Deviceworx Technologies Inc.

xGATWEAY User's Guide

Table of Contents

Revisions

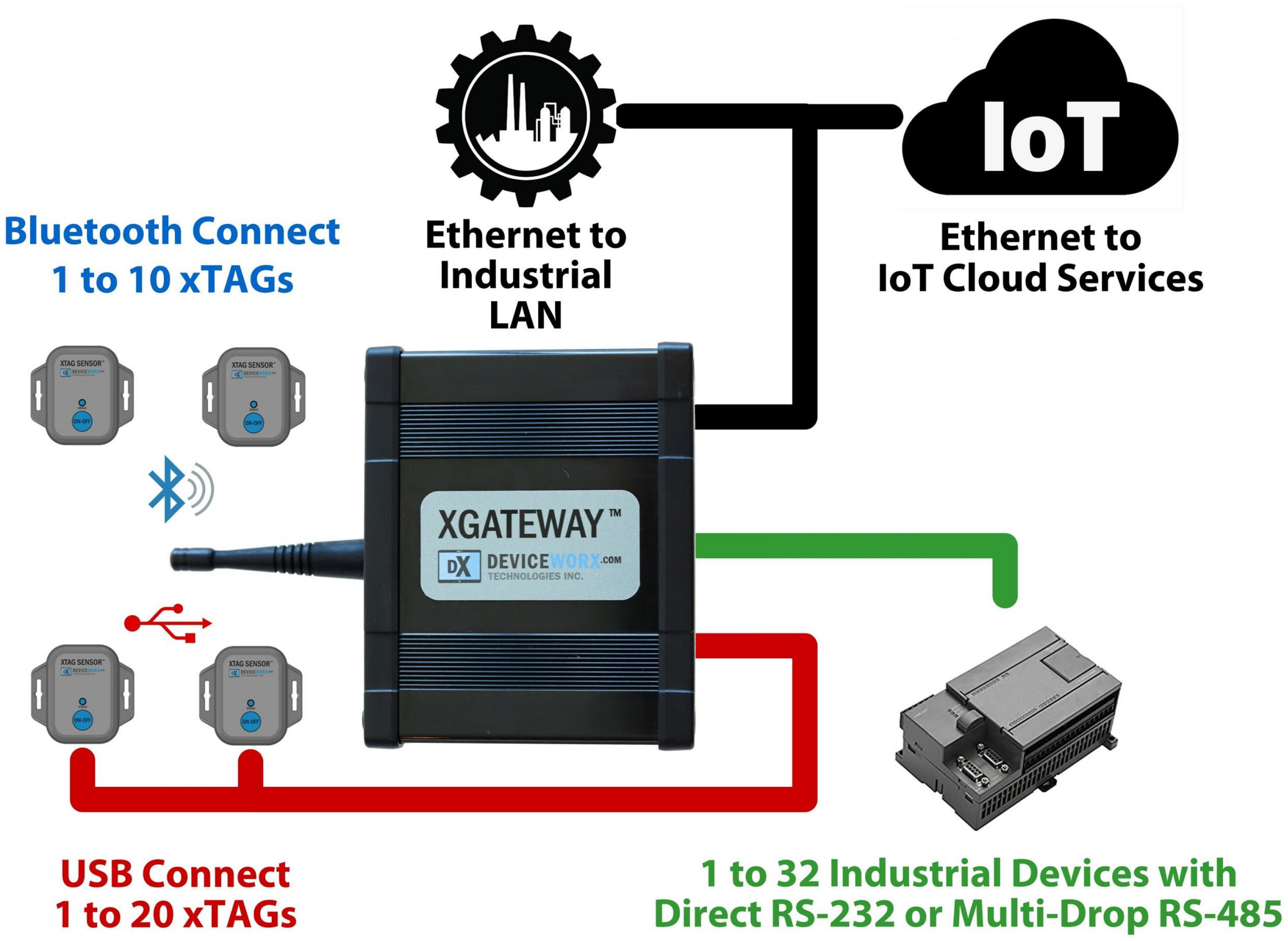
Rev	Author	Notes	Date
1.0	Mark Janke	Original Draft.	Aug 13, 2020
1.1	Mark Janke	Added clarifications on return values and notes as per customer Q's.	Aug 25, 2020
1.2	Mark Janke	Updated for xtagd socket connections and disconnection changes.	Aug 27, 2020
1.3	Mark Janke	Changed stream start cmd to support start only after motion or no-motion. Added wiring diagrams supporting RS-232/422/485 comms.	Sept 17, 2020
1.4	Mark Janke	Updated switch pics to new default of slew rate unlimited.	Sept 28, 2020
1.5	Mark Janke	Updated to include xtagbled and xtagusbd support details.	Oct 6, 2020
1.6	Mark Janke	Updated to support release of xTAG firmware and xGATEWAY firmware and daemons.	Nov 12, 2020
1.7	Mark Janke	Removed typo within xTAG Accelerometer Acquisition Config (0x14).	Nov 18, 2020
1.8	Mark Janke	Added plugged stream msg details (0x17)	Nov 19, 2020
1.9	Mark Janke	Clarified ODR <= 100 s/s on delayed stream.	Dec 14, 2020
1.10	Mark Janke	Updated SSH login details.	April 4, 2021

Document Overview

This document serves to provide xGATEWAY (xGw) user's with information required to setup and use xGw devices. Supplemental marketing collateral provides features and benefits information for xGw and partner xTAG devices.

Functionality

The xGw is an Industrial-grade Internet of Things (IIoT) Gateway device. It is used to collect data from Deviceworx xTAGs via BLE (up to 10 xTAGs) or usb (up to 20 xTAGs). It can push xTAG data to the cloud or provide access to it on a LAN via a Socket Interface or API. Additionally, the xGw can send and receive data from up to 32 legacy industrial devices that utilize industry standard interfaces including RS232, RS422 and RS485. This connectivity can provide legacy industrial devices with cloud-based status, reporting and even control to retrofit IoT functionality into legacy hardware. Lastly, the xGw provides IoT" Edge" functionality (i.e. it can be used to condition data including statistical analysis, alarming and local control).

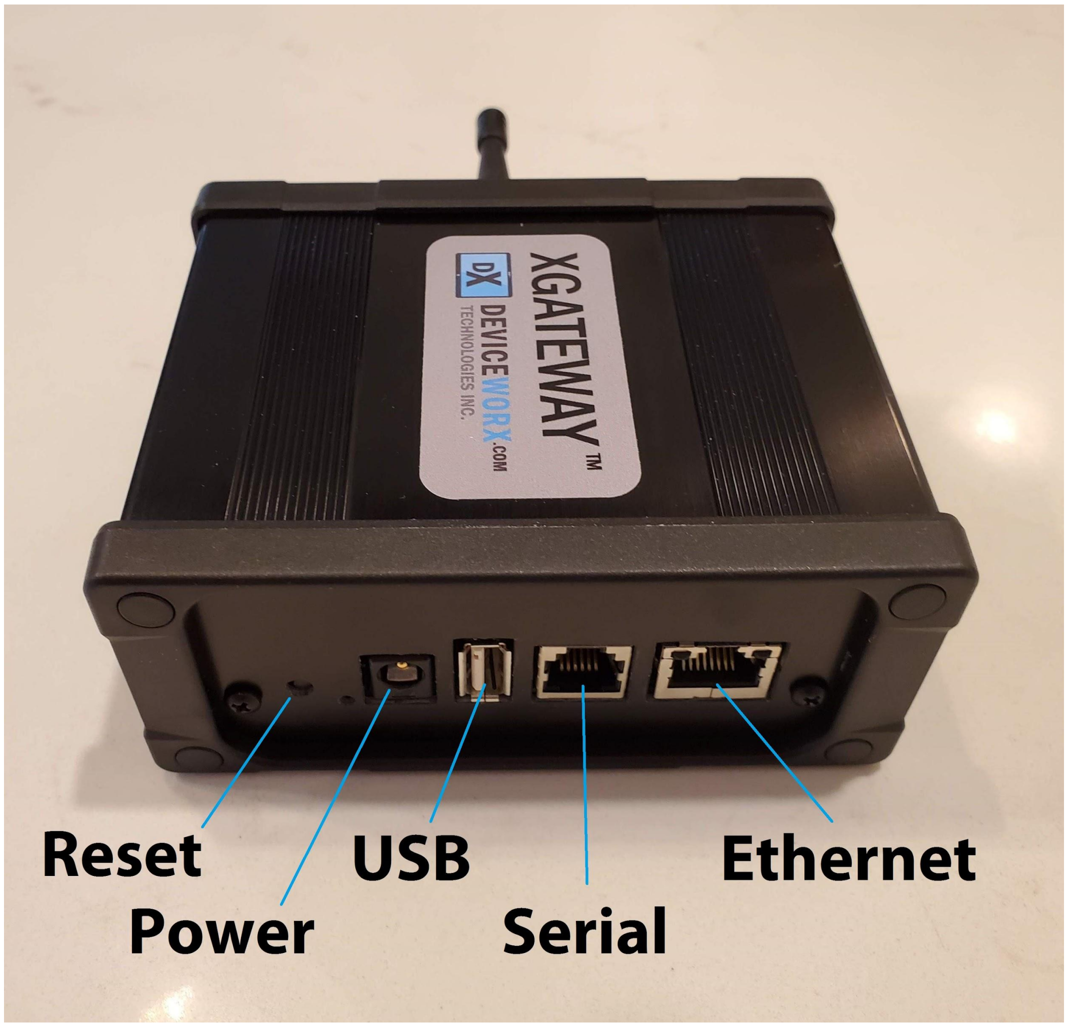


Specifications

- Weight: 340 g or 12 oz
- Dimensions: 110mm (4.33")wide, 90mm (3.5") long, 50mm (2") tall
- Panel mount enclosure
- Bluetooth: BLE 5.0 at 2 Mbps with 100 m (300’) range.
- USB: Host at 12 Mbps (up to 100 m with USB extension).
- Ethernet: RJ45 10/100/1000 mbps. Supports Modbus TCP and many other protocols.
- Serial: RJ11 RS-232 (400 kbps to 15m) or RS422/RS485 (230 kbps to 50m). Supports Modbus RTU, DirectNET and many other protocols.
- Operating Temperature: -35 °C to 85 °C

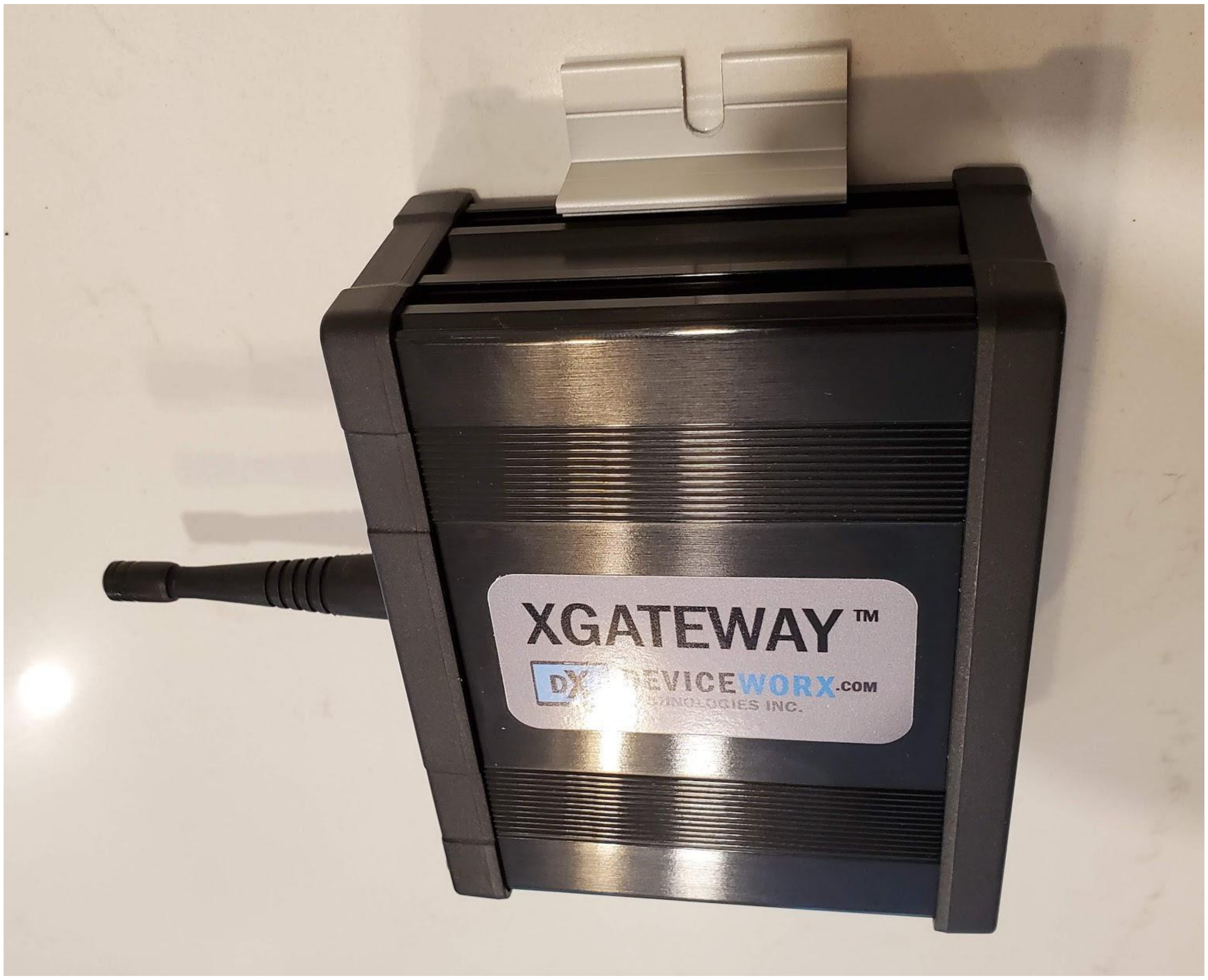
Connections

Device connections are shown within the picture below.



Mounting

The xGw is typically mounted within a field enclosure on that enclosure's back panel. To facilitate this, accessory mount tabs are provided with each xGw upon customer request.



To install the tabs, simply remove the top end cap of the enclosure that holds the antenna, along with its plastic cover. This is done by removing the 4 screws holding both the plastic cover and end cap. Then, simply slide the tabs into the enclosure side channels (each side) and replace the top end cap. Note that the antenna lead should not be removed while the end cap and plastic cover have been removed. The tabs will be loose and slide up and down within the side channels until the enclosure is mated to a back panel and secured by screwing the tabs to the back panel by attaching it using appropriately sized machine screws that fit within tab slots.

Linux OS Support

The xGw runs Debian Linux. It may be used by operators without any Linux access (i.e. when using Deviceworx or partner software on the device). When required, however, customers may access the Linux OS and run their own apps on the xGw.

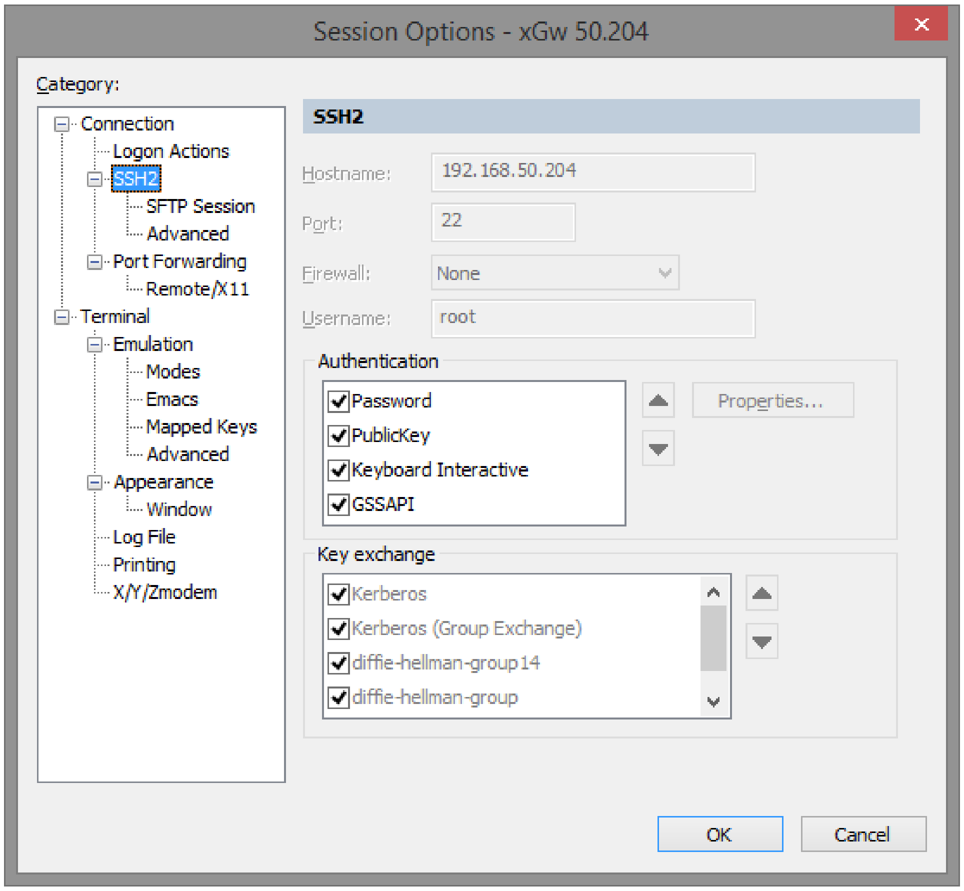
Operators can use package managers to add/remove Linux package support for their apps. The xGw ships with apt and NPM package managers. Consult online documentation for each of these tools for specifics on how they can be used.

Remote Linux access, available app startup and management functions and the pre-installed Deviceworx socket interface (for interacting with xTAGs via BLE or USB) are described within sections that follow.

SSH Access

To connect to an xGw using SSH, ensure that the xGw has a good Ethernet LAN connection. Next, determine the IP address of the xGw by reviewing DHCP allocated or statically allocated addresses on your LAN..

Setup a SSH client connection to the xGw using IP address, standard SSH port 22, matching username and password "root" and authentication as shown in the example SecureCRT setup below:



Hostname: 192.168.50.204
Port: 22
Firewall: None
Username: root

Authentication
- [✓] Password
- [✓] PublicKey
- [✓] Keyboard Interactive
- [✓] GSSAPI

Key exchange
- [✓] Kerberos
- [✓] Kerberos (Group Exchange)
- [✓] diffie-hellman-group14
- [✓] diffie-hellman-group

After saving authentication, no user and password entry will be required upon SSH connection. Ensure that the SSH client is setup to utf-8 character encoding (sometimes called "locale"). This is required by some terminal apps such as "top".

SFTP Access

Any industry standard SFTP client can be used to send/receive files to/from the xGw. The best candidate is FileZilla. Ensure that port 22 is used when setting up an SFTP client and that user "root" and password "root" are used.

Startup and Process Management

If xGw operators choose to deploy their own app(s) to the device, there are options to support automatic app launch and management.

The /etc/rc.local file supports startup commands that will execute at the end of the device startup cycle. Edit this file using vi or install another text editor if required. Add commands as necessary.

The pm2 process manage is installed on the xGw by default. Note that it will already be setup to start the xtagd daemon. Do not alter pm2 setup relating to this daemon, but feel free to add pm2 management support for your app(s). There are plenty of online resources describing how to use pm2 from a command line over an SSH link.

Wiring Diagrams

The xGATEWAY primary serial device can be connected to a single secondary serial device via RS-232, or to a "chain" of up to 32 devices via RS-485/422. Separate sections below outline how to configure the xGATEWAY for distinct serial communication configurations as well as how to wire up serial connection within these configurations.

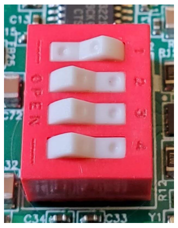
Note that color coding of each diagram matches a common 6 core RJ-11 standard. This standard is implemented within a low-cost cable sold by Digikey (Part Number: A3662R-07-ND) and accessible here:

ASSMAN AT-S-26-6/6/B-7-OE Cable

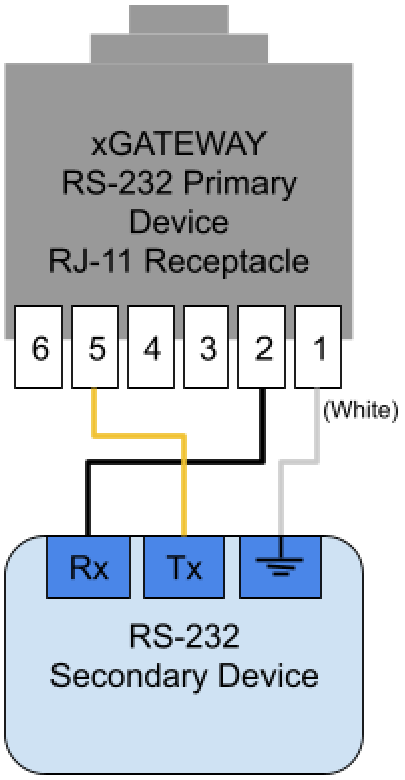
Note that diagrams that follow show the receptacle pinout on the xGATEWAY. The pinout for the plug on the cable will be numbered in reverse order (Pin 1 on the left).

Point to Point RS-232 Full Duplex Wiring

To configure the xGATEWAY to support full duplex RS-232, set on board switches to match that in the picture below (POS1 OPEN, POS2, POS3 & POS4 CLOSED).

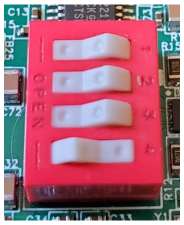


The diagram below shows how to connect a single RS-232 secondary device to the xGATEWAYs serial RJ-11 Connector.

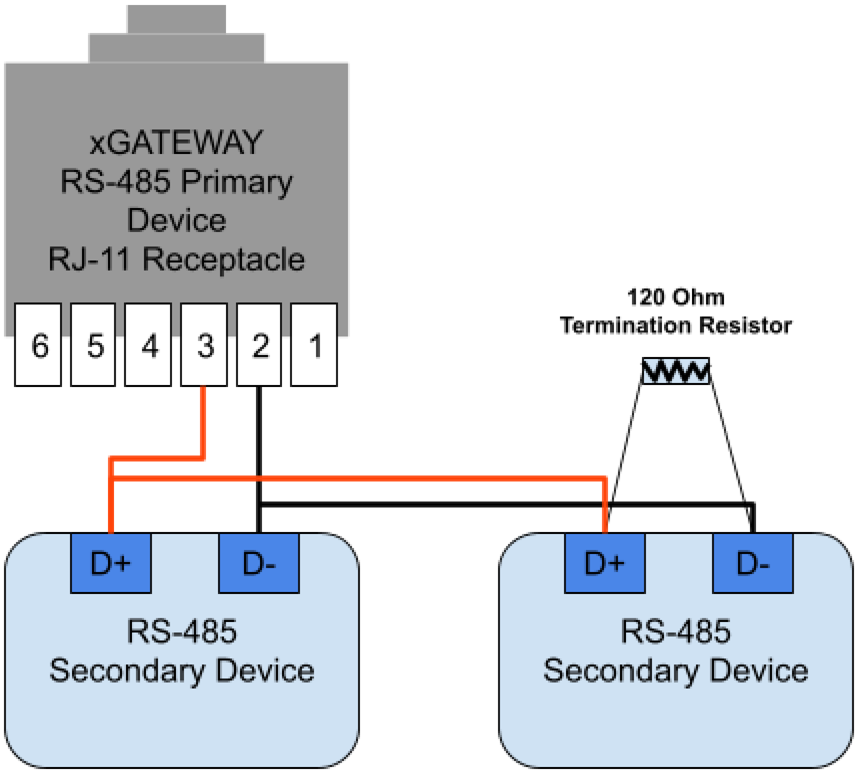


Multi-Drop RS-485 Half Duplex Wiring

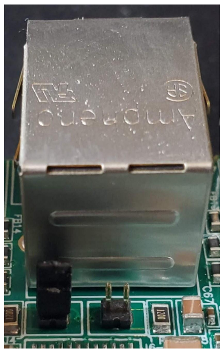
To configure the xGATEWAY to support half duplex RS-485, set on board switches to match that in the picture below (POS1-3 OPEN, POS4 CLOSED).



Many RS-485 secondary devices including panel controllers, Programmable Logic Controllers (PLCs) etc, utilize simple half-duplex wiring. The diagram below shows how to connect 1 or more RS-485 secondary devices (up to 32) to the xGATEWAY's serial RJ-11 Connector.

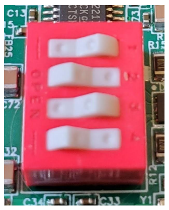


Note that the 120 Ohm termination resistor shown above is only required for very long connections or to help limit EMI (a shielded cable grounded at one end may help with EMI as well). When this termination resistance is added to the last multi-drop connection, also enable the 120 Ohm line termination resistance within the xGATEWAY by installing J4 on the board (installed by default behind the RJ-11 connector) as shown here:

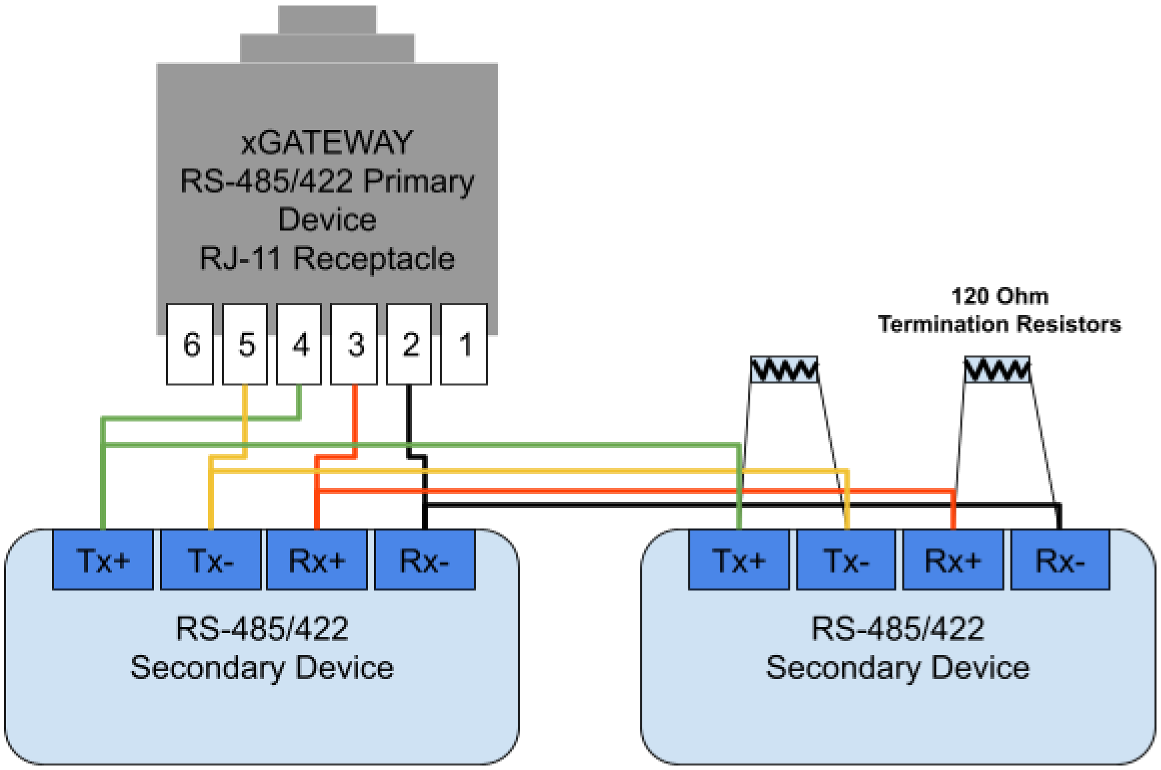


Multi-Drop RS-485/422 Full Duplex Wiring

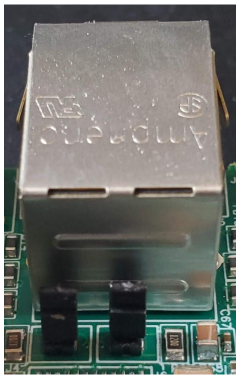
To configure the xGATEWAY to support full duplex RS-485/422, set on board switches to match that in the picture below (POS1 OPEN, POS 2 CLOSED, POS3 OPEN, POS4 CLOSED).



The diagram below shows how to connect 1 or more RS-485/422 secondary devices (up to 32) to the xGATEWAYs serial RJ-11 Connector.



Note that the 120 Ohm termination resistors shown above are only required for very long connections or to help limit EMI (a shielded cable grounded at one end may help with EMI as well). When this termination resistance is added to the last multi-drop connection, also enable the 120 Ohm line termination resistances within the xGATEWAY by installing **J4 and J6** on the board (installed by default behind the RJ-11 connector) as shown here:



Socket Interface through the xTAG Daemons

Two xTAG daemons are pre-installed on the xGw. The first (“xtagbled”) supports interacting with BLE-connected xTAGs. The second (“xtagusbd”) supports interacting with USB-connected xTAGs. These daemons implement a client interface to xTAGs using Berkeley Sockets on the xGw Linux operating system. The daemon client connections will be via standard Berkeley Socket techniques and will not be described within this document. Following socket connections, daemon clients will send commands to the daemons and handle responses. Note that xtagbled client devices will connect to the daemon locally or remotely using ports 3240 and 3241. Note that xtagusbd client devices will connect to the daemon locally or remotely using ports 3242 and 3243. Synchronous commands with responses will be handled on the “Primary port” 3240 (xtagbled) or 3242 (xtagusbd). Stream response messages from the xGw will be sent out on the “Stream port” 3241 (xtagbled) or 3243 (xtagusbd). See Accelerometer Acquisition Stream Data (Responses Only) (0x17) for more detail on streaming. Daemon clients must connect to a Primary port before issuing any synchronous commands and must connect to a Stream port before data streaming is started. Note that if no streaming is required, only a connection to a Primary port is required.

It is strongly recommended that xtagd clients disconnect all sockets before restarting xtagd or reconnecting to it. Failure to release socket connections can require a socket timeout (approx 1 min).

To serve as an example to client app developers, Devicworx has created an xtagd client test app (“xGwTestApp). Deviceworx will provide this Win32 console app (project with sources) to xGw operators upon request.

Daemon Startup and Management

The daemons start automatically when the xGw boots and it will be automatically restarted if it dies (via pm2 control). If daemon clients ever need to reset the interface, restarting the appropriate daemon is the best option. Simply kill it using Linux command line functions, wait 5 seconds for pm2 to restart it, and then reconnect to it using a socket connection.

If either daemon will not be used - remove it from pm2 control using:

pm2 delete /root/xtagusbd or pm2 delete /root/xtagbled

… then restart the xGw. pm2 will not start the deleted daemon anymore.

To add either daemon (or both), add it to pm2 control using:

pm2 start /root/xtagusbd and/or pm2 start /root/xtagbled .. to start the daemon(s).

pm2 startup … to create a startup script or update a script.

pm2 save .. to save the current pm2 managed daemon list so that it will be restarted.

At any time, see daemons currently managed by pm2 using: pm2 list

To specify a command line argument to xtagbled (see the BLE communication index description below), use pm2 start /root/xtagbled -- 2 … the “-- 2” specified command line arg of 2 is passed to xtagbled.

BLE Communication Optimization

A BLE communication “index” must be selected using a single xtagbled command line argument (0, 1, 2, or 3). This single digit index is used to optimize BLE connections for range, power and speed. The index selects both a communications baseband Phy (“f-eye”) as well as other parameters that influence speed and power consumption.

- 0 (Slow - Far): This index supports very long range (300+ m) communications with a slower baseband of 125 kbps. Note that this Phy uses slightly more power than others to maintain connections and should only be selected if very long range communications is required or communications challenges exist within a noisy environment. Note that this index only supports 100 samples per sec or slower.

- 1 (Less Slow - Medium Range - Low Power): This default index supports standard Bluetooth Class 1 communications to 100 m and a reasonably fast baseband of 1 Mbps. It uses less power than the next index. Note that this index only supports 100 samples per sec or slower.
- 2 (Faster - Medium Range). Select this option when speed is prioritized over power. It supports the same range and baseband as index 1, with higher power draw. Note that this index only supports 800 samples per sec or slower.
- 3 (Fastest - Lower Range). This index supports the fastest baseband of 2 Mbps, but is limited to a range of 70 m or less (approximately). This index uses the same power as 2. This index should be selected for 800 (when range allows) or 1600 samples per second.

Typical Client Operations

The following command order of execution applies to xtagd client applications (applicable port in brackets).

1. Interact with xGw if required and read its meta data (3240).
2. List connectable BLE xTAGs nearby or physically connected USB xTAGs (3240).
3. Connect to xTAGs as required using their IDs (3240).
4. Config xTAGs for acquisition (3240).
5. Start acquisition for 1 to many xTAGs (3240).
6. Handle data stream responses (3241).
7. Stop acquisition for 1 to many xTAGs (3240).
8. Shutdown the xtagd if required (3240).

Each of these steps is demonstrated within the daemon client test app discussed above.

Command Formatting

All commands (cmd) will have the following format.

<cmd byte>,<cmd len byte>,<cmd param byte 1>..<cmd param byte n>

Notes

1. <cmd len byte> stipulates total cmd length including the <cmd byte> and all bytes following it.
2. <cmd param byte 1>...<cmd param byte n> are optional and will only be used in commands where passing data to xtagd is required.

Command Response Formatting

Command responses will have the following format.

<cmd byte>,<resp len byte>,<resp error>,<resp data byte 1>…<resp data byte n>

Notes

1. The <resp len byte> stipulates total resp length including the <cmd reflection byte> and all bytes following it.
2. <resp error> will pass an error value or status value indicating successful command execution.
3. <resp data byte 1>…<resp data byte n> may only be returned for some commands and is not required when a response is simply passing back status via the <resp error> byte.

Error Responses

Error responses will be consistent for all commands described in this document. Specifically, they will be 3 bytes long and match this format:

<cmd reflection byte>,<len byte=0x03>,<Error Byte>

Valid <Error Byte> values will be described for each command.

xtagd Socket Interface Commands

xGATEWAY BLE Module Metadata Read (0x01)

Read attributes of the xGATEWAY.

Command Format <Cmd=0x01>,<Len Byte = 0x02>

Response Format<Cmd=0x01>,<Len Byte = 0x09>,<Error Byte>,<RunSec MS Byte>,<RunSec 2nd MS Byte>,<RunSec 3rd MS Byte>,<RunSec LS Byte>,<SW Rev MS Byte>,<SW Rev LS Byte>

Error Byte values:

- 0x00: Success
- 0x01: Bad argument. Bad length. Bad daemon (ie. no support in xtagusbd).

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. RunSec Bytes make up an unsigned long denoting the number of sec since xGATEWAY firmware start. Can be used for troubleshooting as xGATEWAY restarts only occur upon xGATEWAY firmware failures.
3. SW Bytes denote a raw unsigned word with implied decimals indicating xTAG software or firmware revision. E.g. 0x2775 = 10101 denoting rev 1.01.01.

List xTAGs (0x02)

Return a list of connected and connectable xTAGs. If USB is used, response will be quick as device IDs will be stored when they are discovered upon their connection. If BLE is used, an xTAG scan will be required to ensure that the list of connectable xTAGs is accurate. This may take 5-20 sec.

Command Format <Cmd=0x02>,<Len Byte = 0x03>,<Timeout Byte>

Response Format<Cmd=0x02>,<Len Byte>,<Error Byte>,<xTAG1 Con Status>,<xTAG1 MS Byte>,<xTAG1 2nd MS Byte>,<xTAG1 3rd MS Byte>,<xTAG1 4th MS Byte>,<xTAG1 5th MS Byte>,<xTAG1 LS Byte> …. <xTAGn Con Status>,<xTAGn 6 byte address>

Error Byte values:

- 0x00: Success
- 0x01: Bad argument. Bad length.
- 0x02: Error processing the command.

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. Value of success will be returned - even if no xTAGs could be found (i.e. none connected on USB and none could be discovered over BLE).
3. Regardless of physical layer connection type, xTAG IDs will be in the form of MAC addresses with 6 bytes (e.g. 11:22:33:44:55:66 represented in Hex data format).
4. The< xTAGn Con Status> byte will be non-zero when connected and zero when disconnected.
5. Max returned xTAG IDs is 20 (USB or BLE) requiring a response len of (20*(6+1))+3=143 bytes.
6. <Timeout Byte> valid range is 5-20 sec with default of 10 sec if a value outside of this range is specified (e.g. including 0x00). On systems where BLE range is short and reception is consistent, shorter scan timeouts may be used. More than 20 sec should never be required. When a physical USB connection is used, this value is ignored but must be set (cmd len must always be 3 bytes for consistency).

xTAG Connect (0x03)

Connect to a specified xTAG. Multiple connections are supported at once.

Command Format <Cmd=0x03>,<Len Byte = 0x08><xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>
Response Format<Cmd=0x03>,<Len Byte = 0x03>,<Error Byte>

Error Byte values:

- 0x00: Success
- 0x01: Bad argument. Bad length or xTAG provided may not have been scanned.
- 0x02: Connection failure. BLE connect failed. Not uncommon - try again at least 3 times.

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. Connections may take 5 sec on BLE and can fail requiring that the xtagd retry a connection, so allow up to 15 sec for a connection response over BLE. Connections on USB will take less than 1 sec.
3. If an xTAG is connected, calling this will do nothing (ie. no reconnect) and return “Success” quickly.
4. The response does not include the 6-byte xTAG Address. It should not be required when this command is executed sequentially for multiple tags (i.e. wait for a response from each connect command before issuing a connect command for another xTAG).

xTAG Disconnect (0x04)

Disconnect from a specified xTAG.

Command Format <Cmd=0x04>,<Len Byte = 0x08><xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>

Response Format<Cmd=0x04>,<Len Byte = 0x03>,<Error Byte>

Error Byte values:

- 0x00: Success
- 0x01: Bad argument. Bad length or xTAG provided may not have been scanned.
- 0x7F: General Error.

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. Disconnect time is dependent on Physical Link. For USB, this will be < 1 sec. For BLE, expect < 5 sec.
3. If the xTAG was already disconnected, this command will do nothing and Success will be returned.
4. The response does not include the 6-byte xTAG Address. It should not be required when this command is executed sequentially for multiple tags (i.e. wait for a response from each disconnect command before issuing a disconnect command for another xTAG).

xTAG Metadata Read (0x05)

Read attributes of a connected xTAG.

Command Format <Cmd=0x05>,<Len Byte = 0x08><xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>

Response Format<Cmd=0x05>,<Len Byte = 0x18>,<Error Byte>,<xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>,<Type Byte>,<Battery Byte>,<SUP Byte>,<RunSec MS Byte>,<RunSec 2nd MS Byte>,<RunSec 3rd MS Byte>,<RunSec LS Byte>,<UnixSec MS Byte>,<UnixSec 2nd MS Byte>,<UnixSec 3rd MS Byte>,<UnixSec LS Byte>,<HW Rev MS Byte>,<HW Rev LS Byte>,<SW Rev MS Byte>,<SW Rev LS Byte>

Error Byte values:

- 0x00: Success
- 0x01: Bad argument. Bad length or xTAG provided may not have been scanned.
- 0x03: No connection. The xTAG must be connected first.
- 0x7F: General Error.

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. xTAG 6 byte address will be returned when error byte is Success or any error. Nothing will follow the xTAG address on error.
3. Type Byte values: 0x00-Accelerometer, 0x01-Accelerometer+Enviromental.
4. Battery Byte: 0x00-0xFF denoting range 2.2 Vdc to 3.0 Vdc. Always 0xFF for USB-connected xTAGs.
5. SUP Byte is a supplemental byte that is broadcast during BLE advertising. This can be set by the xtagd client for whatever purpose is required. Always 0x00 for USB-connected xTAGs.
6. RunSec Bytes make up an unsigned double-word int (UIN32) denoting the number of sec since xTAG firmware start. Can be used for remote troubleshooting as xTAG restarts only occur upon xTAG firmware failures.
7. UnixSec Bytes make up an unsigned double-word int (UIN32) denoting the number of sec since midnight on Jan 1, 1970 GMT (as per Unix timestamp standard). The xTAG does not store this value between boot cycles, so if an xTAG is restarted (for any reason) a client with a valid clock must set the xTAGs Unix time if this time is to be used (i.e. future timestamping). If unix time is not set, these read values won't be valid. IMPORTANTLY - the UnixSec is only updated every sec from an internal processor timer that will be minimally aperiodic. For this reason, expect that the unix time will have to be updated daily to remain within +/- 1-2 sec accuracy.
8. HW Bytes denote a raw unsigned word with implied decimals indicating xTAG hardware revision. E.g. 0x2711 = 10001 denoting rev 1.00.01.
9. SW Bytes denote a raw unsigned word with implied decimals indicating xTAG software or firmware revision. E.g. 0x2775 = 10101 denoting rev 1.01.01.

xTAG Metadata Write(0x06)

Write attributes of a connected xTAG.

Command Format <Cmd=0x06>,<Len Byte = 0x0D><xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>,<SUP Byte>,<UnixSec MS Byte>,<UnixSec 2nd MS Byte>,<UnixSec 3rd MS Byte>,<UnixSec LS Byte>

Response Format<Cmd=0x06>,<Len Byte = 0x09>,<Error Byte>,<xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>

Error Byte values:

- 0x00: Success
- 0x01: Bad argument. Bad length or xTAG provided may not have been scanned.
- 0x03: No connection. The xTAG must be connected first.
- 0x7F: General Error.

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. SUP Byte is a supplemental byte that is broadcast during BLE advertising. This can be set by the xtagd client for whatever purpose is required. Unused for USB-connected xTAGs (will be ignored).
3. See the note in xTAG Metadata Read (0x05) for UnixSec.
4. Note that this command replaces a legacy “set or write Unix time” command that did not support additional metadata.

xTAG Calibrate in 1 Dimension (0x11)

Position and calibrate the xTAG in one dimension. This ensures that the xTAG correctly measures 1 G in each dimension when positioned where that dimension should measure 1 standard gravity. Calibration will be stored within xTAG memory, even between battery replacement (if required). Execute for x, y and z sequentially.

Command Format <Cmd=0x11>,<Len Byte = 0x09><xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>,<Calibration Dimension Byte>

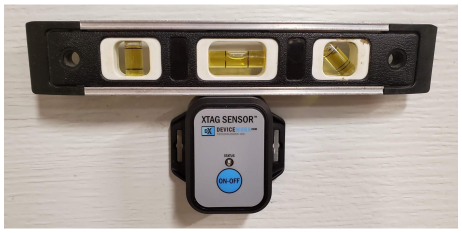
Response Format<Cmd=0x11>,<Len Byte = 0x09>,<Error Byte>,<xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>

Error Byte values:

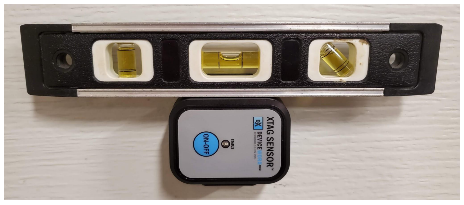
- 0x00: Success
- 0x01: Bad argument. Bad length or dimension..
- 0x02: Processing error: The xTAG accelerometer calibration failed on the xTAG.
- 0x03: No connection. The xTAG must be connected first.

Notes

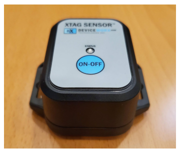
1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. Calibration Dimension Byte = 1 (calibrate x dim), 2 (calibrate y dim), 3 (calibrate z dim).
3. The xTAG must be positioned in the correct x, y or z orientation on a static surface before this command is executed. These positions are shown in pics below. The calibration process takes 1 sec and will support storing the x, y and z calibration data within xTAG permanent memory.



Calibrate X Dimension



Calibrate Y Dimension



Calibrate Z Dimension (on flat tabletop)

xTAG Accelerometer Acquisition Config (0x14)

Config the accelerometer on a connected xTAG.

Command Format <Cmd=0x14>,<Len Byte = 0x0B><xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>,<G Range Byte><G ODR Byte><G US & BWP Byte>

Response Format<Cmd=0x14>,<Len Byte = 0x09>,<Error Byte>,<xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>

Error Byte values:

- 0x00: Success
- 0x03: No Connection Error.
- 0x7F: General Error.

Notes (defaults in green)

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. The <G Range Byte> values are written directly into the ACC_RANGE register (0x41) in the BMI160:
    a. 0x03: 2 G
    b. 0x05: 4 G
    c. 0x08: 8 G
    d. 0x0C: 16 G
3. The <G ODR Byte> values are written directly into the least significant nibble of the ACC_CONF register (0x40) in the BMI160:
    a. 0x06: 25 samples/sec
    b. 0x07: 50 samples/sec
    c. 0x08: 100 samples/sec (Min xtagbled Phy speed of 1 Mbsp required)
    d. 0x09: 200 samples/sec (Min xtagbled Phy speed of 1 Mbsp required)
    e. 0x0A: 400 samples/sec (Min xtagbled Phy speed of 1 Mbsp required)
    f. 0x0B: 800 samples/sec (Min xtagbled Phy speed of 2 Mbsp required)
    g. 0x0C: 1600 samples/sec (Min xtagbled Phy speed of 2 Mbsp required)
4. The <G US & BWP Byte> values are written directly into the most significant nibble of the ACC_CONF register (0x40) in the BMI160. The only supported values are.
    a. 0x02: Undersampling disabled - normal filter mode. No oversampling. See table 12 in the BMI160 guide for 3db cutoff freq in this mode.
    b. 0x01: Undersampling disabled - OSR2 filter mode. Oversampling rate 2.
    c. 0x00: Undersampling disabled - OSR4 filter mode. Oversampling rate 2.
5. See Socket Interface for details on how to set a Phy value of 0 (125 kbps), 1 (1 Mbps) or 2 (2 Mbps). No phy value is required when starting xtagusbd for usb comms to xTAGs.

Accelerometer Acquisition Stream Start (0x16)

Start accelerometer FIFO reads.

Command Format <Cmd=0x16>,<Len Byte = 0x0A><xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>,<On Thresh Byte>,<Off Thresh Byte>

Response Format<Cmd=0x16>,<Len Byte = 0x0C>,<Error Byte>,<xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>,<G Range Byte><ODR Byte><Filter Byte>

Error Byte values:

- 0x00: Success
- 0x01: Bad argument. Bad length.
- 0x02: Processing Error. Non-zero <On/Off Thresh Byte> specified with ODR over 100 samples/s.
- 0x03: No Connection Error.
- 0x7F: General Error.

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. Starting (if successful) will automatically initialize the return of multiple stream data responses.
3. Issue a stream stop cmd to stop stream data responses.
4. G Range, ODR and Filter Bytes are used to feed back currently used config params in data acq. See the xTAG Accelerometer Acquisition Config (0x14) command for byte values.
5. If already streaming, this command will be ignored.
6. The On and Off Thresh Byte arguments support delayed start of the stream. Streaming will continue until stopped through this API. To immediately start streaming, simply set On Thresh Byte and Off Thresh Byte values to zero. A detailed description of delayed streaming follows in a separate paragraph.
7. Only the On Thresh Byte or Off Thresh Byte should be non-zero at any given time, but not both. If both On and Off Thresh Byte values are non-zero, the Off Thresh Byte will be treated as being zero.
8. On and Off Thresh Byte values range from 0-255 (UINT8). This value denotes a G value (within 50 percent of the selected G range). To select a top-range threshold value, use 255 (255/255 * 0.5 of G Range … eg 1G on a +/-2G range). To select a mid-range threshold value, use 127 (127/255 * 0.5 of G Range … eg 1G on a +/-4G range). To select an lower-range threshold value, use 63 (63/255* 0.5 of G Range … eg approx 1G on a +/-8G range). Note that these mid and lower range values are provided as examples only. Any value between 0 - 255 is valid.

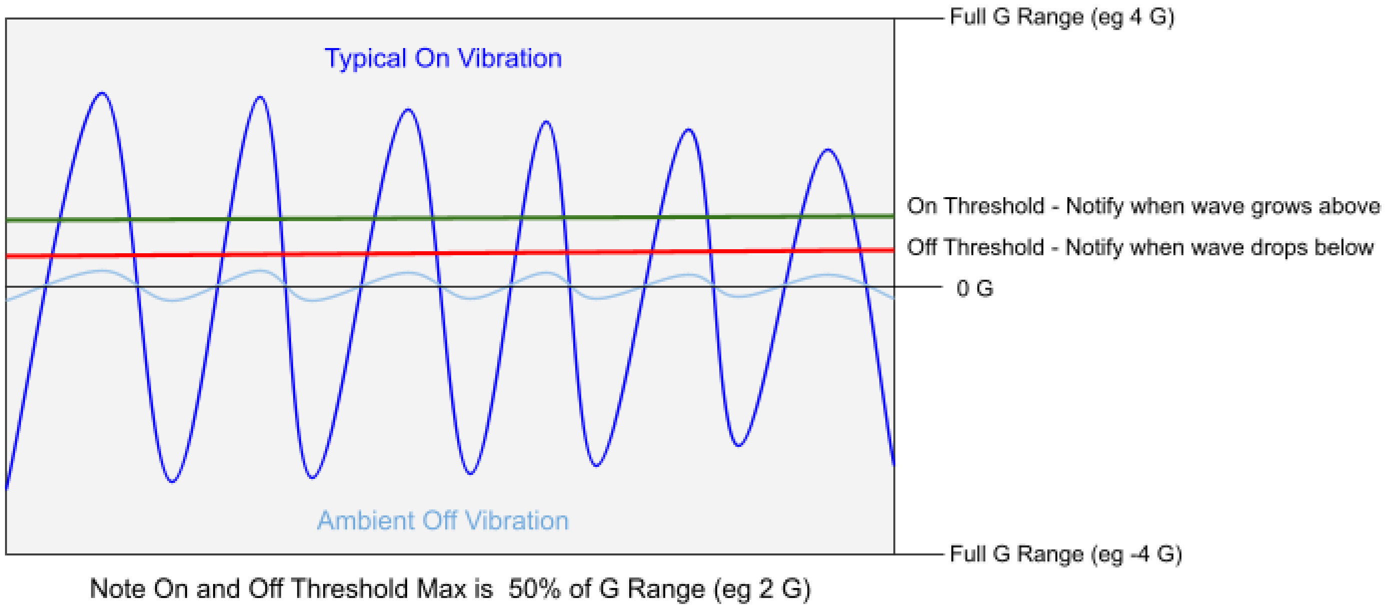
Delayed Stream Start Until Threshold Cross

To simplify xGATEWAY monitoring of machine On or Off status, and dramatically reduce required xTAG battery draw, the xTAG accelerometer's ability to watch for Significant Motion to see when a machine is On (operating or started) or to watch for No Motion to see when a machine is Off (idle or stopped) is leveraged. This is done by setting up the xTAG accelerometer to notify the xTAG microcontroller when it senses a machine is on or off. After getting notified (and not until), the xTAG microcontroller will stream data until a stream stop command is received. The xTAG accelerometer notifies the xTAG microcontroller that a machine is On whenever measured G values increase past a threshold. It notifies the xTAG that a machine is Off whenever measured G values decrease below a threshold. Only On or Off notification is supported at this time. Typical xGATEWAY client steps for using this feature are:

1. The API client Issues a start stream command (0x16) with On Thresh Byte non-zero to watch for a machine start.

2. The client then waits for the stream to start (0x17 responses returned) indicating that the machine has come On. xTAGs can watch for this state change using minimal battery power. Clients process stream response as required.
3. The client issues a stop stream command (0x18) to stop streaming (incoming 0x17 responses).
4. The API client issues a start stream command (0x16) with Off Thresh Byte non-zero to watch for the machine stop.
5. The client then waits for the stream to start indicating that the machine is Off. xTAGs can watch for this state change using minimal battery power. Read stream data as required.
6. Issue stop stream command (0x18) to stop streaming (incoming 0x17 responses).

On and Off Thresh bytes must be selected with machine vibration characteristics in mind. These characteristics include typical On vibration magnitude and typical ambient vibration magnitude (eg. vibration from machinery nearby). When watching for a machine to start or come On, the On threshold must be well above ambient vibration magnitude (to avoid false positives) and well below machine On vibration magnitude (to be reliable). When watching for a machine to stop or turn Off, the Off threshold must still be above ambient vibration (to avoid falsely reporting a machine Off event). The chart below shows candidate threshold values within an example vibration signal.



Contact Deviceworx support (support@deviceworx.com) as required for assistance in setting up delayed stream start for On or Off machine state monitoring.

NOTE - the goal of this feature is to dramatically reduce xTAG battery draw when monitoring a machine state that changes infrequently. For best efficiency, set up the xTAG sample rate at 25 or 50 samples per sec. 100 samples per sec is the max supported rate. Increasing the sample rate above this range will increase battery draw as this sample rate is used by the accelerometer to determine how frequently to sample acceleration values and notify the xTAG microcontroller when a threshold is crossed.

The stream start (due to accelerometer notification) is not immediate. Expect a 0.5 to 2 sec delay before the xTAG sends the first stream message after a threshold crossing has been determined.

Accelerometer Acquisition Stream Data (Responses Only) (0x17)

Whenever data is received from the BMI160 accelerometer (acc) on the sensor (as a result of a Stream Start cmd), this data will be automatically sent to the host as stream data responses. Formatting of these data responses is described here.

Stream Data Respone (no error): <Cmd=0x17><Resp Len>,<Error Byte=0x00>,<xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>,<X1 LSB><X1 MSB><Y1 LSB><Y1 MSB><Z1 LSB><Z1 MSB><X2 LSB><X2 MSB><Y2 LSB><Y2 MSB><Z2 LSB><Z2 MSB>... <Xn LSB><Xn MSB><Yn LSB><Yn MSB><Zn LSB><Zn MSB>

ATYPICAL DATA LSB BEFORE MSB - THIS IS TO MATCH DATA RETRIEVAL FROM THE ACC CHIP.

Ongoing Stream Data Response (plugged stream): <Cmd=0x17><Resp Len=0x04>,<Error Byte=0x05>,<Removed Samples Byte>

Error Byte values:

- 0x00: Cmd Success
- 0x05: Plugged Stream

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. The max number of samples that can be returned from a Sensor within each msg is 40 samples (40x6=240 bytes). Note that multiple msgs will be returned per second to satisfy the sample/sec selected. For example, if a sample rate of 200 samples / sec is selected 5 msgs, with 40 samples each, will be returned each second from that Sensor.
3. Whenever a plugged stream occurs (due to BLE transmission issue), the plugged stream message is sent to specify how many samples have been removed from the stream (0-255). When processing stream sample data, use the number of removed samples to omit data from processing as required. If a plugged stream persists, consider using a different communication index (see BLE Communication Optimization above).
4.

Accelerometer Acquisition Stream Stop (0x18)

Stop accelerometer FIFO stream reads.

Command Format <Cmd=0x18>,<Len Byte = 0x08><xTAG MS Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>

Response Format <Cmd=0x18>,<Len Byte = 0x09>,<Error Byte>,<xTAG 2nd MS Byte>,<xTAG 3rd MS Byte>,<xTAG 4th MS Byte>,<xTAG 5th MS Byte>,<xTAG LS Byte>

Notes

1. MS Byte = Most Significant Byte, LS Byte = Least Significant Byte
2. This command will simply stop the data stream responses (0x17). Expect these responses to stop and then receive the Stream Stop Cmd (0x18) response described here.
3. Sensor streaming can stop if its stream backs up too much due to buffering overflow when the wireless connection does not support the sample rate on the stream.

Error Byte values:

- 0x00: Success
- 0x03: No Connection Error.
- 0x04: Bad state. Not streaming when the stop command is sent.
- 0x7F: General Error.

Shutdown (0xFF)

Cleanly shutdown the xGw daemon and support disconnection of any xTAGs, etc before the socket connection is closed.

Command Format <Cmd=0xFF>,<Len Byte = 0x02>

Response Format<Cmd=0xFF>,<Len Byte = 0x03>,<Error Byte>

Error Byte values:

- 0x00: Success
- 0x7F: General Error. Check things like correct Len Byte.

Notes

1. Ensure that a 5 sec delay is observed after this command is executed before timing out as shutdown tasks won't be immediate.