

u-connectXpress

Extended data mode

Protocol specification

Abstract

This document describes the extended data mode protocol of the u-blox short range radio products. The extended data mode is an extension of the Wireless Multidrop approach, and it allows the user to individually control each active link. Thus, it is possible to transmit and/or receive data individually on each active channel.



Document information

| | | |
|-------------------------------|------------------------|-------------|
| Title | u-connectXpress | |
| Subtitle | Extended data mode | |
| Document type | Protocol specification | |
| Document number | UBX-14044126 | |
| Revision and date | R19 | 29-Nov-2021 |
| Disclosure Restriction | C1-Public | |

This document applies to the following products:

| Product series | Software version |
|-----------------------|-------------------------|
| ANNA-B1 | All |
| ANNA-B41 | All |
| NINA-B1 | 2.0.0 or later |
| NINA-B2 | All |
| NINA-B31 | All |
| NINA-B41 | All |
| NINA-W13 | All |
| NINA-W15 | All |
| ODIN-W2 | All |

u-blox or third parties may hold intellectual property rights in the products, names, logos and designs included in this document. Copying, reproduction, modification or disclosure to third parties of this document or any part thereof is only permitted with the express written permission of u-blox.

The information contained herein is provided "as is" and u-blox assumes no liability for its use. No warranty, either express or implied, is given, including but not limited to, with respect to the accuracy, correctness, reliability and fitness for a particular purpose of the information. This document may be revised by u-blox at any time without notice. For the most recent documents, visit www.u-blox.com.

Copyright © u-blox AG.

Contents

| | |
|--|-----------|
| Document information | 2 |
| Contents | 3 |
| 1 Introduction | 4 |
| 2 Extended data mode protocol | 5 |
| 2.1 Packet..... | 5 |
| 2.2 Payload..... | 5 |
| 2.2.1 Connect event (0x0011)..... | 6 |
| 2.2.2 Disconnect event (0x0021)..... | 7 |
| 2.2.3 Data event (0x0031) | 7 |
| 2.2.4 Data command (0x0036) | 8 |
| 2.2.5 AT request (0x0044)..... | 8 |
| 2.2.6 AT response (0x0045) | 8 |
| 2.2.7 AT event (0x0041) | 9 |
| 2.2.8 Resend connect events command (0x0056) | 9 |
| 2.2.9 Start event (0x0071)..... | 9 |
| A Limitations..... | 10 |
| A.1 Connection setup | 10 |
| Related documents | 11 |
| Revision history | 12 |
| Contact..... | 13 |

1 Introduction

The module can be in three different modes:

- Command Mode (for configuration)
- Data Mode
- Extended Data Mode

The data mode supports multipoint connections using the Wireless Multidrop scheme. This means that anything transmitted on the serial line to one node, is transmitted, over air, to all the connected remote devices. All data received over air, from the remote devices is transmitted on the serial line without any information about the remote device that transmitted the data.

The extended data mode is a protocol to enable control of each individual connection. Thus, it is possible to transmit data to one specific remote device and to know from what remote device the data is received.

By default, the module will enter command mode. It has to be configured to enter extended data mode or data mode.

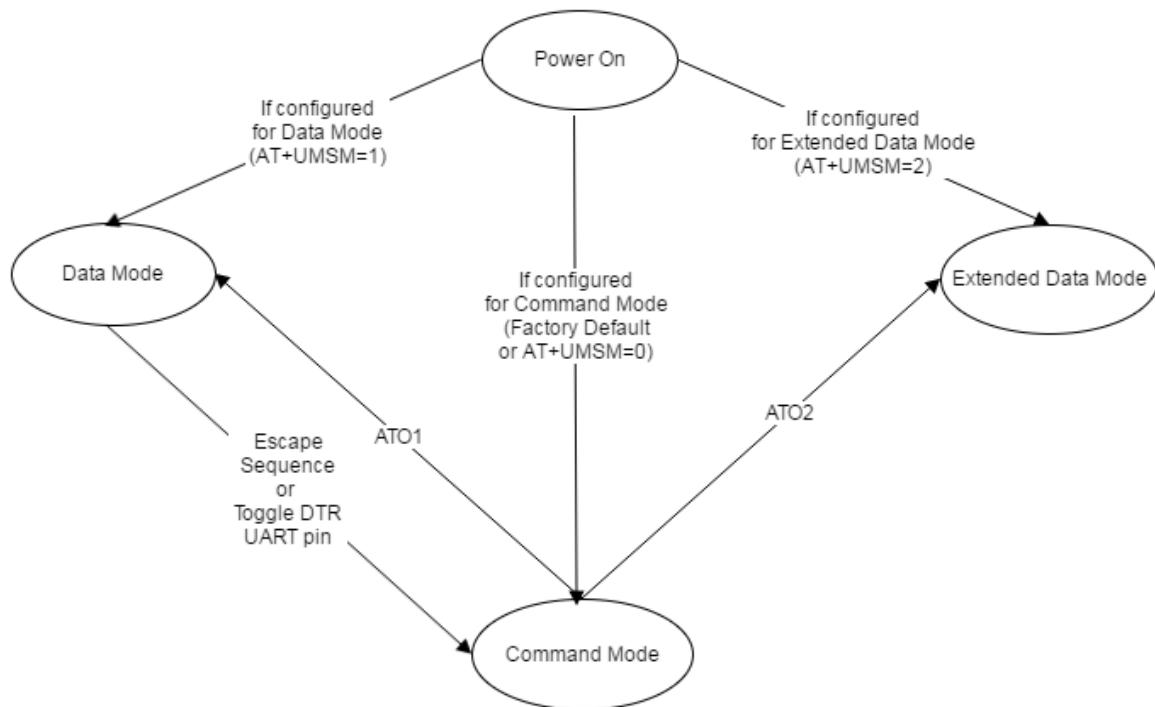


Figure 1: State diagram of the different modes for the module

Since it is possible to execute AT commands as part of the extended data mode protocol, it is not necessary to enter command mode when in the extended data mode.

2 Extended data mode protocol

An extended data mode packet consists of a packet header, payload, and tail.

The payload consists of an identifier followed by an event, command, request, response, indication or confirmation. Byte order for all packets are network order (most significant bytes first).

2.1 Packet

A packet starts with a start byte (0xAA) and ends with a stop byte (0x55) for easy parsing and packet re-synchronization.

The length of the payload is defined by 12 bits. Four bits are reserved for future use. Hence, the total packet length is the payload length plus four (start and stop bytes plus reserved and length bits).

| | | | | |
|--------------------------|----------------------|-----------------------------|---------------------------|-------------------------|
| Start (1 byte = 0xAA) | Reserved (4 bits) | Payload Length (12 bits) | Payload (Length bytes) | Stop (1 byte = 0x55) |
|--------------------------|----------------------|-----------------------------|---------------------------|-------------------------|

2.2 Payload

The payload starts with two header bytes to identify exactly the kind of data that is included in the payload.

| | | |
|-------------------------|------------------|--|
| Identifier (12 bits) | Type (4 bits) | Event, Indication, Response, Request, Confirmation, or Command (Payload Length – 2 bytes) |
|-------------------------|------------------|--|

The Type identifies if the data is an event, indication, response, request, confirmation, or command.

| Type | Name | Description |
|------|--------------|--|
| 0x1 | Event | Transmitted by the module as a notification. It does not require a response from the host. |
| 0x2 | Indication | Transmitted by the module as a notification. The module expects a Response back from the host. |
| 0x3 | Response | If an Indication is received from the module, the host must respond with a Response. |
| 0x4 | Request | A request is transmitted to the module to execute some functionality. The module must respond with a Confirmation. |
| 0x5 | Confirmation | A response to an executed Request. |
| 0x6 | Command | A command is transmitted to the module to execute some functionality. No response is expected. |

The Identifier identifies what event, indication, response, request, confirmation, or command is transmitted or received. Currently, the following packets are defined:

| Identifier + Type | Name | Description |
|-------------------|------------------|--|
| 0x0011 | Connect Event | Sent by the module to inform the host about a new connection. |
| 0x0021 | Disconnect Event | Sent by the module to inform the host about the loss of connection. |
| 0x0031 | Data Event | Sent by the module when data is received over air. |
| 0x0036 | Data Command | Sent to the module to send data over air. No acknowledge is transmitted by the module. |
| 0x0044 | AT Request | Special packet to execute an AT command. One or many AT Confirmation packets are transmitted back by the module. |
| 0x0045 | AT Confirmation | The module sends one or many confirmations as a response to an AT Request. The number of confirmation packets depends on what AT command that is being executed. |
| 0x0041 | AT Event | There are a number of AT events that can be sent by the module. See the u-connectXpress AT commands manual [1] for details. |

| Identifier + Type | Name | Description |
|-------------------|-------------------------------|--|
| 0x0056 | Resend Connect Events Command | Special command to make the module re-transmit Connect Events for connections still active. This can be useful, for example, when the host has reset or just been started. |
| 0x0061 | iPhone Event | Special iPhone events, for example, session status and power state. |
| 0x0071 | Start Event | Sent when the module recovers from reset or at power on. This packet may need special module configuration to be transmitted. |

2.2.1 Connect event (0x0011)

There are a number of different Connect Events, each with different information depending on factors such as the wireless technology that is being used.

| | | | |
|---------------------------------|------------------------|--------------------------|-----------------------------------|
| Id + Type (2 bytes = 0x0011) | Channel Id (1 byte) | Connect Type (1 byte) | Type Specific Payload (n byte) |
|---------------------------------|------------------------|--------------------------|-----------------------------------|

The Connect Event always starts with the two bytes Identifier+Type header (0x0011) followed by a Channel Identifier (1 byte) and a Connect Type (1 byte). The Connect Type identifies what type of information follows subsequently. The channel identifier is unique for each active link and it is used in other packets to identify the link that is considered.

| Connect Type | Value | Comment |
|--------------|-------|---------|
| Bluetooth® | 0x01 | |
| Ipv4 | 0x02 | |
| Ipv6 | 0x03 | |

2.2.1.1 Connect event Bluetooth (0x01)

When a Bluetooth connection is setup, the module sends a connect event to the host.

| | | | | | |
|---------------------------------|------------------------|---------------------------------|---------------------|-------------------------|-------------------------|
| Id + Type (2 bytes = 0x0011) | Channel Id (1 byte) | Connect Type (1 byte = 0x01) | Profile (1 byte) | BD Address (6 bytes) | Frame Size (2 bytes) |
|---------------------------------|------------------------|---------------------------------|---------------------|-------------------------|-------------------------|

The channel is used as the identifier of the connection, and it is unique for each active connection. The profile is used to identify what Bluetooth profile is associated with the connection.

- SPP = 0
- DUN = 1
- Serial Port Service Bluetooth Low Energy = 14

The Bluetooth address is the address of the remote device that is now connected, and the frame size is the maximum data size allowed to be transmitted in a data command or data event packet. The actual extended data mode packet (for example, Data Cmd) will then have a total length greater than the frame size as it will include for example, extended data mode header and tail.

For example:

Channel 3, Bluetooth Connect Event 1, Profile 0, Bluetooth Address 0x112233445566, Frame Size 358 (0x0166)

| | | | | | | | | |
|------|--------|--------|------|------|------|----------------|--------|------|
| 0xAA | 0x000D | 0x0011 | 0x03 | 0x01 | 0x00 | 0x112233445566 | 0x0166 | 0x55 |
|------|--------|--------|------|------|------|----------------|--------|------|

2.2.1.2 Connect event Ipv4 (0x02)

| Id + Type (2 bytes = 0x0011) | Channel Id (1 byte) | Connect Type (1 byte = 0x02) | Protocol (1 byte) | Remote Ipv4 Address (4 bytes) | Remote Port Number (2 bytes) | Local Ipv4 Address (4 bytes) | Local Port Number (2 bytes) |
|------------------------------------|------------------------|---------------------------------|----------------------|-------------------------------------|------------------------------------|------------------------------------|-----------------------------------|
|------------------------------------|------------------------|---------------------------------|----------------------|-------------------------------------|------------------------------------|------------------------------------|-----------------------------------|

The protocol is used to identify the IP protocol that is associated with the connection.

- 0: TCP
- 1: UDP

For example:

Channel 5, Connect Event TCP/Ipv4 2, Remote IP Address 192.168.0.2, Remote Port 5000, Local IP Address 192.168.0.1, Local Port 4000

| | | | | | | | | | | |
|------|--------|--------|------|------|------|------------|--------|------------|--------|------|
| 0xAA | 0x0011 | 0x0011 | 0x05 | 0x02 | 0x00 | 0xC0A80002 | 0x1388 | 0xC0A80001 | 0x0FA0 | 0x55 |
|------|--------|--------|------|------|------|------------|--------|------------|--------|------|

2.2.1.3 Connect event Ipv6 (0x03)

| Id + Type (2 bytes = 0x0011) | Channel Id (1 byte) | Connect Type (1 byte = 0x03) | Protocol (1 byte) | Remote Ipv6 Address (16 bytes) | Remote Port Number (2 bytes) | Local Ipv6 Address (16 bytes) | Local Port Number (2 bytes) |
|------------------------------------|------------------------|---------------------------------|----------------------|--------------------------------------|------------------------------------|-------------------------------------|-----------------------------------|
|------------------------------------|------------------------|---------------------------------|----------------------|--------------------------------------|------------------------------------|-------------------------------------|-----------------------------------|

The protocol is used to identify the IP protocol that is associated with the connection.

- 0: TCP
- 1: UDP

For example:

Channel 5, Connect Event TCP/Ipv6 2, Remote IP Address FE80::2, Remote Port 5000, Local IP Address FE80::1, Local Port 4000

| | | | | | | | | | | |
|------|--------|--------|------|------|------|--|--------|--|--------|------|
| 0xAA | 0x0029 | 0x0011 | 0x05 | 0x03 | 0x00 | 0xFE80000000000000 0000000000000002 | 0x1388 | 0xFE80000000 000000000000 0000000001 | 0x0FA0 | 0x55 |
|------|--------|--------|------|------|------|--|--------|--|--------|------|

2.2.2 Disconnect event (0x0021)

When a previously set up connection is terminated, the module sends the disconnect event.

| | |
|---------------------------------|------------------------|
| Id + Type (2 bytes = 0x0021) | Channel Id (1 byte) |
|---------------------------------|------------------------|

The channel was allocated in the connect event. For example:

Channel 3

| | | | | |
|------|--------|--------|------|------|
| 0xAA | 0x0003 | 0x0021 | 0x03 | 0x55 |
|------|--------|--------|------|------|

2.2.3 Data event (0x0031)

When the module receives data on a peer, it transmits a data event to the host with the received data.

| | | |
|---------------------------------|------------------------|------------------------------------|
| Id + Type (2 bytes = 0x0031) | Channel Id (1 byte) | Data (Payload Length – 3 bytes) |
|---------------------------------|------------------------|------------------------------------|

The channel identifies the remote device from which the data was received. For Bluetooth the number of data bytes cannot be more than the value specified by the frame size in the connect event. For IP the number of data bytes is not greater than the maximum EDM payload size (4095 bytes).

For example:

Channel 3, Data (2 bytes) 0x1234

| | | | | | |
|------|--------|--------|------|--------|------|
| 0xAA | 0x0005 | 0x0031 | 0x03 | 0x1234 | 0x55 |
|------|--------|--------|------|--------|------|

2.2.4 Data command (0x0036)

To transmit data, the host must send a data command to the module.

| | | |
|---------------------------------|------------------------|------------------------------------|
| Id + Type (2 bytes = 0x0036) | Channel Id (1 byte) | Data (Payload Length – 3 bytes) |
|---------------------------------|------------------------|------------------------------------|

The channel identifies the remote device to which the data shall be transmitted. For Bluetooth the number of data bytes must not exceed the frame size in the connect event. For IP when sending data the maximum data size is 635 bytes. No acknowledgement is sent by the module.

When a disconnect event is issued, all internal Bluetooth queues are cleared and the data that is not yet transmitted can be lost. It is recommended to have a short delay between the data transmission and disconnect command.

For example:

Channel 3, Data (2 bytes) 0x1234

| | | | | | |
|------|--------|--------|------|--------|------|
| 0xAA | 0x0005 | 0x0036 | 0x03 | 0x1234 | 0x55 |
|------|--------|--------|------|--------|------|

2.2.5 AT request (0x0044)

To make the module execute an AT command, the AT request command is sent to the module.

| | |
|---------------------------------|--|
| Id + Type (2 bytes = 0x0044) | AT Command (Payload Length – 2 bytes) |
|---------------------------------|--|

The AT Command is a string defined in the AT commands manual for the specific module [1]. The command must always be terminated with “\r”. The module will respond with one or more AT Response packets.

The device can execute only one AT command at a time. If the next command is received when the previous one is executed, the module throws it away and returns ERROR.

For example:

Command “AT\r” (0x41540D)

| | | | | |
|------|--------|--------|----------|------|
| 0xAA | 0x0005 | 0x0044 | 0x41540D | 0x55 |
|------|--------|--------|----------|------|

2.2.5.1 AT request serial settings

The AT+UMRS command to change serial settings does not work exactly the same as in command mode. When executed in the extended data mode, it is not possible to change the settings directly using the <change_after_confirm> parameter. Instead, the <change_after_confirm> parameter must be set to 0 and the serial settings will take effect when the module is reset.

2.2.6 AT response (0x0045)

The module will transmit one or more AT Response packets as a response to an AT Request.

| | |
|---------------------------------|---|
| Id + Type (2 bytes = 0x0045) | AT Command Response (Payload Length – 2 bytes) |
|---------------------------------|---|

The number of response packets depends on the executed AT command.

For example:

Response “\r\nOK\r\n” (0xD0A4F4B0D0A)

| | | | | |
|------|--------|--------|---------------|------|
| 0xAA | 0x0008 | 0x0045 | 0xD0A4F4B0D0A | 0x55 |
|------|--------|--------|---------------|------|

2.2.7 AT event (0x0041)

There are some occasions when the module transmits AT Event packets.

| | |
|---------------------------------|--|
| Id + Type (2 bytes = 0x0041) | AT Event (Payload Length – 2 bytes) |
|---------------------------------|--|

See the u-connect AT commands manual [1] for the event details.

For example:

Disconnect Event: “+UUDPD:3,1\r\n” (0x2B55554450443A332C310D0A)

| | | | | |
|------|--------|--------|----------------------------|------|
| 0xAA | 0x000C | 0x0041 | 0x2B55554450443A332C310D0A | 0x55 |
|------|--------|--------|----------------------------|------|

2.2.8 Resend connect events command (0x0056)

If the host resets or starts, there may be existing connections that are active. To get information about existing connections, there is a Resend Connect Events command, which can be sent to the module. The module will then resend the Connect Events for any existing connections. If there are no existing connections, no events will be sent.

Id + Type (2 bytes = 0x0056)

For example:

| | | | |
|------|--------|--------|------|
| 0xAA | 0x0002 | 0x0056 | 0x55 |
|------|--------|--------|------|

2.2.9 Start event (0x0071)

Depending on the configuration, this event may be sent to inform the host that the module is started.

Id + Type (2 bytes = 0x0071)

For example:

Channel 3

| | | | |
|------|--------|--------|------|
| 0xAA | 0x0002 | 0x0071 | 0x55 |
|------|--------|--------|------|

A Limitations

A.1 Connection setup

It is not supported to set up a connection in AT mode and then enter extended data mode to send data. This may result in lost data.

Related documents

- [1] u-connectXpress AT commands manual, [UBX-14044127](#)
- [2] ODIN-W2 system integration manual, [UBX-14040040](#)
- [3] NINA-B1 system integration manual, [UBX-15026175](#)
- [4] NINA-B3 system integration manual, [UBX-17056748](#)
- [5] NINA-W1 system integration manual, [UBX-17005730](#)
- [6] ANNA-B112 system integration manual, [UBX-18009821](#)
- [7] NINA-B2 system integration manual, [UBX-18011096](#)
- [8] NINA-B4 system integration manual, [UBX-19052230](#)
- [9] ANNA-B4 system integration manual, [UBX-21000517](#)

 For product change notifications and regular updates of u-blox documentation, register on our website, www.u-blox.com.

Revision history

| Revision | Date | Name | Comments |
|----------|-------------|------------------|--|
| R01 | 17-Nov-2014 | pber | Initial release. |
| R02 | 06-Mar-2015 | pber | Adapted to new name, ODIN-W2. |
| R03 | 23-Sep-2015 | smos | New title to reflect general applicability of the protocol specification to a number of u-blox short range modules. |
| R04 | 4-May-2016 | mlju | Added support for ODIN-W260-01B, ODIN-W262-01B, and firmware version 2.0.0. Modified the document status to Early Production Information. |
| R05 | 29-Sep-2016 | mhan, kgom | Added support for ODIN-W2 firmware version 3.0.0. Fixed packet length error in the examples for Connect Event Ipv4 and Ipv6 (sections 2.2.1.2 and 2.2.1.3). Updated description of AT Request (section 2.2.5). Removed PAN example section as it is not supported. |
| R06 | 3-Jan-2017 | mlju, ecar, kgom | Added support for NINA-B1 firmware version 2.0.0 and ODIN-W2 firmware versions – 2.0.2 and 3.0.1. Replaced Document status with Disclosure restriction on page 2. |
| R07 | 30-Mar-2017 | ecar, kgom | On page 2, replaced type numbers and firmware version with All for ODIN-W2 to denote support for all type numbers and firmware versions. |
| R08 | 3-Jul-2017 | kgom | Replaced firmware with software. |
| R09 | 27-Nov-2017 | objo, kgom | Included support for NINA-W1. |
| R10 | 21-Dec-2017 | hwin, kgom | Included support for NINA-B3. |
| R11 | 27-Mar-2018 | kgom | Included support for ANNA-B112. |
| R12 | 17-Apr-2018 | kgom | Included support for NINA-B2. |
| R13 | 12-Sep-2018 | kgom | Minor formatting changes. |
| R14 | 13-Feb-2019 | kgom | Included support for NINA-B316. |
| R15 | 07-May-2019 | hwin | Included additional information in the Data Command (section 2.2.4). |
| R16 | 28-Jun-2019 | kgom | Included support for NINA-W15. |
| R17 | 09-Nov-2020 | mhan, mape | Updated info about IP data command and event (section 2.2.3 and 2.2.4). Added Appendix A limitations. |
| R18 | 08-Mar-2021 | flun, mhan | Included support for NINA-W156, NINA-B41, ODIN-W263. Renamed document and simplified applicable products table on page 2. |
| R19 | 29-Nov-2021 | hisa | Included support for ANNA-B41. |

Contact

For complete contact information, visit us at www.u-blox.com.

u-blox Offices

North, Central and South America

u-blox America, Inc.

Phone: +1 703 483 3180

E-mail: info_us@u-blox.com

Regional Office West Coast:

Phone: +1 408 573 3640

E-mail: info_us@u-blox.com

Technical Support:

Phone: +1 703 483 3185

E-mail: support_us@u-blox.com

Headquarters

Europe, Middle East, Africa

u-blox AG

Phone: +41 44 722 74 44

E-mail: info@u-blox.com

Support: support@u-blox.com

Asia, Australia, Pacific

u-blox Singapore Pte. Ltd.

Phone: +65 6734 3811

E-mail: info_ap@u-blox.com

Support: support_ap@u-blox.com

Regional Office Australia:

Phone: +61 3 9566 7255

E-mail: info_au@u-blox.com

Support: support_ap@u-blox.com

Regional Office China (Beijing):

Phone: +86 10 68 133 545

E-mail: info_cn@u-blox.com

Support: support_cn@u-blox.com

Regional Office China (Chongqing):

Phone: +86 23 6815 1588

E-mail: info_cn@u-blox.com

Support: support_cn@u-blox.com

Regional Office China (Shanghai):

Phone: +86 21 6090 4832

E-mail: info_cn@u-blox.com

Support: support_cn@u-blox.com

Regional Office China (Shenzhen):

Phone: +86 755 8627 1083

E-mail: info_cn@u-blox.com

Support: support_cn@u-blox.com

Regional Office India:

Phone: +91 80 405 092 00

E-mail: info_in@u-blox.com

Support: support_in@u-blox.com

Regional Office Japan (Osaka):

Phone: +81 6 6941 3660

E-mail: info_jp@u-blox.com

Support: support_jp@u-blox.com

Regional Office Japan (Tokyo):

Phone: +81 3 5775 3850

E-mail: info_jp@u-blox.com

Support: support_jp@u-blox.com

Regional Office Korea:

Phone: +82 2 542 0861

E-mail: info_kr@u-blox.com

Support: support_kr@u-blox.com

Regional Office Taiwan:

Phone: +886 2 2657 1090

E-mail: info_tw@u-blox.com

Support: support_tw@u-blox.com