

OxTech MCM User Guide

OxTech MCM

10/16/2023 | Document Version: 1.0

Table of Contents

1. Introduction	3
2. OxTech MCM Module	4
2.1. OxTech MCM (Multi Connectivity Module) PCB Appearance	4
2.2. MCM Pins	5
3. OxTech MCM EVK	6
3.1. EVK's J-Link Connection	7
3.2. Module Power up via EVK	8
3.3. EVK Pins for the MCM Command Interface	8
3.4. Debug Log Interface	8
4. MCM Command Manual	9
4.1. Serial Port	9
4.2. Optional GPIO Synchronization Lines	9
4.2.1. COMMAND Line	9
4.2.2. BUSY Line	9
4.2.3. EVENT Line	10
4.3. Message Flow	10
4.3.1. Message Flow for Command and Response using GPIO Lines	10
4.3.2. Message Flow for Modem Events using GPIO Lines	11
4.3.3. Message Flow for Command and Response without GPIO Lines	11
4.3.4. Message Flow for Modem Events without GPIO Lines	11
4.4. Message Format	11
4.4.1. Input Command Packets	11
4.4.2. Response Packets/Modem Event Command Packets	12
4.5. Input Commands and Responses	13
4.5.1. Overview of Commands and Responses	13
4.5.2. Commands Details	13
4.5.3. Input commands and their respective response	14
4.5.3.1. GetEvent	14
4.5.3.2. GetVersion	15
4.5.3.3. Reset	16
4.5.3.4. Factory Reset	17
4.5.3.5. RequestTx	18
4.5.3.6. FSK Link Request	18
4.5.3.7. CSS Link Request	19
4.5.3.8. BLE Link Request	20
4.5.3.9. BLE Connection Request	20
4.5.4. LED Pattern	21
5. Configure AWS CLI	22
6. Create and Flash Manufacturing File	22

6.1. Create Manufacturing File	22
6.2. Flash Manufacturing File	23
7. Create Custom Application (Standalone mode)	23
7.1. Create an Amazon Sidewalk Project for Oxit Sidewalk Module	24
7.2. Modify Amazon Sidewalk Sample Application	25
7.3. Build your Amazon Sidewalk Application	27
7.4. Flash your Amazon Sidewalk Application	27
7.5. Check Amazon Sidewalk Application logs	28
8. Monitor Uplink Data	28
9. Send A Downlink	28
10. Generale Notes	29
11. Known Issues	29
12. CONFIDENTIAL & PROPRIETARY	30

Revision History

Date	Version	Description
10/16/2023	1.0	First draft

1. Introduction

The OxTech MCM (Multi Connectivity Module) is a combination of hardware platform and modem firmware running on it. It delivers a self-contained HW and FW Amazon Sidewalk solution.

The module embeds an EFR32MG24 microcontroller, with advanced security characteristics and low power consumption. It also embeds Semtech's SX126X LoRa Transceiver.

The Oxtech MCM solution provides Amazon Sidewalk connectivity with all three different link types:

- BLE (Low Energy Bluetooth)
- FSK (Frequency Shift Keying)
- CSS (Chirp Spread Spectrum)

The MCM modem is designed to function in two different modes. These modes are achieved by flashing different firmware binaries into the MCM's EFR32MG24 microcontroller.

1. **Network co-processor mode** - The MCM acts as a sidewalk modem which accepts serial commands from another main controller/processor. The main controller is also referred to as "application MCU" or "Host".
2. **Standalone mode** - Users can flash their own application firmware into the modem's EFR32MG24 microcontroller.

Note: The main focus of this document is to describe the "**Network co-processor mode**" and the supported commands. Additionally, this document also describes the "[Standalone mode](#)" towards the latter part.

2. OxTech MCM Module

The OxTech MCM is a 29-pin SMD module with a formfactor of 25.4 mm x 22.8 mm x 4.0 mm.

2.1. OxTech MCM (Multi Connectivity Module) PCB Appearance

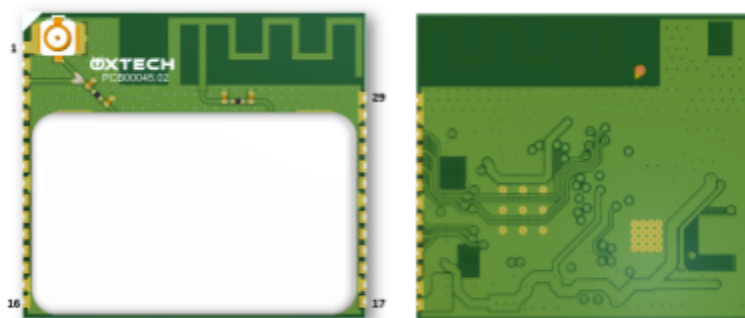


Figure 1: MCM module top and bottom views

2.2. MCM Pins

Pin No	Terminal Name	Type	Description
1	GND	Power	Ground
2	GND	Power	Ground
3	ANT	RF	LoRa RF path for application board antenna
4	GND	Power	Ground
5	JTAG-SWO	O	Debug Serial Wire Output
6	JTAG-SWIO	I/O	Debug Serial Wire I/O
7	JTAG-SWCLK	CLK	Debug Serial Wire Clock
8	RESET	I	Sidewalk Application MCU Reset
10	UART TX	O	UART TX, Module to host
11	UART RX	I	UART RX, Host to Module
12	COMMAND	I	Module Command input
13	EVENT	O	Module Event Out
14	BUSY	O	Module Busy Out
15	GND	Power	Ground
16	GND	Power	Ground
17	VDD	POWER	Module VDD Supply
18	GND	Power	Ground
19	EXT_GPIO4	I/O	General Purpose Input/Output
20	EXT_GPIO1	I/O	General Purpose Input/Output
21	EXT_GPIO2	I/O	General Purpose Input/Output
22	EXT_GPIO3	I/O	General Purpose Input/Output

23	GND	Power	Ground
24	FLASH_SCLK	CLK	Module/eFlash SPI Clock
25	FLASH_MOSI	I	Module/eFlash SPI Master Out Slave In
26	FLASH_MISO	O	Module/eFlash SPI Master In Slave Out
27	FLASH_CS	I	Module/eFlash SPI Chip Select
28	GND	Power	Ground
29	GND	Power	Ground

3. OxTech MCM EVK

The OxTech MCM EVK is an evaluation kit designed for easily interfacing with the MCM module and developing Sidewalk applications with an external application MCU (Host).

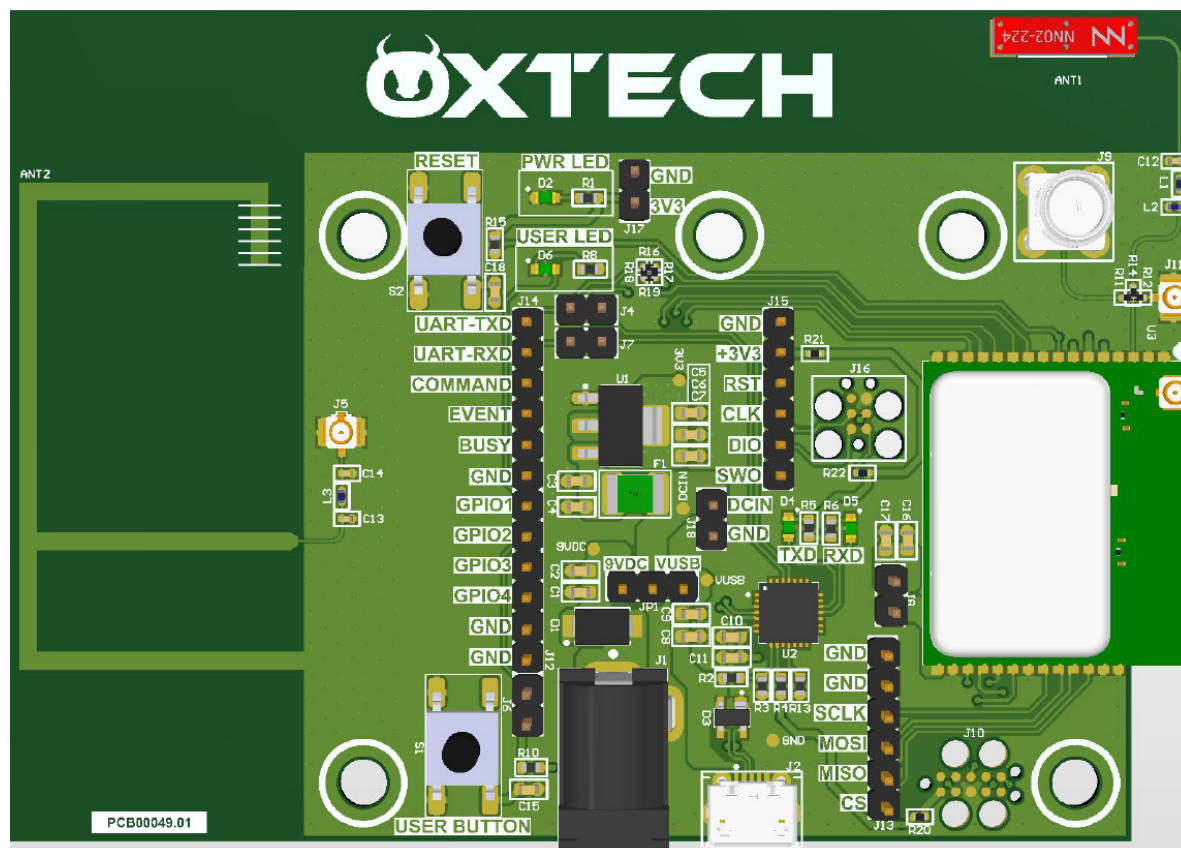


Figure 2: PCB Top View

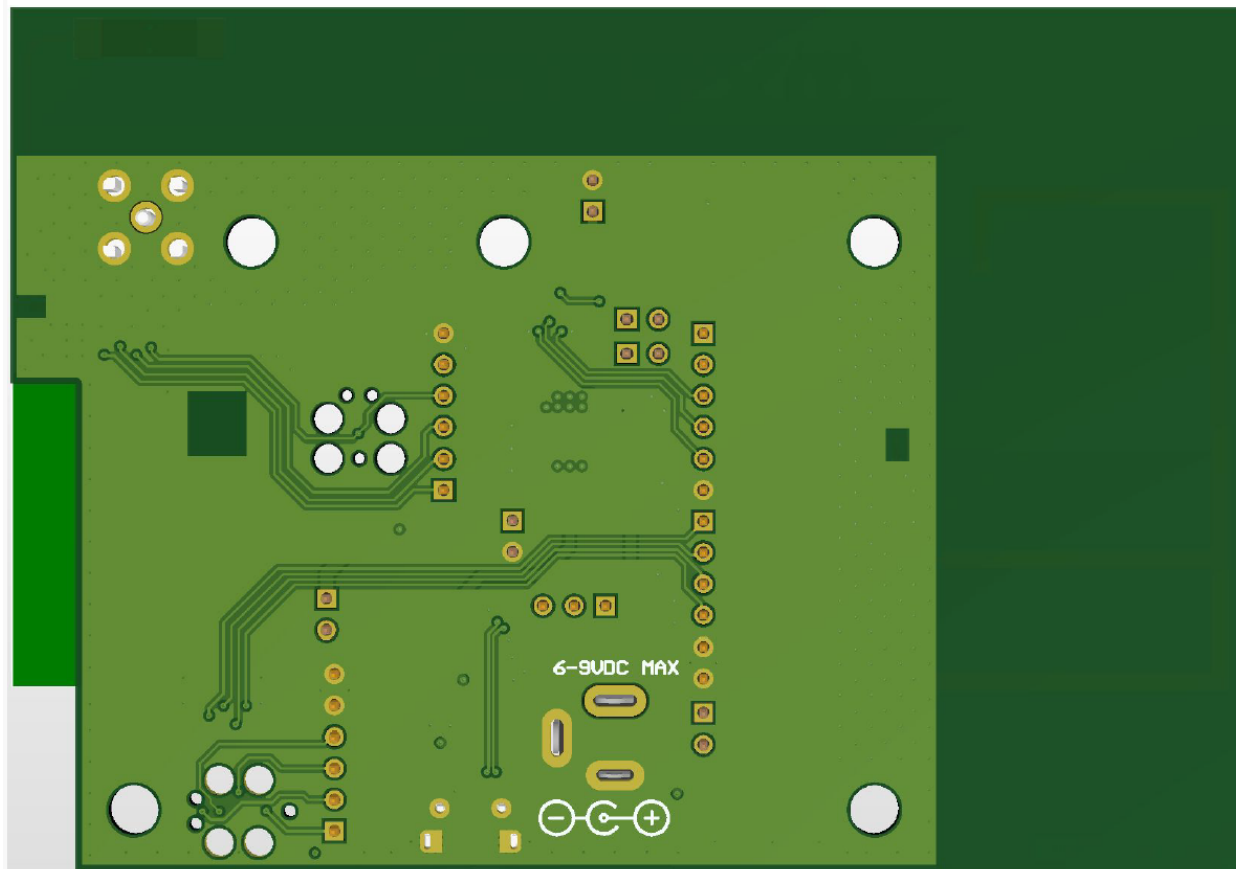
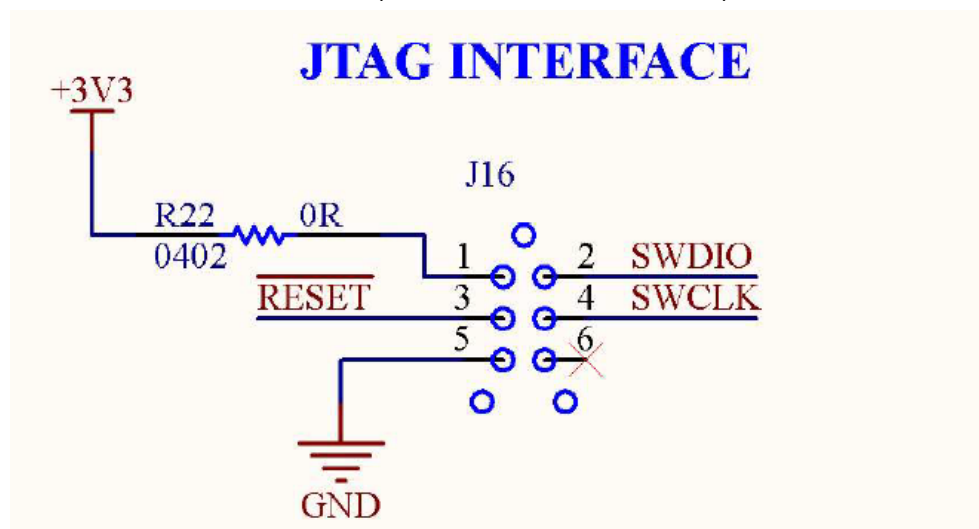


Figure 3 : PCB Bottom View

3.1. EVK's J-Link Connection

J-Link can be connected to the module through J16. This is only required when updating the MCM firmware via the EVK. (Used in Standalone mode)



3.2. Module Power up via EVK

Below are necessary connections to power up the module:

- Short pin 3 and 2 of JP1 (connect a jumper to pin 2 and 3 of JP1)
- Connect a jumper to J8
- Connect a jumper to J4
- Connect a jumper to J7
- Connect USB from your PC to J2

3.3. EVK Pins for the MCM Command Interface

These are the pins of the EVK that expose the UART based command interface of the MCM. The external controller/processor communicated with the MCM via this interface.

Terminal Name	Type	Description
UART-TXD	O	UART TX, Module to Host
UART-RXD	I	UART RX, Host to Module
COMMAND	I	Module Command Input
EVENT	O	Event Line
BUSY	O	Busy Line
GPIO3/ UART log TX	O	Log TX
GPIO4/ UART log RX	I	Log RX
GPIO1/ USER LED	O	User LED Indicator
GPIO 2/ USER BUTTON	I	User Button

3.4. Debug Log Interface

For EVK logs, user must connect FTDI to J12 as follow:

J12 pin 3 (GPIO3)-> FTDI RX

J12 pin 4 (GPIO4)-> FTDI TX

Parameter	Value
Baud Rate	115,200 bps
Data Bits	8
Stop Bits	1
Parity	None

4. MCM Command Manual

When the MCM is in the “Network Co-processor Mode”, the OxTech MCM module plays the role of a wireless communication modem (sidewalk modem) which is controlled by an external application controller/processor known as “application MCU” or “Host”. The application MCU (Host) communicates with the modem using the command interface which includes a UART port and some optional GPIO pins for synchronization. This section introduces instructions on how to interact with Oxit’s proprietary Amazon Sidewalk coprocessor application firmware.

4.1. Serial Port

UART is used to exchange commands and responses.

- UART-TXD: Module to Host
- UART-RXD: Host to Module

The following serial port parameters are used:

Parameter	Value
Baud Rate	9,600 bps
Data Bits	8
Stop Bits	1
Parity	None

4.2. Optional GPIO Synchronization Lines

4.2.1. COMMAND Line

Active-low, GPIO output line

This line is used to signal to the sidewalk module that the Application(Host) MCU is about to transmit a command. After driving the line low, the host has to wait until the BUSY line goes low before sending the first character. This allows the module to wake up and start reception. After the command is sent, the line must be driven high, at which point the Modem will process the command. If a valid command was received, the Modem will transmit its response.

4.2.2. BUSY Line

Active-high, GPIO output line

This line signals whether the module is busy or ready to receive commands. It is high while the Modem is busy and will go low as soon as the module is ready to receive a command. The

BUSY line will go high again after the command has been received and the COMMAND line has been released.

4.2.3. EVENT Line

Active-high, GPIO output line

This line signals to the Application MCU that the module has event data pending. The Application MCU can use the *GetEvent* command to retrieve such data.

4.3. Message Flow

Messages are exchanged via UART. Using the COMMAND, BUSY and EVENT GPIO lines for synchronization is optional. The MCM network co-processor mode firmware comes in two variants.

1. Firmware variant 1 - Using GPIOs for communication
2. Firmware variant 2 - Without using GPIOs for communication

The MCM comes pre-flashed with Firmware variant 2. The variant that uses GPIOs can be made available to users on special request.

The following section describes the message flow between Application MCU and the Modem.

Note:

When using GPIOs, the COMMAND line is set by the Application MCU; the BUSY and EVENT lines are set by the Modem.

4.3.1. Message Flow for Command and Response using GPIO Lines

Flow for sending commands and reading synchronous responses:

- a. Sending commands is initiated via the COMMAND line: the Application MCU pulls the COMMAND line low.
- b. The Modem drives the BUSY line low.
- c. The Application MCU sends the command using the UART RX line.
- d. The Application MCU must drive the COMMAND line high after the last character of the command has been transmitted. The Modem will only start interpreting the command after the COMMAND line is high.
- e. The Modem drives the BUSY line high.
- f. After validation the Modem will process the command and will send a matching response.

Notes:

- The Application MCU must not send a new command before the response has been fully received.

- If modem event data becomes available asynchronously, the modem will either signal this using the EVENT line (when using GPIOs) or will send a `NotifyEvent` command to the host (When not using GPIOs).

4.3.2. Message Flow for Modem Events using GPIO Lines

When the Modem has data available, it drives the EVENT line asynchronously. The event data can be retrieved via the `GetEvent` command using the message flow described above. The EVENT line is cleared when all pending events have been retrieved.

4.3.3. Message Flow for Command and Response without GPIO Lines

- a. The Application MCU sends the command using the UART RX line.
- b. The Modem will start interpreting the command.
- c. The Application MCU should wait to receive a response before sending another command.
- d. After validation the Modem will process the command and will send a matching response.

4.3.4. Message Flow for Modem Events without GPIO Lines

When the Modem has data available, it sends a `NotifyEvents` command to the application MCU (Host). The event data can be retrieved via the `GetEvent` command using the message flow described above. The `NotifyEvents` command is sent each time a new event is added to the queue.

Note:

In case when not using the GPIO lines, the application MCU (host) must be powered on before the module to not miss reset pending event command when the module first boots up.

4.4. Message Format

This section describes the packet format running on the UART interface, for input command packets, response packets and modem event commands. The maximum packet payload size of the input command is 255. The maximum packet payload size of the response packet/modem events command packet is 261 bytes.

4.4.1. Input Command Packets

All command messages consist of a command code that defines the operation to be performed, a payload length and optional payload, and a trailing checksum character which is the XOR of all previous bytes.

Field	Size (bytes)	Description
Cmd code	1	Command Code

len	2	Length of the payload field
payload	0 - 255	Parameters for the command as binary data
chk	1	Packet Checksum (xor over cmd code, len, payload)

The Modem begins analyzing the command message. After the packet structure is successfully verified (correct number of bytes and correct check character), the Modem will interpret the command and will send a response. If the packet structure is invalid, the Modem ignores the packet and will send a response with the return code.

4.4.2. Response Packets/Modem Event Command PACKets

The response format is similar to the command format; instead of the command code, it holds an output/return code that indicates the successful completion or an error condition of a command sent by Application MCU or can indicate a modem event command.

Field	Size (bytes)	Description
Output code/Return code	1	Return Code of the command/ Output command code
len	2	Length of the payload field
payload	0 - 261	Response data of the command as binary data or Respective payload of command initiated by the modem
chk	1	Packet Checksum (xor over return code, len, payload)

The output/return code bytes are described below:

Code	Type	Name	Description
0x00	return code	Ok	Command executed without errors
0x01	return code	Unknown	Command Unknown
0x02	return code	Not implemented	Command not implemented
0x08	return code	BAD CRC	Command CRC error
0x0A	return code	BAD Size	Command Size error
0x20	output code	NotifyEvents	Output command sent by the modem to indicate that host has pending events to read

4.5. Input Commands and Responses

4.5.1. Overview of Commands and Responses

The serial communication with the Modem is performed using alternating command and response messages. Response messages convey a return code and the result of the operation triggered by the preceding command message.

Name	Code	Description	Input	Output
GetEvent	0x00	Retrieve pending events		Type[1], count[1], eventdata[n]
GetVersion	0x01	GetVersion		Bootversion[4], Firmware version[3], Hardware version[3], Sidewalk version[3]
Reset	0x02	Reset Modem		
Factory Reset	0x03	Perform Modem Factory reset		
RequestTx	0x29	Transmit frame	data[n]	
FSK Link Request	0xF8	Request FSK Link		
CSS Link Request	0xF9	Request CSS Link		
BLE Link Request	0xFA	Request BLE Link		
BLE Connection Request	0xFB	Trigger BLE connection to be able to transmit data		

4.5.2. Commands Details

All input commands have a command code and, optionally, can have parameters to control the requested operation. All responses have a return code indicating success or failure.

Detailed parameters, return codes, and return data are described for every command in the following subsections. If not mentioned otherwise, the commands do not have parameters or return data.

Note:

All multi-byte fields representing integers contained in command or response payloads are encoded in little endian byte order (LSBF, least-significant-byte-first).

4.5.3. Input commands and their respective response

4.5.3.1. GetEvent

This command can be used to retrieve pending events from the Modem. Pending events are indicated by the EVENT line. The EVENT line will be de-asserted after all events have been retrieved and no further events are available. When not using GPIO synchronization lines, pending events are indicated by receiving *NotifyEvents* output command.

Input payload format:

Field	Size (bytes)	Payload
Cmd code	1	0x00
Len	2	0x0000
Payload	-	-
chk	1	0x00

Response payload format:

Field	Size (bytes)	Description
Return code	1	0x00: OK
Len	2	Payload length including type and count
type	1	Event type, see following table
count	1	Number of pending events that can be retrieved from modem
Payload	0 - 259	Event specific data
chk	1	checksum

Events description:

Name	Value (type field)	Payload	Description
------	----------------------	---------	-------------

Reset	0x00	2 bytes for reset count	Modem has been reset/Factory reset
Time Sync Success	0x02	-	Modem Successfully time synced with GW
Time Sync Failure	0x0a	-	Attempt to time sync with GW failed
TX Status	0x03	1 byte: 0x02 -> Successfully sent or 0x00-> Failed to send	Frame Transmission Status
Down Data	.0x04	5 to 259 bytes: <i>Data[0]</i> : RSSI <i>Data[1]</i> : SNR <i>Data[2] - Data[3]</i> : downlink sidewalk sequence number <i>Data[4...258]</i> : Down data received from server	Downlink data received
No Event	0xFF	-	This will be returned when application MCU try to get event and no event is pending

Notes:

- The application has a queue that can store events in a FIFO (First-In, First-Out) manner and that when the queue is full any new events will cause the oldest event (the first element) to be deleted to make room for the new event.
- On boot up, the modem will trigger a reset event, either by using *NotifyEvent* command or EVENT line.
- After sending the Reset/Factory Reset command the application must retrieve the reset event before the modem proceeds with reset.

4.5.3.2. GetVersion

This command returns the version of the bootloader (if it exists), the version of the installed firmware and the version of the Sidewalk stack.

Note: Oxit MCM application does not include bootloader. Bootloader version will be returned as 0.

Input payload format:

Field	Size (bytes)	Payload
Cmd code	1	0x01
Len	2	0x0000
Payload	-	-
chk	1	0x01

Response payload format:

Field	Size (bytes)	Description
Return code	1	0x00: OK
Len	2	13 bytes of payload
Payload	13	4 bytes: bootloader version 3 bytes: Modem FW version(major, minor, patch) 3 bytes: Modem HW version(major, minor, patch) 3 bytes: Sidewalk version(major, minor, patch)
chk	1	checksum

4.5.3.3. Reset

This command performs a reset of the Modem MCU. All transient state information (including session data) will be lost and the Modem will need to join the sidewalk network again. The Application MCU must retrieve the reset event by sending the GetEvent command so the modem can proceed with reset. The modem will reset after 500ms from retrieving the reset event.

Input payload format:

Field	Size (bytes)	Payload
Cmd code	1	0x02
Len	2	0x0000
Payload	-	-
chk	1	0x02

Response payload format:

Field	Size (bytes)	Description
Return code	1	0x00: OK
Len	2	0x0000
Payload	-	-
chk	1	0x00

4.5.3.4. Factory Reset

This command performs a factory reset of the Modem MCU. All transient state information (including session data) will be lost and the Modem will need to join the sidewalk network again. The Application MCU must retrieve the reset event by sending the GetEvent so the modem can proceed with reset. The modem will factory reset after 500ms from retrieving the reset event.

Notes:

- Factory reset command does not erase the manufacturing file.
- Factory reset command Deregister module from Sidewalk network.
- Factory reset command won't erase the reset counter.
- Factory reset command can only be sent if one of the sidewalk link types is established. If not established factory reset command will act as reset command and won't Deregister module.

Input payload format:

Field	Size (bytes)	Payload
Cmd code	1	0x03
Len	2	0x0000
Payload	-	-
chk	1	0x03

Response payload format:

Field	Size (bytes)	Description
Return code	1	0x00: OK
Len	2	0x0000
Payload	-	-

chk	1	0x00
-----	---	------

4.5.3.5. RequestTx

This command can be used to send uplink data to Sidewalk Network, the request will be queued and the frame will be sent as soon as bandwidth is available. A TxDone event is generated when the frame either has been sent successfully or not. The parameter of the TxDone event indicates whether the frame was sent (0x02) or not sent (0x00).

Notes:

- Maximum transmission for different link types:
 - BLE 255 bytes
 - FSK 200 bytes
 - CSS 19 bytes
- Modem should time sync with GW to be able to transmit/receive data.
- For BLE , User should initiate a Connection request using the *BLE Connection Request command* to be able to transmit data.

Input payload format:

Field	Size (bytes)	Payload
Cmd code	1	0x29
Len	2	Payload length of the data to be transmitted
Payload	1-255	Payload data
chk	1	checksum

Response payload format:

Field	Size (bytes)	Description
Return code	1	0x00: OK
Len	2	0x0000
Payload	-	-
chk	1	0x00

4.5.3.6. FSK Link Request

Input payload format:

Field	Size (bytes)	Payload
Cmd code	1	0xF8
Len	2	0x0000
Payload	-	-
chk	1	0xF8

Response payload format:

Field	Size (bytes)	Description
Return code	1	0x00: OK
Len	2	0x0000
Payload	-	-
chk	1	0x00

4.5.3.7. CSS Link Request

Input payload format:

Field	Size (bytes)	Payload
Cmd code	1	0xF9
Len	2	0x0000
Payload	-	-
chk	1	0xF9

Response payload format:

Field	Size (bytes)	Description
Return code	1	0x00: OK
Len	2	0x0000
Payload	-	-
chk	1	0x00

4.5.3.8. BLE Link Request

Input payload format:

Field	Size (bytes)	Payload
Cmd code	1	0xFA
Len	2	0x0000
Payload	-	-
chk	1	0xFA

Response payload format:

Field	Size (bytes)	Description
Return code	1	0x00: OK
Len	2	0x0000
Payload	-	-
chk	1	0x00

Notes:

- FSK/ CSS and BLE commands initiate requests to join the Sidewalk network. During the procedure, the modem will try to time sync with the gateway, a time sync failure or time sync success events will be generated to indicate if the modem successfully synchronized its time with the network or not.
- If the device is not registered, the modem will not be able to ever sync with GW. To make sure the device is registered, a BLE link request should be initiated first as the registration happens through BLE. After receiving the time sync event through BLE, users can switch between different link types and start communicating with the Sidewalk network.

4.5.3.9. BLE Connection Request

This command is used to open a BLE connection, this will allow the device to transmit frames.

Note:

- According to the Sidewalk protocol specification, the BLE connection will timeout after 30 seconds and disconnect itself, if no uplink or downlink is detected within the 30 seconds.

- Sending a downlink when Sidewalk is connected through BLE, will auto initiate a connection with GW as per Sidewalk specification and downlink will be received, no need to send BLE Connection request for downlinks.

Input payload format:

Field	Size (bytes)	Payload
Cmd code	1	0xFB
Len	2	0x0000
Payload	-	-
chk	1	0xFB

Response payload format:

Field	Size (bytes)	Description
Return code	1	0x00: OK
Len	2	0x0000
Payload	-	-
chk	1	0x00

4.5.4. LED Pattern

LED Patterns are implemented using the user LED on the board to visualize some events related to the module as follows :

Event	Pattern 1 count	Pause between pattern 1 and 2 in ms	Pattern 2 count
Boot up	3	500	4
Time sync failure	3	500	2
Time sync success	3	500	5
Uplink successfully sent	2	500	1

Downlink received	2	500	2
-------------------	---	-----	---

Note:

Pattern 1 and pattern 2 (both) are defined as a single blink (On + Off) = (200ms + 200ms).

5. Configure AWS CLI

AWS CLI can be used to send downlinks to sidewalk devices.

Install and configure the AWS CLI on your system:

- [Getting started with the AWS CLI](#)
- [Configure your AWS CLI](#)

For configuration after installing AWS CLI and making sure it is in the environment path, go to your terminal and type *aws configure* and input the following:

- Access Key ID and AWS Secret Access Key , you can create them after you create IAM User.
- For default region name choose **us-east-1**
- For default output format choose **json**

6. Create and Flash Manufacturing File

6.1. Create Manufacturing File

To create Manufacturing File for the Sidewalk devices in an automated way for prototyping follow this link:

<https://docs.silabs.com/amazon-sidewalk/1.0.0/sidewalk-developers-guide/application-development>

Make sure that the Simplicity Commander is already in system Path and python3 is installed in addition to AWS CLI.

You will basically have to clone the required scripts from the Github repository available in that link.

Make sure to modify *config.yaml* in the root directory after downloading the GitHub repository and change the DESTINATION_NAME to the corresponding destination used for uplink routing.

① **INFO** ①: To create destination, role and rule for your sidewalk devices follow this link <https://docs.aws.amazon.com/iot/latest/developerguide/iot-sidewalk-destination-create.html>

You will have to additionally modify USERNAME, PASSWORD, this corresponds to AWS sign in credentials, modify HARDWARE_PLATFORM too to SILABS.

Follow these steps in window subsystem linux terminal to create manufacturing files and wireless ids for your devices, in the terminal navigate to the root directory:

aws-iot-core-for-amazon-sidewalk-sample-app

1. `python3 -m pip3 install --user virtualenv`
2. `python3 -m venv sample-app-env`
3. `sample-app-env\Scripts\`
4. `Activate.bat`
5. Navigate back to root folder
6. `python3 -m pip3 install --upgrade pip`
7. `python3 -m pip3 install -r requirements.txt`
8. `python3 -m pip3 install pyjwt -t .\ApplicationServerDeployment\lambda\authLibs`
9. `python3 env_check.py`

You should run step 1 to 9 only one time, step 10 below can be performed each time you will need to create a new manufacturing file for new devices.

10. `python3 EdgeDeviceProvisioning/provision_sidewalk_end_device.py -instances <number_of_instances>`

The script creates a directory structure as follows in the EdgeDeviceProvisioning folder:

DeviceProfile_7c51bc6b-0556-2083-6f0d-aeb750c94508

```
├── DeviceProfile.json
├── WirelessDevice_db57b1c9-22ad-c052-07ae-1e1cae9bb384
│   ├── SiLabs_MFG.nvm3
│   ├── Silabs_xG21.s37
│   ├── Silabs_xG24.s37
│   └── WirelessDevice.json
```

6.2. Flash Manufacturing File

Connect J Link to Oxite Sidewalk Module then Flash Manufacturing file *Silabs_xG24.s37* using J Flash lite.

7. Create Custom Application (Standalone mode)

This section presents instructions on how to use the OxTech Sidewalk module with Simplicity Studio to test custom firmware.

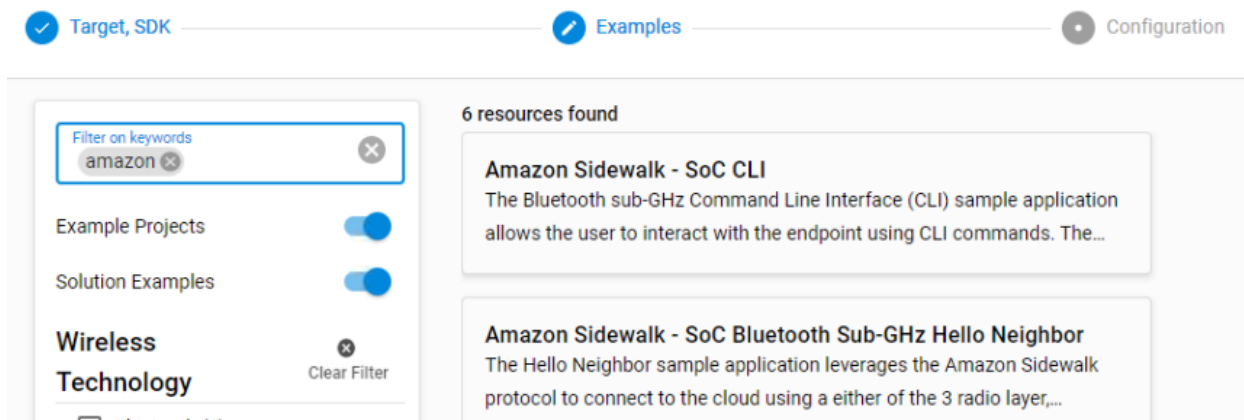
7.1. Create an Amazon Sidewalk Project for Oxit Sidewalk Module

Open Simplicity Studio 5.

- Download and install GSDK 4.3.1 with its respective Sidewalk extension 1.2.0 .
- Click on File / New / Silicon Labs Project Wizard
- Choose appropriate target board,device, SDK and IDE/toolchain to use for the project.
 - Target Boards: Custom Board
 - Target Device: EFR32MG24B220F1536IM48
 - SDK: Gecko SDK Suite 4.3.1
 - IDE / ToolchainL Simplicity IDE / GNU ARM v10.3.1



- Click on Next
- In the Examples filter on keyword Amazon
- You will have two examples for Amazon Sidewalk

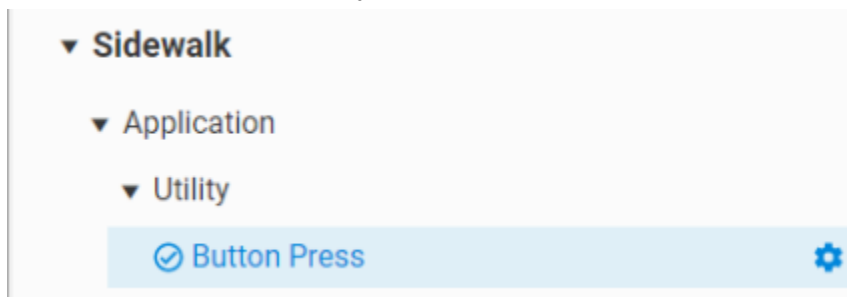


- You can choose one of the examples, for this demo choose Amazon Sidewalk - SoC Bluetooth Sub-GHz Hello Neighbor
- Click Next and Finish

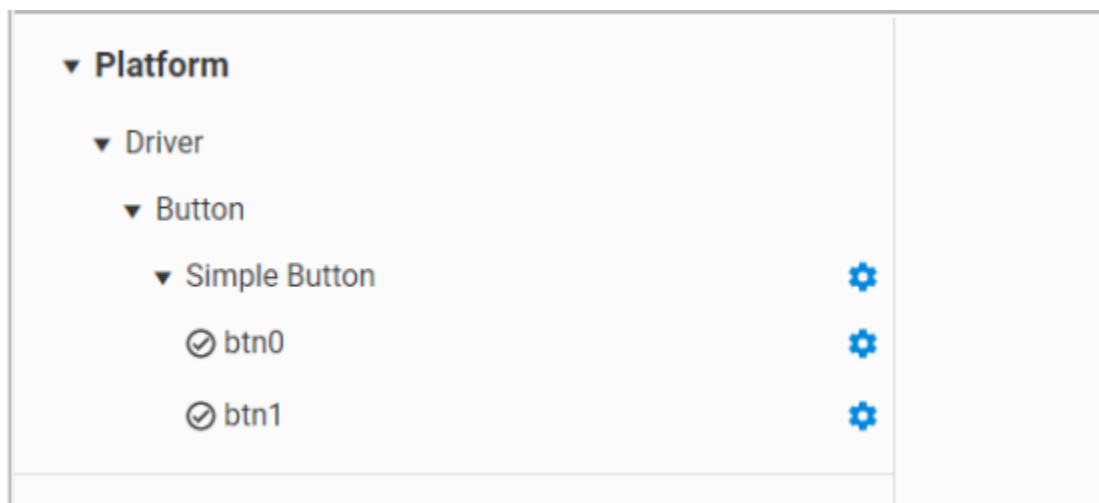
7.2. Modify Amazon Sidewalk Sample Application

As Amazon sample applications are based on Silicon labs wireless starter kit, there are few modifications that need to be done:

- Go to the project .slcp / Software components and follow the below instructions:
 - Uninstall Button Press Utility



- Uninstall btn0 and btn1



- Modify SPIDRV EUSART exp pins to match Oxit Sidewalk module Sx1262 SPI pins
- Module: EUSART1
- CS: PA05
- SCLK: PA06
- TX: PA07

○ RX:

PA08

SPI clock mode
SPI mode 0: CLKPOL=0, CLKPHA=0

SPI master chip select (CS) control scheme.
CS controlled by the application

SPI slave transfer start scheme
Transfer starts immediately

SL_SPIDRV_EUSART_EXP

Selected Module
EUSART1

CS
PA05

CTS
None

RTS
None

RX
PA08

SCLK
PA06

TX
PA07

○ Install LFXO


amazon_sidewalk_soc_hello_neighbor OVERVIEW **SOFTWARE COMPONENTS** CONFIGURATION TOOLS


Filter components by ☒ Configurable ☐ Installed ☐ Installed by you ☐ SDK Extensions ☐ Quality

▼ Services

▼ Device Initialization

▼ Peripherals

⌚ Low Frequency Crystal Oscillator (LFXO) 

Low Frequency Crystal Oscillator (LFXO)  Configure

Description
Initiate the component for Low Frequency Crystal Oscillator (LFXO) initialization based on the provided configuration, such as crystal accuracy and startup time delay.

○ Install GPIOINT

▼ Platform

▼ Driver

⌚ GPIOINT

GPIOINT

Description

- Go to config folder in the project explorer and open app_gpio_config.h and modify SX1262 pins according to Oxite Sidewalk Module as follows:

```

7
8 // BUSY on PA00
9 // Used to indicate the status of internal state machine
10 #define SL_BUSY_PIN 0
11 #define SL_BUSY_PORT gpioPortA
12
13 // ANT_SW on PA04
14 // External antenna switch to control antenna switch to RECEIVE or
15 // TRANSMIT.
16 #define SL_ANTSW_PIN 4
17 #define SL_ANTSW_PORT gpioPortA
18
19 // DIO1 on PA09
20 // IRQ line from sx126x chip
21 // See sx126x datasheet for IRQs list.
22 #define SL_DIO_PIN 9
23 #define SL_DIO_PORT gpioPortA
24
25 // SX NRESET on PD05
26 // Factory reset pin. Will be followed by standard calibration procedure
27 // and previous context will be lost.
28 #define SL_NRESET_PIN 5
29 #define SL_NRESET_PORT gpioPortD
30

```

- Go to `app_init.c` and `app_process.c` files and remove `#include "app_button_press.h"`
- Go to `app_init.c`, `app_init(void)` function and remove `app_button_press_enable();` callback.

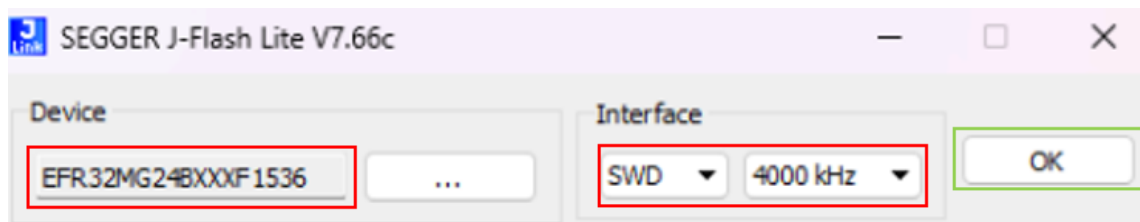
7.3. Build your Amazon Sidewalk Application

After modifying the Amazon Sidewalk application you can now successfully build the project and build your application.

7.4. Flash your Amazon Sidewalk Application

After installing the J-Link Software tools, in the Search bar search for J-Flash Lite and open it.

- For the device section choose EFR32MG24BXXXF536
- For the interface choose SWD and 400 kHz as speed
- Click ok



Next, you can simply browse your application `.s37` output file that does not require specifying the Prog. addr and click on Program Device.

Notes:

- Flashing custom applications will override MCM firmware.
- Erase chip button will erase all MCU content including MFG file.
- The user can erase MFG at any time by erasing the chip and can flash the same MFG again or a different one within the same OxTech MCM EVK.
- Erasing MFG will Deregister the Sidewalk module. The User should proceed with registration again.

7.5. Check Amazon Sidewalk Application logs

To monitor application logs you can simply connect J Link to the board and open J Link RTT viewer.

- Open the J-Link RTT Viewer.
- In the **Configuration** panel, **Connection to J-Link** RTT section, select **USB**.
- In the Specify Target Device list, select the connected part EFR32MG24Bxxx1536.
- In the **Target Interface & Speed** panel, select **SWD** and **4000 kHz**.
- In the **RTT Control Block** panel, select **Auto Detection**.
- Click **OK**.

A terminal opens and the Sidewalk application traces are output to it.

Note:

J Link RTT Viewer is used as default to view logs in Amazon Sidewalk example project.

8. Monitor Uplink Data

To monitor data sent from device to cloud follow below:

1. Login to your AWS account and click on IAM user and enter the user credentials.
2. Go to the [MQTT test client](#) in AWS.
3. Subscribe to the topic created when creating the destination for the device.
4. You should see messages come up on the MQTT broker.

9. Send A Downlink

After installing and configuring AWS CLI on your system, the user should be able to run the AWS command to send payload data.

1. Run this command from the terminal:

```
aws iotwireless send-data-to-wireless-device  
--id=39c3d7af-6bf4-4dca-9858-465f7cc7e681 --transmit-mode 0 --payload-data="AQ=="  
--wireless-metadata "Sidewalk={Seq=1}"
```

 - a. Seq=x, **x should be unique any time you run this command**

- b. `-id` should be the “Wireless Device ID” of the device in this case
`39c3d7af-6bf4-4dca-9858-465f7cc7e681`
- c. `-payload-data` : base64 format of the payload to send in that case “AQ==”, it represents 0x01 in hex.

10. Generale Notes

- Boards come pre-flashed with Oxit's proprietary Amazon Sidewalk coprocessor application firmware that does not use GPIO synchronization lines.
- Boards do not come pre flashed with a manufacturing file.
- Manufacturing files need to be flashed separately to be tied to the desired test account.
- Boards can be re-flashed at any time to host any custom firmware and manufacturing file.
- Flashing the manufacturing file does not erase Oxit's proprietary Amazon Sidewalk coprocessor application firmware.
- Fully erasing the device or flashing custom application will cause erasing Oxit's firmware.

11. Known Issues

- Application will crash if CSS is started before FSK. This is a known issue with Sidewalk proprietary stack. The user must start FSK before CSS at first to prevent application crashes.
- Application will crash if no MFG is flashed and the user sends FSK or CSS link request, this is also a known issue in Sidewalk stack.

12. CONFIDENTIAL & PROPRIETARY

This document contains confidential and proprietary information belonging to Oxit LLC. It is intended solely for the use of the individual or entity to whom it is addressed and may contain privileged and confidential information. Any unauthorized review, use, disclosure, distribution, or copying of this document, in whole or in part, is strictly prohibited. If you have received this document in error, please notify the sender immediately and delete the original message and any attachments.

All intellectual property rights, including but not limited to copyrights, trademarks, trade secrets, and patents, in the contents of this document are owned by Oxit LLC. or its respective owners. No part of this document may be reproduced, transmitted, or distributed in any form or by any means, electronic or mechanical, including photocopying, recording, or any information storage and retrieval system, without prior written permission from Oxit LLC.

By accepting and reviewing this document, you agree to be bound by the terms and conditions stated herein. Failure to comply with these terms may result in legal action and remedies under applicable laws.

This document does not constitute an offer, agreement, or commitment of any kind, unless expressly stated otherwise. The views and opinions expressed in this document are those of the individual author and do not necessarily reflect the official policy or position of Oxit LLC.

Oxit LLC,

3131 Westinghouse Blvd, Charlotte, NC 28273

<https://oxit.com/>