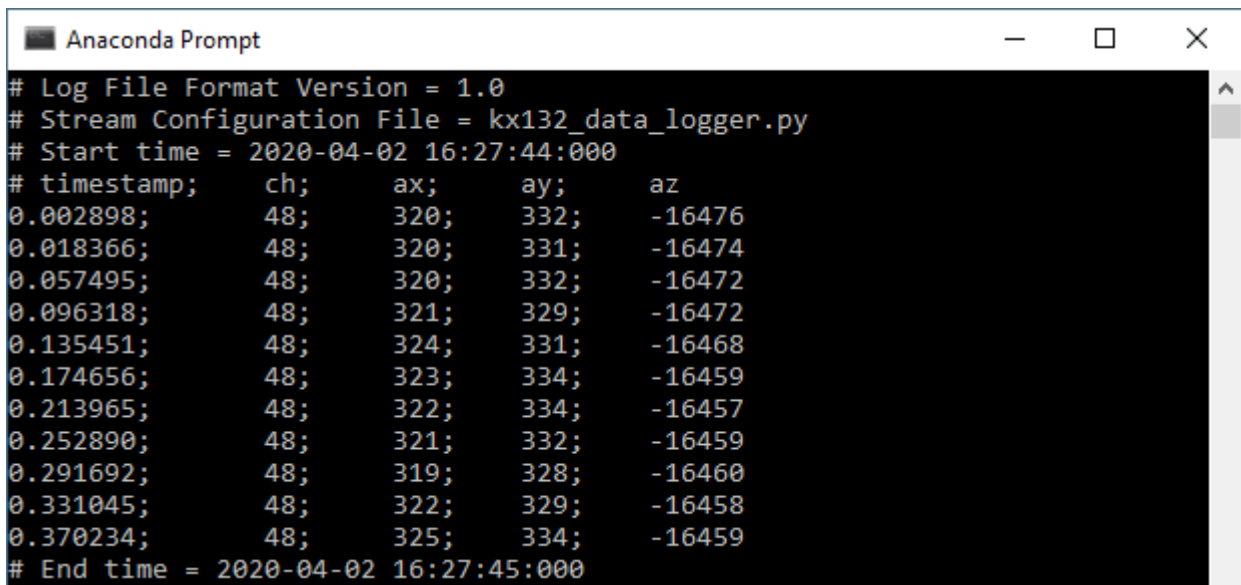
Figure 67. *RoKiX Python CLI* Help Menu

5.7.1. Data Logger

The data logger application prints the output data as it is being read from the output data registers. To run the data logger application, type:

```
cd kx132
python kx132_data_logger.py
```

To stop the streaming, press CTRL+C.



```
Anaconda Prompt
# Log File Format Version = 1.0
# Stream Configuration File = kx132_data_logger.py
# Start time = 2020-04-02 16:27:44:000
# timestamp;    ch;    ax;    ay;    az
0.002898;      48;    320;   332;   -16476
0.018366;      48;    320;   331;   -16474
0.057495;      48;    320;   332;   -16472
0.096318;      48;    321;   329;   -16472
0.135451;      48;    324;   331;   -16468
0.174656;      48;    323;   334;   -16459
0.213965;      48;    322;   334;   -16457
0.252890;      48;    321;   332;   -16459
0.291692;      48;    319;   328;   -16460
0.331045;      48;    322;   329;   -16458
0.370234;      48;    325;   334;   -16459
# End time = 2020-04-02 16:27:45:000
```

Figure 68. Data Logger Application

Device data can also be directed to a file with a specified name in the same directory using the following command:

```
python kx132_data_logger.py > kx132.csv
```

or it is possible to simultaneously print in the console and save into a file using `--filename` argument:

```
python kx132_data_logger.py --filename kx132.csv
```

Another way to provide the data logger file name without typing it in the console is to write in the `rokix_settings.cfg`:

```
...
# if not left empty then sensor data will be written to file
filename=kx132.csv
...
```

The data logged into a file can be viewed with a plotter (5.7.4).

5.7.2. Data Logger with Wake-up / Back-to-Sleep Engine Test

The functionality of the Wake-Up engine and Back-to-Sleep engine can be easily tested using the `kx132_data_wu_logger.py` test application :

```
python kx132_data_wu_logger.py
```

The Wake / Back-to-Sleep state can be monitored using the value printed in the `stat` column, which corresponds to the decimal output value read from the `STATUS_REG (0x19)` of the KX132 (Figure 69).

```

2020-04-03 20:23:44,541 INFO : kx132_test_wu_bts.py (96) : enable_wu_bts : enable_wu_bts start
100 0x64 KX132_1211_ADP_CNTL1 0x00 0b00000000 000
# Log File Format Version = 1.0
# Stream Configuration File = kx132_data_wu_bts_logger.py
# Start time = 2020-04-03 20:23:44:000
# timestamp; ch; ax; ay; az; stat
0.002330; 48; 517; 217; -16429; 0
0.073665; 48; 516; 211; -16426; 0
0.152183; 48; 513; 209; -16421; 0
0.230034; 48; 514; 211; -16421; 0
0.308288; 48; 298; 181; -16417; 0
0.386552; 48; 604; 97; -18112; 0
0.464563; 48; 1509; -1103; -21487; 0
0.542677; 48; -677; -137; -13832; 17
0.621150; 48; -1493; -588; -10837; 1
0.699524; 48; 113; -2060; -14777; 1
0.777229; 48; -945; -869; -13848; 1
0.855583; 48; -514; -1569; -13982; 1
0.933664; 48; -1428; -483; -13721; 1
1.012113; 48; -805; 475; -15517; 1
1.090079; 48; 423; 1252; -13835; 1
1.168456; 48; -192; 1539; -11322; 1
1.246677; 48; -716; -1282; -10643; 1
1.325184; 48; -1684; -2858; -7456; 1
1.403186; 48; -886; -1882; -4174; 1
1.481458; 48; -370; -500; -6954; 1
1.559041; 48; 93; -142; -12011; 1
1.637306; 48; 315; 37; -14371; 1
1.715462; 48; 418; 121; -15470; 1
1.793982; 48; 465; 159; -15978; 16
1.871773; 48; 488; 179; -16214; 0
1.950014; 48; 502; 190; -16326; 0
2.028168; 48; 508; 193; -16377; 0
2.106355; 48; 508; 195; -16401; 0
2.184366; 48; 507; 197; -16409; 0
2.262721; 48; 509; 197; -16415; 0
2.340821; 48; 513; 198; -16422; 0
2.418887; 48; 515; 198; -16423; 0
2.497284; 48; 512; 197; -16420; 0
2.575157; 48; 509; 198; -16417; 0
2.653300; 48; 508; 202; -16415; 0
2.731879; 48; 509; 201; -16413; 0
# End time = 2020-04-03 20:23:47:000

```

Figure 69. Data Logger with Wake-Up / Back-to-Sleep Engines Test

5.7.3. Additional Test Applications

Additional test applications for the KX132 and KX134 sensors can be found in the following installation folder:

```
\Documents\RoKiX\RoKiX-Python-CLI\kx132
```

NOTE: The following list of applications can be update over time:

- `python kx132_register_dump.py`

Description: Printout of the current values of all control registers of the KX132

- `python kx132_raw_adp_logger.py`

Description: Data logger for the data recorded in the standard output registers and in the Advance Data Path (ADP) output registers

- `python kx132_raw_adpwufbts_logger.py`

Description: This test application is similar to `kx132_data_wu_logger.py` described in section 0 but also adds printout of the ADP outputs.

- `python kx132_test_wu_bts.py`

Description: This test application helps to test Wake-Up and Back-to-Sleep engines, including direction of the motion.

5.7.4. Other provided tools

With all the Python-based scripts, a command line option `--help` shows the available arguments.

The following generic tools can be found in the [examples](#) installation folder:

```
\Documents\RoKiX\RoKiX-Python-CLI\examples
```

- `python rokix_board_version_info.py`

Description: This test application that reports the information about the host adapter platform including firmware information (RoKiX Protocol Version, RoKiX Firmware Version), as well as the hardware information such as Board ID and Board Unique ID (UID) when available (Figure 70).

```
Protocol version  2.0
Board ID         10
Board UID        44:00:02:04:18:10:1A:54
Firmware version 4c62e49f
```

Figure 70. Board Config Info

The following generic tools can be found in the following installation folder:

\Documents\RoKiX\RoKiX-Python-CLI

- `python get_configs.py`

Description: This is a Python script for downloading and extracting new board configurations. The board configurations will be placed to the following folder:

\Documents\RoKiX\RoKiX-Python-CLI\cfg.

- In addition to prebuilt data loggers, a general data logging tool using stream_configuration file is offered:

`python stream_logger.py <stream_configuration_file_name>`

This is a similar tool that is also available with RoKiX Windows GUI and RoKiX Android App.

- It is possible to view the data logged with Data Logger (5.7.1) using the `plot.py` application (Figure 71). Since the `kx132.csv` output file is located in \Documents\RoKiX\RoKiX-Python-CLI\kx132, it can be plotted using the following command when executed from the same folder:

`python ..\plot.py kx132.csv`

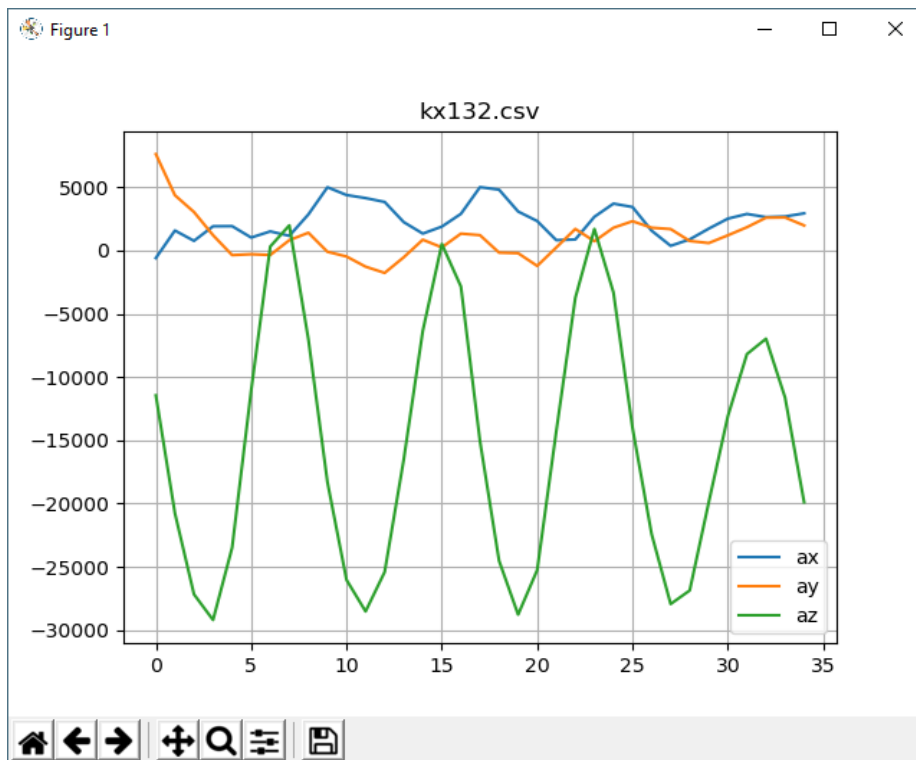


Figure 71. Recorded signals shown by the `plot.py` application

- The frequency content of the recorded data can be analyzed and shown using the `plot.py` application with a special option `-f` as follows:

`python ..\plot.py kx132.csv -f`

5.8. Changing test application configuration

Test applications allow the user to modify the most common device configuration alternatives; for example, g-range, filtering and ODR settings (Figure 72). It is recommended that the new configurations are given by the main program calling the test application and then overriding the default settings as required.

```

64
65 def enable_data_logging(sensor,
66                         odr=25,
67                         max_range='2G',
68                         lp_mode=False,
69                         low_pass_filter='ODR_9',
70                         power_off_on=True,           # set to False if this function is part of other configuration
71                         int_number=None,
72                         ch_out=CH_ACC):
73
74     LOGGER.info('enable_data_logging start')
75
76     #
77     # parameter validation
78     #
79
80     assert sensor.name in KX132Driver.supported_parts
81
82     assert convert_to_enumkey(odr) in e.KX132_1211_ODCNTL_OSA.keys(), 'Invalid for odr value {}'.format(
83         convert_to_enumkey(odr), e.KX132_1211_ODCNTL_OSA.keys())
84
85     assert max_range in e.KX132_1211_CNTL1_GSEL.keys(), 'Invalid for range value {}'.format(
86         max_range, e.KX132_1211_CNTL1_GSEL.keys())
87
88     assert (lp_mode in list(e.KX132_1211_LP_CNTL1_AVC.keys()) +
89             [False]), 'Invalid for lp_mode value {}'.format(
90                 lp_mode, e.KX132_1211_LP_CNTL1_AVC.keys())
91
92     assert low_pass_filter in list(e.KX132_1211_ODCNTL_LPRO.keys()) + \
93         ['BYPASS'], 'Invalid filter value {}'.format(
94             filter, e.KX132_1211_ODCNTL_LPRO.keys())
95
96     #assert ch_out in CH_ACC | CH_ADP | CH_TEMP, "not valid measurement channel"
97
98     # Set sensor to stand-by to enable setup change
99     if power_off_on:
100         sensor.set_power_off()
101
102     #
103

```

Figure 72. Python code of an example test application

5.9. Reference driver implementation

The RoKiX Python CLI offers platform independent reference device driver software implemented by Python. Since only the device related configurations are in the driver, and the platform specific dependencies are not visible, it gives a good starting point for porting the driver software to a desired target platform (Figure 73).

All drivers follow the same interface defined in `kx_lib%sensor_base.py`

```

kx132_driver.py x
22 import struct
23 from . import imports # pylint: disable=unused-import
24 from kx_lib.kx_configuration_enum import BUS1_I2C, BUS1_SPI
25 from kx_lib.kx_sensor_base import SensorDriver, AxisMapper
26 from kx_lib import kx_logger
27 from kx_lib.kx_util import delay_seconds, bin2uint16
28 from kx_lib.kx_configuration_enum import CH_ACC, CH_ADP, ACTIVE_LOW, ACTIVE_HIGH
29 from kx132 import kx132_1211_registers
30
31 LOGGER = kx_logger.get_logger(__name__)
32 # LOGGER.setLevel(kx_logger.DEBUG)
33
34 r = kx132_1211_registers.registers()
35 b = kx132_1211_registers.bits()
36 m = kx132_1211_registers.masks()
37 e = kx132_1211_registers.enums()
38
39 # activity modes
40 SLEEP, WAKE = range(2)
41
42 # fl_1a, fl_ba, fl_ca, fl_ish, fl_osh
43
44 filter1_values = {
45     'LP_ODR_4': (22, 0, 1439258, 1, 1),
46     'LP_ODR_8': (72, 3954428, 2796203, 2, 0),
47     'LP_ODR_16': (117, 6099540, 4815580, 4, 0),
48     'LP_ODR_32': (10, 7230041, 6354764, 5, 0),
49     'LP_ODR_64': (20, 7807115, 7301172, 7, 0),
50     'LP_ODR_128': (25, 8097550, 7826024, 9, 0),
51     'LP_ODR_256': (27, 8243038, 8102435, 11, 0),
52     'LP_ODR_512': (29, 8315818, 8244280, 13, 0),
53     'LP_ODR_1024': (29, 8352212, 8316131, 15, 0),
54     'LP_ODR_2048': (30, 8370410, 8352291, 17, 0),
55     'LP_ODR_4p400': (15, 2934914, 2176803, 1, 0),
56     'LP_ODR_4p266': (52, 4213708, 2986360, 2, 0),
57     'LP_ODR_4p830': (18, 4674381, 3358701, 2, 0),
58
59 }

```

Figure 73. Screen capture of a KX132 driver software

6. Troubleshooting and known issues

6.1. ODR accuracy and Timestamping

- Both with *RoKiX Windows GUI* and *RoKiX Python CLI*, timestamping is done on a PC and it is not accurate with high ODRs. This influences the delta time statistics.
- The real-time ODR shown in the *RoKiX Windows GUI* may show fluctuating and be off the nominal ODR value. If the value is within ~10% of the nominal value, the behavior is normal and can be due to combination of factors such as fluctuation of the actual sensor ODR due to internal oscillator jitter, as well as the timestamping error mentioned above. For cases where the ODR value is either significantly lower or higher than the nominal value, and “ODR has not reached the target value” pop-up window is shown, see section 0 for details.

6.2. USB Communication Troubleshooting

- USB communication may miss device data samples, or the USB connection is lost randomly: Use good quality USB cables which are USB certified.



- USB performance is not good on all Windows machines. The root cause is yet unknown.

6.3. RoKiX Windows GUI

- If an error message shown in Figure 74 appears, the Windows .NET installation is not up to date. Please run Windows update or install the required .Net version manually to resolve the issue.

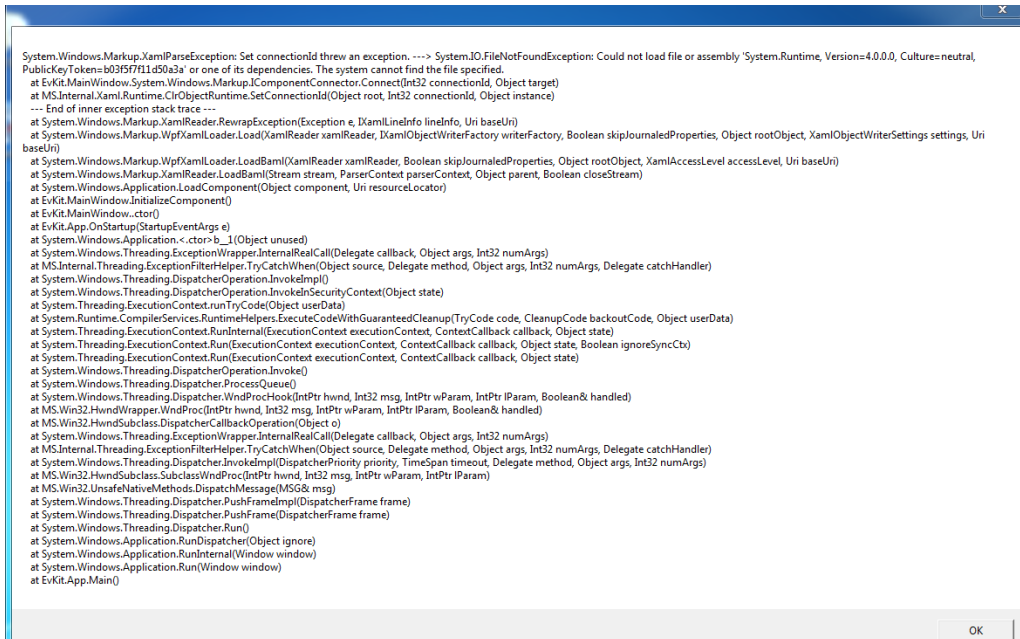


Figure 74. A Windows error message that indicates that Windows .NET installation is out-of-date.

- Sometimes after installation the desktop shortcuts will not work. To overcome this, please uninstall and reinstall the application again to a different destination directory.

- In case of connection problems or application crash, please check the error log file of the *RoKiX Windows GUI*. The default path of this file is: `\Documents\RoKiX\RoKiX-Windows-GUI\errorlog.txt`.

6.4. *RoKiX Python CLI Troubleshooting*

- In case there are issues with the *RoKiX Python CLI* operations, it is possible to change the logging level from default INFO to DEBUG for example.

```
logging_level = ERROR ; DEBUG / INFO / WARNING / ERROR / CRITICAL
```

- Additionally, logs can be saved to file by defining a file name for the key `log_file`, which is empty (e.g., no logging to file) by default.
- *The RoKiX Python CLI* verifies that it can capture all interrupts (before starting to monitor interrupt lines it is first verified that interrupt is not yet triggered). If the `logging_level` is set to WARNING or higher and interrupt speed (for example ODR) is too high a warning message will be displayed.
- If the driver or lib code has been modified and execution fails to import error, delete all `*.pyc` files.
- When having issues with accessing the serial ports on some Linux systems (e.g., Ubuntu 18.04) do the following: Create a file `/etc/udev/rules.d/50-rohm.rules` with the following content (as an administrator or use 'sudo'):

```
# CY8CKIT-059:
```

```
ACTION=="add", SUBSYSTEM=="usb", ENV{DEVTYPE}=="usb_device",
ATTRS{idVendor}=="04b5", ATTRS{idProduct}=="0601", ENV{ID_MM_DEVICE_IGNORE}="1"
ACTION=="add", KERNEL=="ttyACM[0-9]*", ATTRS{idVendor}=="04b5",
ATTRS{idProduct}=="0601", MODE="0666"
```

Reload newly created rules:

```
sudo udevadm control --reload-rules
```

- For additional common errors, see *RoKiX Python CLI relevant errors* in section 6.5 *RoKiX Development Kit Communication Issues Troubleshooting*.

6.5. RoKiX Development Kit Communication Issues Troubleshooting

The communication between the *RoKiX Windows GUI* or *RoKiX Python CLI* may not work for number of reasons. The issue can be related to hardware, software, or both. The following steps can be used as a guidance to troubleshoot such issues.

6.5.1. RoKiX Windows GUI Status Bar “Status: Disconnected”

Connection: USB (COM118) | Status: Disconnected | ODR: 0 | Stream: KX132-1211 / Accel data 50Hz ±2g high performance | Board: CY8CKIT-059 / RoKiX adapter A3 / I2C

If “Status: Disconnected” is shown in the Status bar, please check the following:

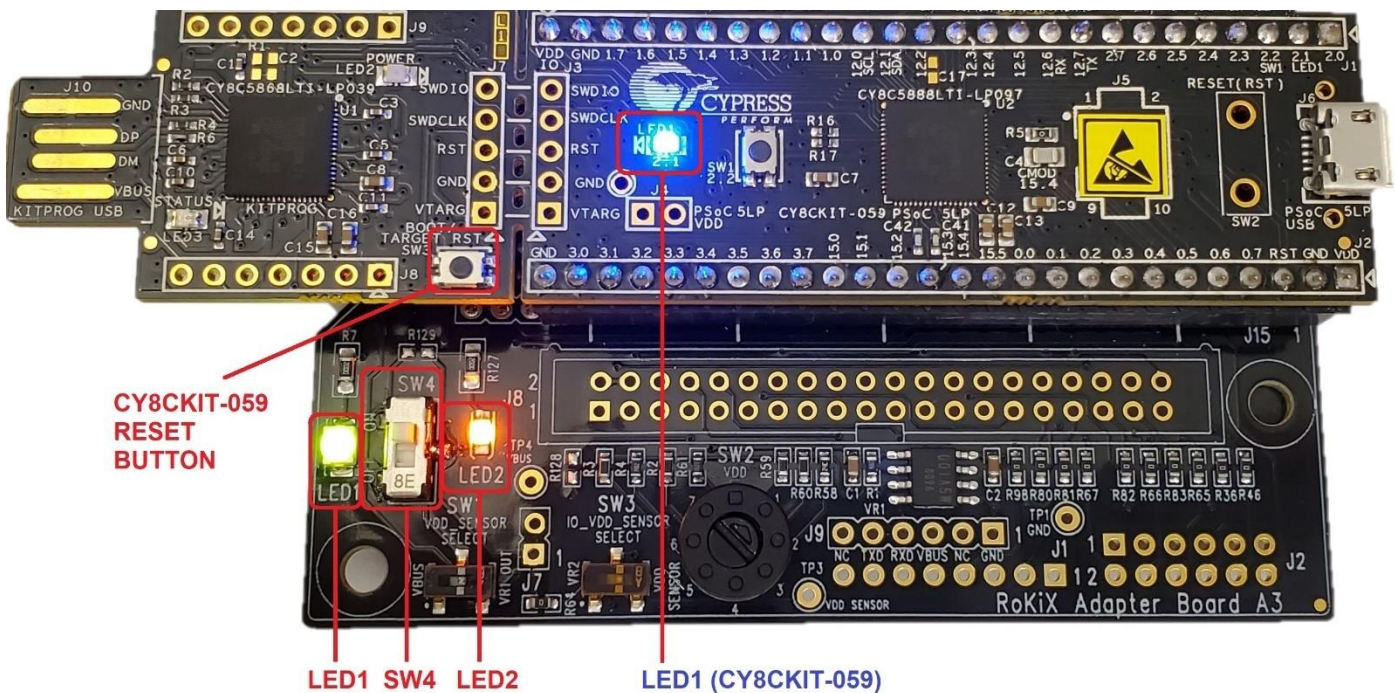


Figure 75. RoKiX Development Kit status: LED1, SW4, LED2, LED1 (CY8CKIT-059) must be ON

1. The **blue** LED1 (CY8CKIT-059) should be constantly ON and not blinking (Figure 75)
 - If the **blue** LED1 (CY8CKIT-059) is blinking, the CY8CKIT-059 is not programmed with the RoKiX firmware. Please program the latest RoKiX firmware (see 3.1.4.1 for details).
 - If the **blue** LED1 (CY8CKIT-059) is turned OFF, please try the following:
 - Check the microUSB cable is securely connected to *CY8CKIT-059 Prototyping Kit* and to the USB port on the PC (Figure 13)
 - Connect to a different USB port on the PC.
 - Replace microUSB cable with a new, high quality, USB certified cable.

2. The **green** LED1 (Figure 75).

- If the **green** LED1 is OFF but **blue** LED1 (CY8CKIT-059) is ON:
 - Ensure the CY8CKIT-059 Prototyping Kit is securely connected to the *RoKiX Adapter Board A3*.

6.5.2. *RoKiX Windows GUI* Status Bar “Status: No data”

Connection: USB (COM118) Status: No data | ODR: 0 Stream: KX132-1211 / Accel data 50Hz ±2g high performance Board: CY8CKIT-059 / RoKiX adapter A3 / I2C

If connection status as indicated in the status bar says, “No data”, the *RoKiX Windows GUI* does not receive the sensor data. To troubleshoot the issue, please check the following:

1. The **orange** LED2 should be ON (Figure 75). If **orange** LED2 is OFF, check the following:

- Check if the SW4 is in the ON position (i.e., moved up as seen from above) (Figure 75).
- If the SW4 is ON, check that the 7-position rotary switch SW2 (Figure 3) is not stuck in the intermediate position. The switch can be turned with a small flat screwdriver (see Table 2 for details).

NOTE: When using the *RoKiX Windows GUI* and the SW4 is not in the correct position, the Help menu item About Host Adapter Board would still work properly because the information about the host adapter firmware is read out from the *CY8CKIT-059 Prototyping Kit*, not the sensor.

2. The **orange** LED2 is ON:

- Check if the Stream selected corresponds to the sensor being tested (0).

NOTE: The *RoKiX Windows GUI* does not check the value of the WHO-AM-I register. Thus, while the stream for KX132-1211 would work for KX134-1211 and vice versa (although SI values in Raw Data setting will be incorrect), neither of those streams would work if KXTJ3 sensor is connected and vice-versa.

- Evaluation sensor board is securely connected to the *RoKiX Adapter Board* either directly or with the ribbon cable (Figure 10).
- Press CY8CKIT-059 Reset Button (Figure 75). Wait few seconds. If status changes to “Connected”, press “Streaming” button.
- Unplug the *CY8CKIT-059 Prototyping Kit* from the PC and plug back. Wait few seconds. If status changes to “Connected”, press “Streaming” button.
- Close the *RoKiX Windows GUI*. Unplug the *CY8CKIT-059 Prototyping Kit* from the PC. Plug the *CY8CKIT-059 Prototyping Kit* and restart the *RoKiX Windows GUI*.

6.5.3. RoKiX Python CLI “Automatic search found no devices”

```
Traceback (most recent call last):
  File "kx132_data_logger.py", line 191, in <module>
    main()
  File "kx132_data_logger.py", line 185, in main
    l = KX132DataLogger([KX132Driver])
  File "..\kx_lib\kx_data_logger.py", line 41, in __init__
    self.connection_manager = ConnectionManager(
  File "..\kx_lib\kx_board.py", line 104, in __init__
    found_ports = bus2connection.get_com_port()
  File "..\kx_lib\kx_bus2.py", line 294, in get_com_port
    raise EvaluationKitException('Automatic search found no devices')
kx_lib.kx_exception.EvaluationKitException: Automatic search found no devices
```

Figure 76. Error Message while running kx132_data_logger.py

If “Automatic search found no devices” error is shown when attempting to run any of the RoKiX python applications, e.g. Data Logger (5.7.1), please check that correct board configuration is selected in the rokix_settings.cfg file (5.5.1).

6.5.4. RoKiX Python CLI “No data received.”

```
Traceback (most recent call last):
  File "kx132_data_logger.py", line 191, in <module>
    main()
  File "kx132_data_logger.py", line 185, in main
    l = KX132DataLogger([KX132Driver])
  File "..\kx_lib\kx_data_logger.py", line 48, in __init__
    self.add_sensor(sensor())
  File "..\kx_lib\kx_data_logger.py", line 60, in add_sensor
    self.connection_manager.add_sensor(sensor)
  File "..\kx_lib\kx_board.py", line 483, in add_sensor
    _probe_status = sensor_driver.probe()
  File "..\kx132\kx132_driver.py", line 127, in probe
    resp = self.read_register(r.KX132_1211_WHO_AM_I)
  File "..\kx_lib\kx_sensor_base.py", line 196, in read_register
    return self.connection_manager.read_sensor_register(self, register, length)
  File "..\kx_lib\kx_board.py", line 217, in read_sensor_register
    return self.kx_adapter.adapter_read_sensor_register_i2c(target, sad, register, length)
  File "..\kx_lib\kx_adapter_evk.py", line 42, in adapter_read_sensor_register_i2c
    message_type_, message_data = self.receive_message(self.protocol.EVKIT_MSG_READ_RESP)
  File "..\kx_lib\kx_adapter_evk.py", line 285, in receive_message
    resp = self.engine.receive_single_message(wait_for_message, cache_messages)
  File "..\kx_lib\kx_protocol.py", line 273, in receive_single_message
    received_message = self._receive_single_message()
  File "..\kx_lib\kx_protocol.py", line 298, in _receive_single_message
    length_byte = array('B', [ord(self.connection.read(1))])
  File "..\kx_lib\kx_bus2.py", line 73, in read
    raise ProtocolTimeoutException('No data received.')
kx_lib.kx_exception.ProtocolTimeoutException: No data received.
```

Figure 77. RoKiX Python CLI “No data received” error

If “No data received” error is received when attempting to run any of the RoKiX python application, e.g. Data Logger (5.7.1), please follow the troubleshooting steps outlined in the section 6.5.2.

6.5.5. RoKiX Python CLI “Permission Error 'Access is denied.’”

```

Traceback (most recent call last):
  File "kx132_data_logger.py", line 191, in <module>
    main()
  File "kx132_data_logger.py", line 185, in main
    l = KX132DataLogger([KX132Driver])
  File "..\kx_lib\kx_data_logger.py", line 41, in __init__
    self.connection_manager = ConnectionManager(
  File "..\kx_lib\kx_board.py", line 111, in __init__
    bus2connection.initialize(com_port)
  File "..\kx_lib\kx_bus2.py", line 253, in initialize
    self._conn = serial.Serial(
  File "C:\Users\achernyakov\AppData\Local\Continuum\anaconda3\envs\rokix\lib\site-packages\serial\serialwin32.py", line 31, in __init__
    super(Serial, self).__init__(*args, **kwargs)
  File "C:\Users\achernyakov\AppData\Local\Continuum\anaconda3\envs\rokix\lib\site-packages\serial\serialutil.py", line 240, in __init__
    self.open()
  File "C:\Users\achernyakov\AppData\Local\Continuum\anaconda3\envs\rokix\lib\site-packages\serial\serialwin32.py", line 62, in open
    raise SerialException("could not open port {r}: {r}".format(self.portstr, ctypes.WinError()))
serial.serialutil.SerialException: could not open port 'COM118': PermissionError(13, 'Access is denied.', None, 5)

```

Figure 78. 'COMx' : PermissionError(13, 'Access is denied.', None, 5)

When 'COMx' : PermissionError(13, 'Access is denied.', None, 5) error is shown when attempting to run any of the RoKiX python applications, e.g. Data Logger (5.7.1), please check the access to the COM port is not used by another instance of the *RoKiX Python CLI* or *RoKiX Windows GUI*.

6.5.6. RoKiX Python CLI “EVKIT_ERR_BUS1; sensor-bus operation failed”

```

Traceback (most recent call last):
  File "kxtj3_data_logger.py", line 171, in <module>
    main()
  File "kxtj3_data_logger.py", line 165, in main
    l = KXTJ3DataLogger([KXTJ3Driver])
  File "..\kx_lib\kx_data_logger.py", line 48, in __init__
    self.add_sensor(sensor())
  File "..\kx_lib\kx_data_logger.py", line 60, in add_sensor
    self.connection_manager.add_sensor(sensor)
  File "..\kx_lib\kx_board.py", line 483, in add_sensor
    _probe_status = sensor_driver.probe()
  File "..\kxtj3\kxtj3_driver.py", line 62, in probe
    resp = self.read_register(r.KXTJ3_WHO_AM_I)
  File "..\kx_lib\kx_sensor_base.py", line 196, in read_register
    return self.connection_manager.read_sensor_register(self, register, length)
  File "..\kx_lib\kx_board.py", line 217, in read_sensor_register
    return self.kx_adapter.adapter_read_sensor_register_i2c(target, sad, register, length)
  File "..\kx_lib\kx_adapter_evk.py", line 42, in adapter_read_sensor_register_i2c
    message_type_, message_data = self.receive_message(self.protocol.EVKIT_MSG_READ_RESP)
  File "..\kx_lib\kx_adapter_evk.py", line 287, in receive_message
    return self.protocol.unpack_response_data(resp)
  File "..\kx_lib\kx_protocol_2_x.py", line 504, in unpack_response_data
    raise ProtocolException('Protocol error: ({}; {})'
kx_lib.kx_exception.ProtocolException: Protocol error: (EVKIT_ERR_BUS1; sensor-bus operation failed)

```

Figure 79. “EVKIT_ERR_BUS1; sensor-bus operation failed” error when running kxtj3_data_logger.py

When : “EVKIT_ERR_BUS1; sensor-bus operation failed” error is received, please make sure the correct sensor is connected to the *RoKiX Adapter Board A3*. An example of such error can be received when attempting to run the stream made for one sensor while having another sensor connected, or no sensor is connected at all (Figure 79).