XM123 Module Software

User Guide

XM123 Module Software

User Guide

Author: Acconeer AB

Version:a111-v2.15.1

Acconeer AB June 9, 2023

Contents

1 Introduction

The module software enable register-based access to radar functionality from external devices connected to a module. The module software is delivered as an image.

Typical usages of the module software are:

- Integration of radar functionality in your product to decrease development cost and time to market.
- Module evaluation and algorithm development in Python together with the "Acconeer Python Exploration Tool" that is available for download on GitHub https://github.com/acconeer/.

The module software provides a rich register-based API that can be accessed over UART, SPI and I^2C depending on module. The module software currently support the following services and detectors:

- Sparse Service
- Presence detector

Note that the performance and max range of the different detectors and services depends on the module that is being used as well as the configured settings like update rate and downsampling factor. Depending on use case the performance might not be good enough when using a low power module.

Support for more detectors is planned for future module software releases. A software image comprising the module software is available for download from Acconeer's website. See "Installing Software Image" at page 4 for instruction on how to install the module software. For an introduction to Acconeer's technology and product offer refer to "Introduction to Acconeer's sensor technology", available at the Acconeer website.

2 Installing Software Image

The XM123 uses the STM32L431 MCU which contains a ROM bootloader. The MCU is configured to enable the bootloader during manufacturing.

Another option is to use a SWD debugger, this requires additional hardware which is suitable when developing your own applications.

2.1 Flash Over UART Using STM32CubeProgrammer

Download and install STM32CubeProgrammer.

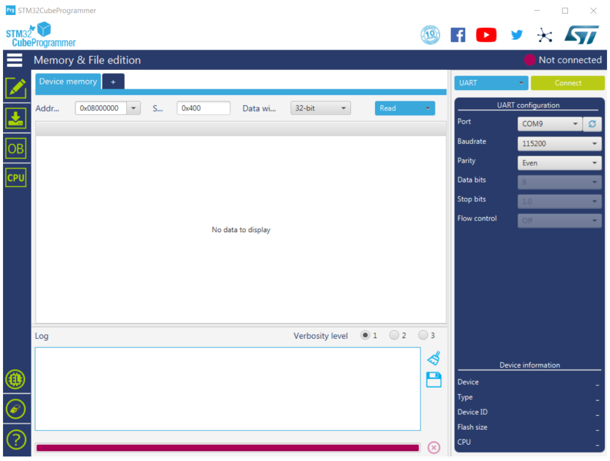
2.1.1 Boot the XM123 in bootloader mode

1. Connect the XE123 to your PC with an USB-C cable to the USB connector
2. Press and hold the “DFU” button on the board
3. Press the “RESET” button (still holding the “DFU” button)
4. Release the “RESET” button
5. Release the “DFU” button

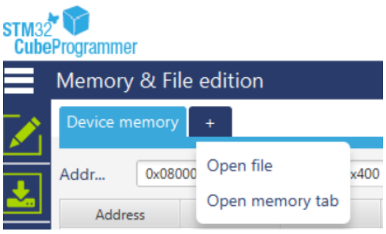
Your XM123 device is now in “DFU” mode waiting for a software upgrade procedure to be started.

2.1.2 Program the XM123

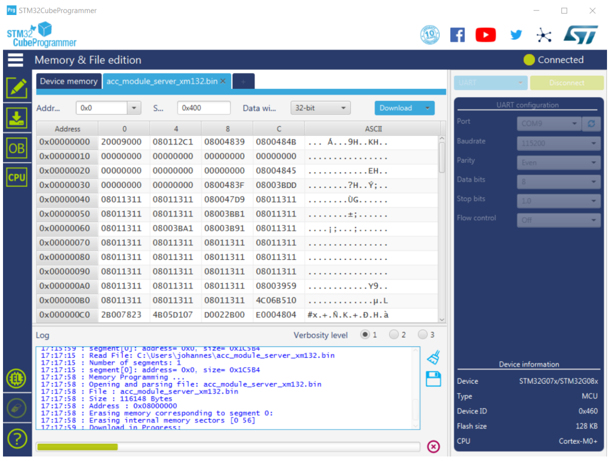
1. Start the STM32CubeProgrammer



2. Select correct port to the right. E.g. COM9.
3. Press “Connect” in the upper right corner
4. Press The “+” button and the “Open file”



5. Browse to and select the binary you like to program, e.g. “acc_module_server.bin”
6. Press the “Download” button. The green progress bar in the bottom indicates the progress



7. Once programming is complete press the "Disconnect" button
8. Press the "RESET" button or do a power cycle to start the embedded application

2.2 Flash Over UART Using stm32loader

The stm32loader is a python program. See pypi.org/project/stm32loader/ for more information.

Install it using "pip install stm32loader"

1. Set the XM123 into bootloader mode, see above for how to do this
2. Program the device with "stm32loader -p /dev/ttyUSB0 -e -w -v acc_module_server.bin". Make sure to specify correct port.
3. Press "RESET" or power cycle the device to start the embedded application

3 Power Save

Related Physical pins:

Pin Name	Functionality	Description
A11	WAKE_UP	This pin is active high and is used to wake up the module.
B9	MCU_INT	This pin is active high. Also see "INTERRUPT_MODE" and "INTERRUPT_MASK" registers.

The power consumption of the module is mainly affected by three registers: MODULE_POWER_MODE, SENSOR_POWER_MODE and UPDATE_RATE.

The registers for SENSOR_POWER_MODE, UPDATE_RATE and REPETITION_MODE mostly corresponds to the configuration for respective service and detector in the software API, see the documents at developer.acconeer.com.

3.1 MODULE_POWER_MODE

This controls the modules power mode. 0x00 (default) means highest performance with lowest latency. This is suitable to use when a high and accurate update frequency is needed.

0x01 Means that the module still is responsive, but there might be some delays and the update rate is not as accurate. Before communicating with the module the WAKE_UP pin must be set to high level. This mode is suitable when running lower frequency update rates where the REPETITION_MODE is set to 0x02 (on demand), SENSOR_POWER_MODE is set to HIBERNATE and UPDATE_RATE is set to 0. This enables the host controller to wakeup the module (e.g. once every minute) by raising the WAKE_UP pin and then clear the data and wait for the result.

3.2 SENSOR_POWER_MODE

The values corresponds towards the different ACC_POWER_SAVE_MODE_ modes in the RSS API: OFF(0), SLEEP(1), READY(2), ACTIVE(3), HIBERNATE(4). See the Service User Guide for respective service for more information.

Not all modes support this register, see the documentation for respective detector or service.

3.3 UPDATE_RATE

This controls the update rate. A value of 0 together with REPETITION_MODE set to 0x02 (on demand) means that the data is served as fast as possible once the data ready bit in the status register have been cleared by writing 0x04 to the MAIN_CONTROL register.

Not all modes support this register, see the documentation for respective detector or service.

3.4 REPETITION_MODE

This controls if the sensor or the module controls the update rate.

Not all modes support this register, see the documentation for respective detector or service.

4 Startup Timing

After providing power to the module or after a reset there is a 50 ms delay before the software is ready to be used. During this period no communication should be performed with the module.

5 Physical Interfaces

5.1 UART protocol

5.1.1 UART settings

The baud rate can be adjusted by writing to the UART_BAUDRATE register with the following sequence:

1. Write desired baudrate to the UART_BAUDRATE register
2. Wait for the "Register Write Response" packet
3. Change to the new baudrate

Default baud rate	115200
Byte size	8-bit
Parity	None
Flow control	RTS/CTS

The maximum supported baud rate is 1 Mbps. This can also be read from the PRODUCT_MAX_UART_BAUDRATE register.

The actual used baudrate is calculated as:

$$ActualBaudRate = \frac{80MHz}{USARTDIV}$$

For more detailed description see RM0394 Rev 4 chapter 38.5.4.

When using the XE123 the CP2105 (ECI block) is used between the host computer. CP2105 calculates its actual used baud rate as:

$$ActualBaudRate = \frac{48MHz}{2*ClockDivider}$$

For more detailed description see "6.1. ECI Baud Rate Generation" in CP2105 data sheet.

5.1.2 Byte Order

Multi byte integers are coded in little endian format.

5.1.3 Payload length

The payload length below is the length of the packet excluding start marker, the payload length itself, packet type and end marker. It can be used to read a packet without knowing anything about the different packet types. Also see 5.1.11 for a couple of example UART packages.

5.1.4 Register Read Request

Start marker	Payload length	Packet Type	Register Address	End Marker
0xCC	2 bytes	0xF8	1 byte	0xCD

5.1.5 Register Read Response

Start marker	Payload length	Packet Type	Register address	Register value	End Marker
0xCC	2 bytes	0xF6	1 byte	4 bytes	0xCD

5.1.6 Register Write Request

Start marker	Payload length	Packet Type	Register address	Register value	End Marker
0xCC	2 bytes	0xF9	1 byte	4 bytes	0xCD

5.1.7 Register Write Response

Start marker	Payload length	Packet Type	Register address	Register value	End Marker
0xCC	2 bytes	0xF5	1 byte	4 bytes	0xCD

5.1.8 Buffer Read Request

Start marker	Payload length	Packet Type	Buffer index	Buffer offset	End Marker
0xCC	2 bytes	0xFA	0xE8	2 bytes	0xCD

5.1.9 Buffer Read Response

Start marker	Payload length	Packet Type	Buffer index	Buffer data	End Marker
0xCC	2 bytes	0xF7	0xE8		0xCD

The format of the buffer data can be found under “Buffer Format” at page 13.

5.1.10 Buffer Streaming Payload

The streaming mode is primarily intended for communication with the Acconeer Python exploration package that is available on GitHub. The format of the steaming payload may be updated in a non-backward compatible way in future versions of the module software.

Start marker	Payload length	Packet Type	Streaming payload	End Marker
0xCC	2 bytes	0xFE		0xCD

The streaming payload consists of:

Result info marker	Result info length	Result info	Buffer marker	Buffer length	Buffer data
0xFD	2 bytes		0xFE	2 bytes	

The result info and the streaming buffer are the outputs from the Acconeer Service APIs encoded in little endian format.

The result info is a list of register (1 byte) and its value (4 bytes). The number of items in result info depends on the current mode. The list is terminated with 0xFE. More data may be added in future versions of the module software.

The format of the streaming buffer depends on the service.

Note that a streaming packet is sent asynchronous which means that the client must be able to handle that a streaming packet is received when e.g. a “Register Write Request” is sent but the “Register Write Response” has not yet been received.

The format of the buffer data can be found under “Buffer Format” at page 13.

5.1.11 Examples

5.1.12 Read Status Register

0xCC	0x01	0x00	0xF8	0x06	0xCD

5.1.13 Write Mode

0xCC	0x05	0x00	0xF9	0x02	0x02	0x00	0x00	0x00	0xCD

5.1.14 Buffer Streaming Payload

Index	Data	Description
0	0xCC	Start marker
1...2	0x3E 0x10	Payload length = 0x103E = 4158 bytes
3	0xFE	Packet type (Buffer streaming payload)
4	0xFD	Result info marker
5...6	0x14 0x00	Result info length = 0x0014 = 20 bytes
7	0xA1	Register 0xA1 (MISSED_DATA)
8...11	0x00 0x00 0x00 0x00	MISSED_DATA Value = 0x0000 0000 (No missed data)
12	0xA0	Register 0xA0 (DATA_SATURATED)
13...16	0x00 0x00 0x00 0x00	DATA_SATURATED Value = 0x0000 0000 (Data not saturated)
17	0xA3	Register 0xA3 (DATA_QUALITY_WARNING)
18...21	0x00 0x00 0x00 0x00	DATA_QUALITY_WARNING Value (No data quality warning)
22	0xA4	Register 0xA4 (SENSOR_COMM_ERROR)
23...26	0x00 0x00 0x00 0x00	SENSOR_COMM_ERROR Value (No comm error)
27	0xFE	Buffer marker
28...29	0x24 0x10	Buffer length = 0x1024 = 4132 Bytes
30...31	0xF4 0x00	Envelope data index 0 = 0x00F4
32...33	0xFA 0x00	Envelope data index 1 = 0x00FA
34...35	0x00 0x01	Envelope data index 2 = 0x0100
35...4124	...	Envelope data index 3...2065
4125	0xCD	End marker

5.2 I²C protocol

The module server supports communicating using I²C. Note that it is required that the host supports ”clock stretching”.

The device has a configurable address that is selected by the I2C_ADDRESS PIN according to the following table:

Connected to GND	0x51
Not Connected	0x52
Connected to VIN	0x53

The address is configured during start of the module software.

5.2.1 I²C Register Read Request

In order to read a register an I²C write transaction should first be performed:

Packet Type	Register Address
0xF8	1 byte

After this the register value can be read with an I2C read transaction:

Register Value
4 bytes

5.2.2 I²C Register Write Request

Register write can be performed in one transaction:

Packet Type	Register Address	Register Value
0xF9	1 byte	4 bytes

5.2.3 I²C Buffer Read Request

In order to read the buffer content an I²C write transaction should first be performed:

Packet Type	Buffer Index	Buffer Offset
0xFA	0xE8	2 bytes

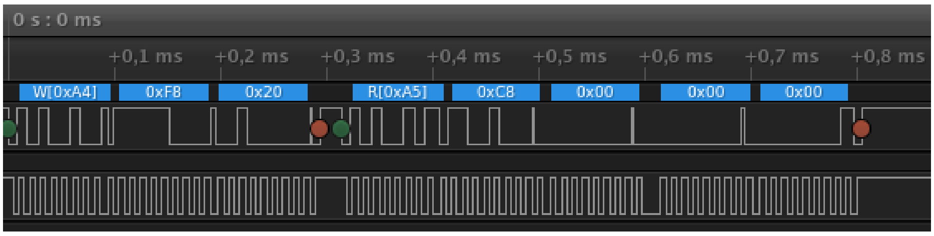
After this the buffer can be read with an I²C read transaction:

Buffer Data

The format of the buffer data can be found under “Buffer Format” at page 13.

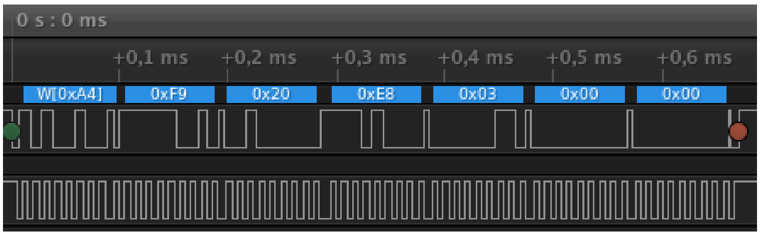
5.2.4 I²C Register Read Request Example

The following image shows an example when reading register 0x20 (RANGE_START). The returned register value in this example is 0xC8 (=200) mm.



5.2.5 I²C Register Write Request Example

The following image shows an example when writing 1000 (0x03E8) to register 0x20 (RANGE_START).



6 Buffer Format

The content of the buffer depends on which mode that is active:

<table>
<tr><th>Mode</th><th colspan="2">Streaming buffer format</th></tr>
<tr><td>Sparse</td><td colspan="2">Array of 16-bit unsigned integers</td></tr>
<tr><td rowspan="4">Presence</td><td>Offset</td><td>Description</td></tr>
<tr><td>0</td><td>0: No presence detected
1: Presence detected</td></tr>
<tr><td>1..4</td><td>Score (float)</td></tr>
<tr><td>5..8</td><td>Distance (float)</td></tr>
</table>

7 Register Map

7.1 General Registers

Addr	Read/ Write	Register Name	Function
0x02	R/W	MODE_SELECTION	Selects one of the supported sensor or service mode for the module.

0x04:	Sparse service mode.
0x400:	Presence detector mode.

Addr	Read/ Write	Register Name	Function
0x03	W	MAIN_CONTROL	Main Control Register. This register is used to control the operation of the module.

0x00:	Stop any started service or detector.
0x01:	Create the current service or detector. Sets the 'error_creation' status bit in case of error.
0x02:	Activate the current service or detector. Sets the 'error_activation' status bit in case of failure.
0x03:	Create and activate the current service or detector.
0x04:	Clears any status bits in the status register.

Addr	Read/ Write	Register Name	Function
0x05	R/W	STREAMING_CONTROL	Controls the streaming functionality.

0x00:	Disables UART data streaming.
0x01:	Enables UART data streaming.

Addr	Read/ Write	Register Name	Function
0x06	R	STATUS	Module Status Register. This register is a bit mask with current status of the module.

0x00000000:	No bits set.
0x000000FF:	Bits that can't be cleared with the clear status command.
0xFFFFFF00:	Mask with bits that can be cleared.
0xFFFF0000:	Mask with error bits.
0x00000001:	Service or detector is created.
0x00000002:	Service or detector is activated.
0x00000100:	Data is ready to be read from the buffer.
0x00010000:	An error occurred in the module.
0x00020000:	Invalid command or parameter received.
0x00040000:	Invalid mode
0x00080000:	Error creating the requested service or detector.
0x00100000:	Error activating the requested service or detector.
0x00200000:	An attempt to write a register or read the buffer when the module is in wrong state.

continued ...

...continued

Addr	Read/ Write	Register Name	Function
0x07	R/W	UART_BAUDRATE	Controls the baudrate for the UART interface. Read the product_max_uart_baudrate register to get the maximum supported baudrate. 0x1C200: Default baudrate for the module.
0x08	R/W	INTERRUPT_MASK	Mask for interrupts. Interrupt is active when corresponding bit in the status register is set. The interrupt is inactive when the bit is cleared. Also see interrupt_mode register. 0x00000000: No interrupts. 0x00000001: Interrupt when service or detector is created. 0x00000002: Interrupt when service or detector is activated. 0x00000100: Interrupt on data ready. 0x00010000: Interrupt on error. 0x00020000: Interrupt on invalid command. 0x00040000: Interrupt on invalid mode. 0x00080000: Interrupt on error creating service or detector. 0x00100000: Error activating the requested service or detector. 0x00200000: An attempt to write a register or read the buffer when the module is in wrong state.
0x09	R/W	INTERRUPT_MODE	Set mode for interrupt 0x00: Interrupt disabled, MCU_INT pin is always inactive. 0x01: MCU_INT is active when interrupt is active.
0x0A	R/W	MODULE_POWER_MODE	Module power configuration. This register is hardware specific and described in the ”Power Save” chapter.
0x10	R	PRODUCT_IDENTIFICATION	Module Identification register. 0xACC0: The module is a XM112. 0xACC1: The module is a XM122. 0xACC2: The module is a XM132. 0xACC3: The module is a XM131. 0xACC5: The module is a XM124. 0xACC6: The module is a XM123.
0x11	R	PRODUCT_VERSION	Software product version register as 0xMMIIPP where MM is major, II is minor and PP is patch version.
0x12	R	PRODUCT_MAX_UART_BAUDRATE	The maximum UART baudrate supported by the module.
0xE9	R	OUTPUT_BUFFER_LENGTH	Length of data in output buffer.

7.2 Sparse Registers

Registers which are writable can be used to set a configuration. Registers which are read only contain metadata which is updated either after create or when data is produced. It is recommended to read the the service and detector user guides for more information on configuration and metadata.

Addr	Read/ Write	Register Name	Function
0x20	R/W	RANGE_START	Start range in mm of the measurement.
0x21	R/W	RANGE_LENGTH	Length of the range in mm.
0x22	R/W	REPETITION_MODE	Repetition mode for the measurement.

0x01:	The sensor controls the update rate with high precision according to the value in the update_rate register.
0x02:	The update rate is software limited according to the value in the update_rate register. A value of 0 means no limit of the update rate.

0x23	R/W	UPDATE_RATE	The measurement update rate in mHz (i.e. step in 1/1000Hz). See the repetition_mode register for more information.
0x24	R/W	GAIN	Receiver gain, 0-1000 where 0 is the lowest gain and 1000 the highest.
0x25	R/W	SENSOR_POWER_MODE	Radar sensor power mode. See the Service User Guide for respective service for more information.

0x00:	Sensor off power mode. Whole sensor is shutdown between sweeps, consumes least power, supports lower frequencies.
0x01:	Sensor sleep power mode.
0x02:	Sensor ready power mode.
0x03:	Sensor active power mode. Whole sensor is active. Consumes most power, supports higher frequencies.
0x04:	Sensor hibernate power mode. Sensor is still powered but the internal oscillator is turned off and the application needs to clock the sensor by toggling a GPIO a pre-defined number of times to enter and exit this mode. Only supported for the sparse service on XM122, XM123, XM124, XM131 and XM132 currently.

0x26	R/W	TX_DISABLE	Used to measure RX noise floor and to support TX off spectrum regulation measurements.
0x28	R/W	PROFILE_SELECTION	Each profile consists of a number of settings for the sensor that configures the RX and TX paths.

0x01:	Profile 1 maximizes on the depth resolution
0x02:	Sliding scale between profile 1 and 5.
0x03:	Sliding scale between profile 1 and 5.
0x04:	Sliding scale between profile 1 and 5.
0x05:	Profile 5 maximizes on radar loop gain with a sliding scale in between.

continued ...

. . . continued

Addr	Read/ Write	Register Name	Function
0x29	R/W	DOWNSAMPLING_FACTOR	Downsampling factor to be used in sensor.
0x30	R/W	HW_ACC_AVERAGE_SAMPLES	The number of hardware accelerated averaged samples for each data point.
0x32	R/W	MAXIMIZE_SIGNAL_ATTENUATION	Maximize signal attenuation to avoid saturation in direct leakage.
0x33	R/W	ASYNCHRONOUS_MEASUREMENT	Used to enable/disable asynchronous mode.
0x34	R/W	MUR	The maximum unambiguous range.

0x06:	Maximum unambiguous range 11.5 m, maximum measurable distance 7.0 m
0x09:	Maximum unambiguous range 17.3 m, maximum measurable distance 12.7 m

0x40	R/W	SPARSE_SWEEPS_PER_FRAME	The number of sweeps per frame.
0x41	R/W	SPARSE_REQ_SWEEP_RATE	The sweep rate in mHz. Set to 0 for maximum possible.
0x42	R/W	SPARSE_SAMPLING_MODE	Sampling mode

0x00:	A
0x01:	B

0x81	R	START	Start of the sweep in mm.
0x82	R	LENGTH	Length of the sweep in mm.
0x83	R	DATA_LENGTH	Length of the sparse data.
0x84	R	SWEEP_RATE	Sweep rate in mHz.
0x85	R	STEP_LENGTH	Distance in um between adjacent data points.
0xA0	R	DATA_SATURATED	Indication of sensor data being saturated, can cause result instability.
0xA1	R	MISSED_DATA	True if data was lost. Try lowering the update_rate or read the data more often.
0xA4	R	SENSOR_COMM_ERROR	True is an indication of a sensor communication error, service or detector probably needs to be restarted.

7.3 Presence Registers

Registers which are writable can be used to set a configuration. Registers which are read only contain metadata which is updated either after create or when data is produced. It is recommended to read the the service and detector user guides for more information on configuration and metadata.

Addr	Read/ Write	Register Name	Function
0x20	R/W	RANGE_START	Start range in mm of the measurement.
0x21	R/W	RANGE_LENGTH	Length of the range in mm.
0x23	R/W	UPDATE_RATE	The measurement update rate in mHz (i.e. step in 1/1000Hz). See the repetition_mode register for more information.
0x24	R/W	GAIN	Receiver gain, 0-1000 where 0 is the lowest gain and 1000 the highest.
0x25	R/W	SENSOR_POWER_MODE	Radar sensor power mode. See the Service User Guide for respective service for more information.

0x00:	Sensor off power mode. Whole sensor is shutdown between sweeps, consumes least power, supports lower frequencies.
0x01:	Sensor sleep power mode.
0x02:	Sensor ready power mode.
0x03:	Sensor active power mode. Whole sensor is active. Consumes most power, supports higher frequencies.
0x04:	Sensor hibernate power mode. Sensor is still powered but the internal oscillator is turned off and the application needs to clock the sensor by toggling a GPIO a pre-defined number of times to enter and exit this mode. Only supported for the sparse service on XM122, XM123, XM124, XM131 and XM132 currently.

0x28	R/W	PROFILE_SELECTION	Each profile consists of a number of settings for the sensor that configures the RX and TX paths.

0x01:	Profile 1 maximizes on the depth resolution
0x02:	Sliding scale between profile 1 and 5.
0x03:	Sliding scale between profile 1 and 5.
0x04:	Sliding scale between profile 1 and 5.
0x05:	Profile 5 maximizes on radar loop gain with a sliding scale in between.

0x29	R/W	DOWNSAMPLING_FACTOR	Downsampling factor to be used in sensor.
0x30	R/W	HW_ACC_AVERAGE_SAMPLES	The number of hardware accelerated averaged samples for each data point.
0x33	R/W	ASYNCHRONOUS_MEASUREMENT	Used to enable/disable asynchronous mode.
0x40	R/W	THRESHOLD	Detection threshold in 1/1000 for presence.
0x41	R/W	SWEEPS_PER_FRAME	Sweeps per frame for the data from the underlying (sparse) service.
0x42	R/W	INTER_FRAME_DEV_TIME_CONST	Time constant in 1/1000 s of the low pass filter for the (inter-frame) deviation between fast and slow.
0x43	R/W	INTER_FRAME_FAST_CUTOFF	Cutoff frequency in mHz of the low pass filter for the fast filtered subsweep mean.

continued . . .

...continued

Addr	Read/ Write	Register Name	Function
0x44	R/W	INTER_FRAME_SLOW_CUTOFF	Cutoff frequency in mHz of the low pass filter for the slow filtered subsweep mean.
0x45	R/W	INTRA_FRAME_TIME_CONST	Time constant in 1/1000 s for the intra frame part.
0x46	R/W	INTRA_FRAME_WEIGHT	The weight, 0-1000, of the intra-frame part in the final output. A value of 1000 corresponds to only using the intra-frame part and a value of 0 corresponds to only using the inter-frame part.
0x47	R/W	OUTPUT_TIME_CONST	Time constant in 1/1000 s of the low pass filter for the detector output.
0x48	R/W	NBR_REMOVED_PC	The number of principal components removed in the PCA based noise reduction. Value between 0 and 2 where 0 disables the PCA based noise reduction completely.
0x49	R/W	REQ_SWEEP_RATE	The sweep rate in mHz. Set to 0 for maximum possible.
0xA0	R	DATA_SATURATED	Indication of sensor data being saturated, can cause result instability.
0xA4	R	SENSOR_COMM_ERROR	True is an indication of a sensor communication error, service or detector probably needs to be restarted.
0xB0	R	DETECTED	Presence detected or not
0xB1	R	SCORE	Score of the detected movement
0xB2	R	DISTANCE	Distance in mm to the detected movement

8 Examples

8.1 Python Example

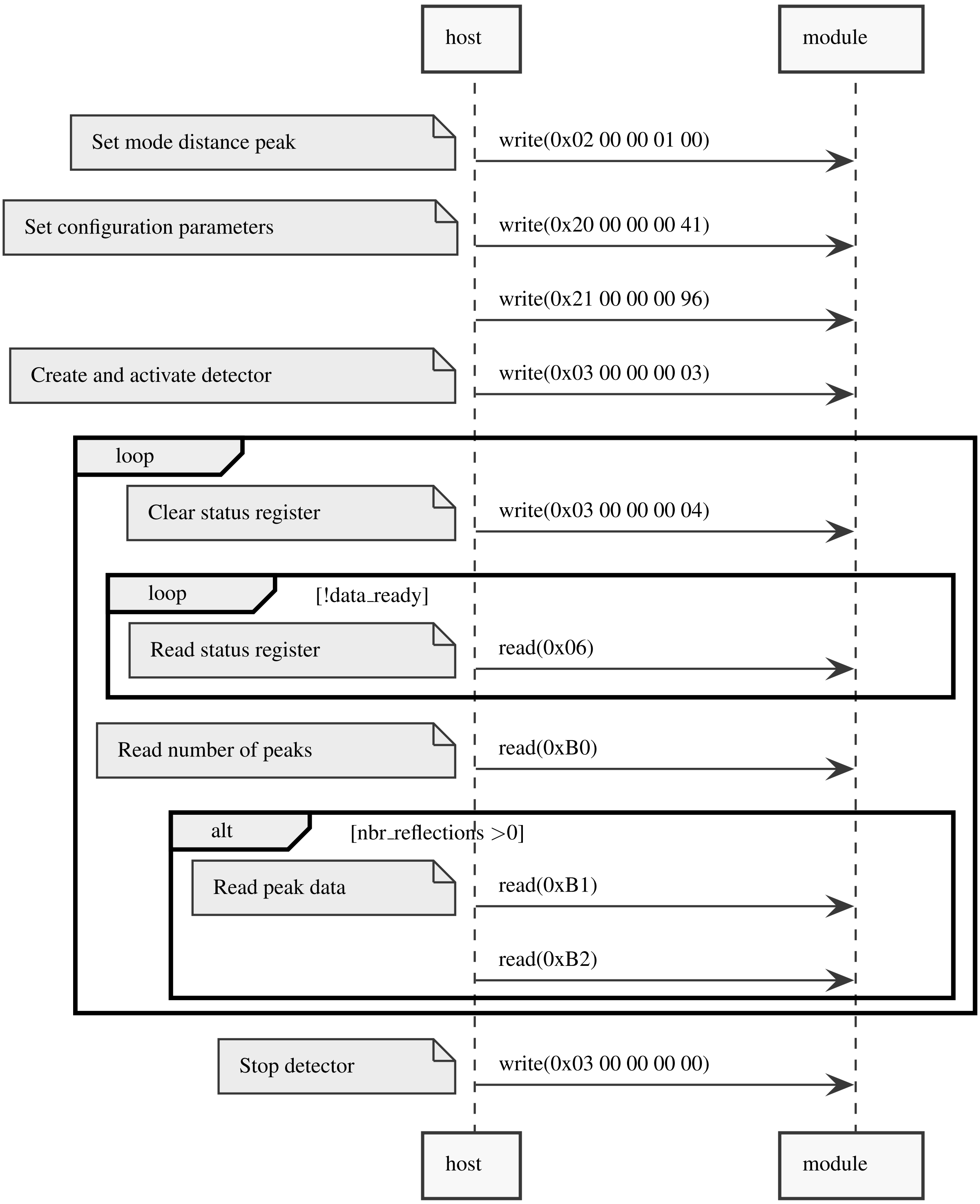
There is a simple python example delivered together with the module software binary. This shows how to communicate with the module software over the UART interface.

The following examples only works for XM124 but the principle is the same for the presence detector.

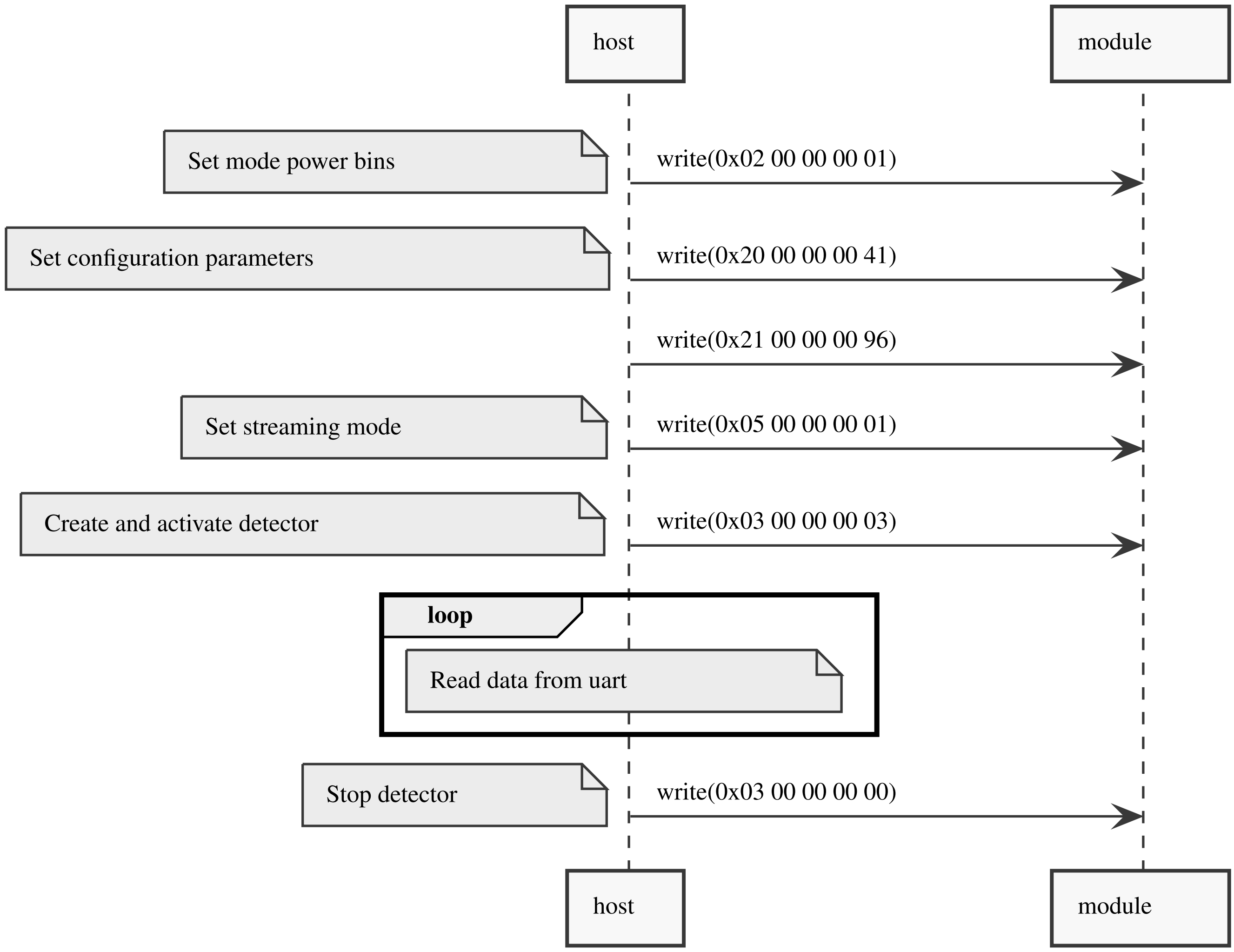
Example:

python3 module_software_example.py --port /dev/ttyUSB0

8.2 Reading Distances



8.3 Reading Power Bin Data (UART Streaming)



9 Debug Logging Output

RSS and module server logs can be retrieved from the second UART that is exposed through the USB. If the USB is not used, please use the DBUG-RX pin on XE123 which in turn is connected to UART_DEBUG_TX. This means that you need to connect your UART_RX pin on your host to DBUG-RX on your XE123 board.

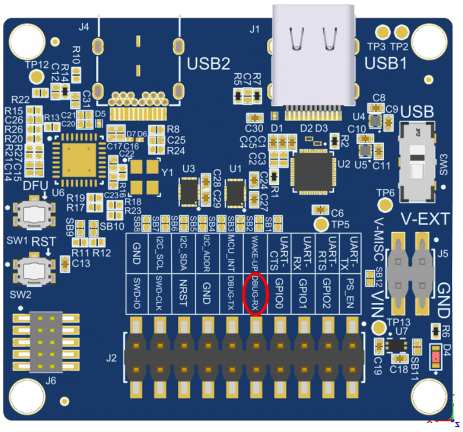


Figure 1: DBUG-RX on XE123

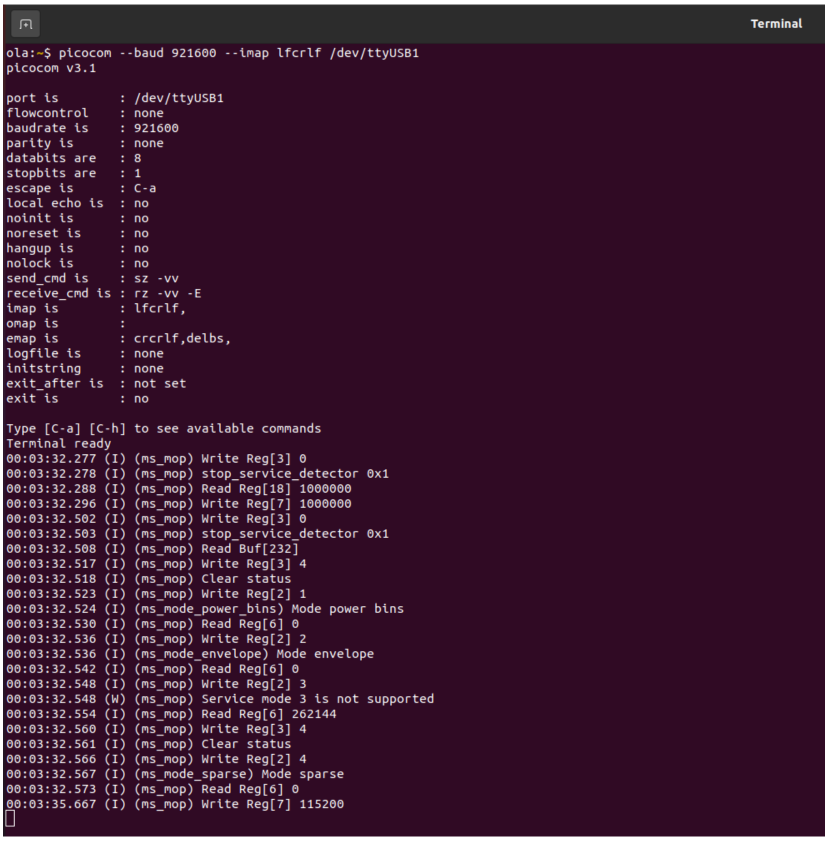


Figure 2: Example log output

Baudrate	921600
Byte size	8
Parity	None
Stop bits	1

Table 4: Debug UART Settings

10 Disclaimer

The information herein is believed to be correct as of the date issued. Acconeer AB ("Acconeer") will not be responsible for damages of any nature resulting from the use or reliance upon the information contained herein. Acconeer makes no warranties, expressed or implied, of merchantability or fitness for a particular purpose or course of performance or usage of trade. Therefore, it is the user's responsibility to thoroughly test the product in their particular application to determine its performance, efficacy and safety. Users should obtain the latest relevant information before placing orders.

Unless Acconeer has explicitly designated an individual Acconeer product as meeting the requirement of a particular industry standard, Acconeer is not responsible for any failure to meet such industry standard requirements.

Unless explicitly stated herein this document Acconeer has not performed any regulatory conformity test. It is the user's responsibility to assure that necessary regulatory conditions are met and approvals have been obtained when using the product. Regardless of whether the product has passed any conformity test, this document does not constitute any regulatory approval of the user's product or application using Acconeer's product.

Nothing contained herein is to be considered as permission or a recommendation to infringe any patent or any other intellectual property right. No license, express or implied, to any intellectual property right is granted by Acconeer herein.

Acconeer reserves the right to at any time correct, change, amend, enhance, modify, and improve this document and/or Acconeer products without notice.

This document supersedes and replaces all information supplied prior to the publication hereof.