International Headquarters
Myllyojankatu 2 A
FIN-24100 SALO
FINLAND
tel +358 (0)2 727 7700
fax +358 (0)2 727 7720
www.nordicid.com

NORDICID

Nordic ID

Command Protocol for Nordic ID UHF Reader (NUR)

Revision 14

API DLL version 1.7.9.1 (and above)

FW version 3.6-A and above for NUR module

FW version 4.9-E and above for L2-family

NORDICID

Contents

1 Vocabulary

Term	Is
NUR05WL2	Current 500mW Nordic ID UHF RFID reader module.
NUR10W	Current 1W (1000mW) Nordic ID UHF RFID reader module.
”L2 module family”, ”L2 family”	Currently refers to NUR05L2 and NUR10W.
BYTE	Unsigned 8-bit integer.
WORD	Unsigned 16-bit integer; length is 2 bytes.
DWORD	Unsigned 32-bit integer; length is 4 bytes.
QWORD	Unsigned 64-bit integer; length is 8 bytes.
EAS	Electronic Article Surveillance.
short	Signed 16-bit integer; length is 2 bytes.
RSSI	Return Signal Strength Indicator
Link frequency, LF	Means the frequency offset from the channel’s center frequency the tag uses for its response
Singulation, to singulate	Means of addressing a certain tag of interest. Usually means selecting a tag for reading, writing, locking, killing or for proprietary/custom command execution that is intended for the one selected (singulated) tag only.
Stream	An autonomous operation performed by the module that aims to remove the execution overhead from the host i.e. the stream is started and then the stream only notifies the host that it has some new results related to the started stream.
Inventory	An explicitly commanded operation that aims to find as many tags as possible in the RF field with given parameters (Q, session, target)
Inventory stream	An inventory that is started and then autonomously reports its results to the host.
Inventory read	An inventory, any inventory, operation that is instructed to also read from the tag during an inventory. The inventory read is configured before an inventory call and thus does not do anything itself; the previous inventory + read configuration states what the inventory should do when called the next time (or stream is started).

2 Notation

In this document the protocol and commands are described in terms of fields. A ‘field” can be byte, word, double word, quad word or byte array; see “Vocabulary”. All the values occupying more than one byte are in little-endian format. For example the hexadecimal value AABBCCDD in little-endian format is stored into memory like DD CC BB AA. So when reading the command specification it should be kept in mind that there is no length field specified for the fixed size parameters such as WORD or DWORD.

Generic example of a data structure containing one WORD, one short and one DWORD parameter and how the parameters are placed into memory:

Offset	Size in bytes	Type and value	Stored like…
0	2	Parameter 1: WORD, 0x1234	0x34 0x12
2	2	Parameter 2:Short, -17 = 0xFFEF	0xEF 0xFF
4	4	Parameter 3:DWORD, 0x01020304	0x04 0x03 0x02 0x01

The byte array would the look like (HEX notation):

Byte	0	1	2	3	4	5	6	7
Value	34	12	EF	FF	04	03	02	01
Parameter	1: WORD.		2: short		3: DWORD			

3 Protocol description

The protocol consists of a command (and response) header and payload. Both of them have separate checksums; the header has a simple XOR checksum and the payload is appended with a CRC-16 checksum.

Communication packet		
Header	Command + parameters or response / notification + parameters	CRC-16
6 bytes	Command / response / notification payload	

In the normal operating mode the protocol is simple command-response sequence. However the module can also send unsolicited notifications such as I/O pin change or tag trace event.

3.1 Command / response header

The header's length is 6 bytes. Fields in the header are:

Offset	Field	Is
0	Start byte.	Value is 0xA5.
1	Length word.	Payload length including the CRC-16 field.
3	Flag word.	Command / response flags.
5	Checksum	Simple checksum over the header; start with value 0xFF and covers the start byte, length field and flag field.

3.2 Response header flags

The following flags are defined:

Bit	Word value	Is
0	0x0001	Means unsolicited message i.e. notification from the module.
1	0x0002	Means that this notification is from inventory stream and that in addition to the tag data there is also inventory + read data present in the tag entries. This is NUR05WL2 module's feature.

3.3 Command payload

The payload consists of a one byte command and its parameters. In some cases where the command is a query the parameter(s) may be omitted. Example of a command payload that takes a DWORD parameter:

Offset	Field	Note
0	Command byte.	
1	DWORD parameter.	In little-endian format.
5…6	CRC-16	

3.4 Response / notification payload

The response payload starts with the echoed command followed by a status byte indicating either success or fail:

Offset	Field	Note
0	Command byte.	Echo of the last command or notification code.
1	Status byte.	0 means success, otherwise it is an error code. Error code is only present in a response to a command.
2…N-1	Response	N bytes.
N…N+1	CRC-16.	Response CRC.

3.5 CRC-16

The CRC-16 is done with start value of 0xFFFF and the polynomial is 0x1021. CRC is calculated only for the command payload. Note that in this documentation only the command and response fields are described; the CRC-16 is always appended to the command, response and notification. See "CRC-16 calculation".

4 Notifications (unsolicited messages)

4.1 List of all notifications

Value	Meaning
0x80	Boot notification
0x81	I/O change
0x82	Inventory, contents here.
0x83	RFU: ignore
0x84	Trace tag
0x85	Triggered read
0x86	Frequency hop event
0x87	NID internal: ignore
0x88	Extended inventory
0x89	NXP alarm (see also proprietary commands)
0x8A	EPC enumeration (EPC enumeration command)
0x8B	External in *(Ethernet Sampo: documented separately)*.
0x8C	General / reserved: ignore
0x8D	Autotune (see RF control: Automatic tuning)
0x8E	*WLAN search: Ethernet Sampo / Area Reader (AR): documented separately.*

4.2 Boot notification

This notification is always sent when the module starts up. The content of the notification is:

Offset	Field	Note
0	Notification code.	0x80 (boot)
1…6	ASCII ‘LOADER’.	When in bootloader mode.
1…3	ASCII “APP”	When in application mode.

4.3 I/O pin change notification

The module can also be configured to trigger an event when a pin that is configured as an input changes its state. This is useful when for example triggering an inventory. Notification is then:

Offset	Field	Note
0	Notification code.	0x81 (I/O state change)
1	Source	Source is: 0…0x7F = I/O pin number 0x80: tap sensor (only w/ USB table reader) 0x81: light sensor (only w/ USB table reader) Others are reserved.
2	Direction	0 = from high to low (with light sensor means 'light out') 1 = from low to high (with light sensor means 'light on') With tap sensor this value is meaningless; the source byte defines that "unit was tapped".

4.4 Inventory notification

Offset	Field	Note
0	Notification code.	0x82 (inventory stream notification)

This notification is sent during inventory stream. See "Inventory stream response".

4.5 Trace tag notification

The tag trace operation is a continuous operation where the module tries to select a single tag based on given criteria. This feature is useful for, like the name says, tracing a specific tag from a larger population of tags. When the scan is successful the module sends this notification:

Offset	Field	Is
0	Notification code.	0x84 (tag traced)
1	RSSI (short).	Negative RSSI estimate.
2	Scaled RSSI	BYTE: 0…100%
3	Antenna ID	Antenna through which the tag was found, 0…3.
4…n-1	EPC	The responding tag's EPC. **NOTE:** this field is present only if the trace tag was configured to send also the EPC of the tag.

4.6 Triggered read notification

If an I/O pin, light sensor or tap sensor has been configured to scan a single tap when GPIO changes its state or there is a sensor event then the notification is:

Offset	Field	Note
0	Notification code.	0x85 (triggered read)
1	Source	Source is: 0…0x7F = I/O pin number 0x80: tap sensor (only w/ USB table reader) 0x81: light sensor (only w/ USB table reader) Others are reserved.
2	RSSI	Signed 16-bit RSSI value for the scanned tag.
3	Scaled RSSI.	BYTE: 0…100%.
4…N-1	EPC	The scanned tag's EPC.

4.7 Frequency hop notification

If the module is configured to trigger an event upon a frequency hop then the notification is:

Offset	Field	Note
0	Notification code.	0x86 (frequency hop)
1	Number of the current region.	Value is currently 0…6.
2	Index of the current frequency in the hop table.	Value depends on the selected region. For example for the EU hop table this value is 0…3.

5 Basic commands

5.1 Ping

Ping is simply used to check the communication with the module. Ping command value is 1.

Ping command

Offset	Field	Note
0	Ping command.	Value is 1.

Ping response

Offset	Field	Note
0	Command echo.	Value is 1.
1	Status	Always 0.
2	ASCII ‘O’.	Module responds with “OK”.
3	ASCII ‘K.	

5.2 Reset

This command causes the RF part re-setup and currently saved setup restoring.

Reset command

Offset	Field	Note
0	Reset.	Value is 3.

Reset response

Offset	Field	Note
0	Command echo.	Value is 3.
1	Status	Always 0.

5.3 Get mode

As there are two modes that the module can be in (bootloader and application) this command returns the current state. Mode can also be determined by the boot notification.

Get mode command

Offset	Field	Note
0	Get mode.	Value is 4.

Get mode response

Offset	Field	Note
0	Command echo.	Value is 4.
1	Status	Always 0.
2	Mode byte.	'A' = application 'B' = bootloader.

5.4 Clear ID buffer

This command clears all tag data currently stored into the module's memory. Note that executing an inventory does not automatically clear the ID buffer, it has to be done with either this or get ID buffer command if it is needed to make sure that there is no tag information present.

Clear ID buffer command

Offset	Field	Note
0	Clear ID buffer.	Value is 5.

Clear ID buffer Response

Offset	Field	Note
0	Command echo.	Value is 5.
1	Status	Always 0.

5.5 Get ID buffer

This command is used to read the current contents of the ID buffer. The command can also be instructed to empty the buffer after it has been sent.

Get ID buffer command

Offset	Field	Note
0	Get ID buffer.	Value is 6.
1	Clear flag	Optional. Set this value to 1 in order to clear the ID buffer after it has been sent.

Get ID buffer response

Offset	Field	Note
0	Command echo.	Value is 6.
1	Status	0 on success (there are tags in the storage. 0x20: no tags. In this case there are no more data in the response.
…	Tag data.	See below

Per tag basis the data is presented like this:

Offset	Field	Note
0	Block length.	Number of bytes to follow.
1	Antenna ID.	Currently not used.
2…(2+n-1)	EPC data	Tag's EPC. Length is block length – 1.

5.6 Get ID buffer with metadata

This command is used to read the current contents of the ID buffer pre-pended with some additional tag related data: timestamp, RSSI, frequency, PC and channel number. The command can also be instructed to empty the buffer after it has been sent.

Get ID buffer with metadata command

Offset	Field	Note
0	Get ID buffer and metadata.	Value is 7.
1	Clear flag	Optional. Set this value to 1 in order to clear the ID buffer after it has been sent.

Get ID buffer with metadata response

Offset	Field	Note
0	Command echo.	Value is 7.
1	Status	0 on success (there are tags in the storage). 0x20: no tags. In this case there is no more data in the response.
…	Tag metadata + EPC.	See below

Per tag basis the data is presented as follows:

Byte(s)	Field	Note
0	Block length.	Number of bytes to follow.
1	RSSI, signed 8-bit.	Negative number presenting the best RSSI for this tag.
2	Scaled RSSI.	Scaled RSSI value in range 0…100.
3…4	Timestamp.	Millisecond offset from the beginning of the inventory; last time the tag was seen (best RSSI).
5…8	Frequency in kHz.	DWORD value presenting the frequency that the tag was inventoried in (last time).
9…10	PC (+NSI/AFI) WORD.	Contents from the tag's EPC/UII memory's word address 1.
11	Channel (BYTE).	Channel number for the last time that the tag was inventoried.
12	Antenna ID.	Antenna from which the tag was inventoried from.
13…	EPC bytes.	The tag's EPC: length is block length – 12.

5.7 Metadata per tag basis when inventory + read data is present

When the module last inventory operation was inventory + read then the per tag basis metadata is as follows:

Byte(s)	Field	Note
0	Block length.	Number of bytes to follow + now includes also the data length byte (+1) as well as the data appended after EPC.
1	RSSI, signed 8-bit.	Negative number presenting the best RSSI for this tag.
2	Scaled RSSI.	Scaled RSSI value in range 0…100.
3…4	Timestamp.	Millisecond offset from the beginning of the inventory; last time the tag was seen (best RSSI).
5…8	Frequency in kHz.	DWORD value presenting the frequency that the tag was inventoried in (last time).
9	**Length of the data part in the EPCData field**	**Tells how many bytes in the EPCData field is resulting from the inventory + read operation. Note that this byte is 0 if the inventory + read type is 1 i.e. "data only".**
10…11	PC (+NSI/AFI) WORD.	Contents from the tag's EPC/UII memory's word address 1.
12	Channel (BYTE).	Channel number for the last time that the tag was inventoried.
13	Antenna ID.	Antenna from which the tag was inventoried from.
14…	EPC + data (EPCData).	The tag's EPC + appended with the read data.

5.8 Fetching single tag

With either "Get ID buffer" or "Get ID buffer with metadata" it is also possible to retrieve single tag's information. This can be done after an inventory command, not when inventory stream is running. The command is then appended with tag number ranging from 0 to last tag number – 1. The last tag is determined by the inventory response's "Tags in memory" -field .

The extended command structure is then:

Offset	Field	Note
0	Get ID buffer	0x06 = get ID buffer (antenna + EPC) 0x07 = get ID buffer (includes metadata)
1	DWORD	The tag number ranging from 0…stored tags – 1.

The response is the same as it is with getting the ID buffer but only contains one tag data.
If the tag number is too big the error 0x05 (INVALID_PARAMETER) is returned. In case there are currently no tags stored into the module's memory then error 0x020 (32, ERROR_NO_TAG) is returned.

5.9 Get reader information

The reader information contains:

- Serial number
- Alternative serial number (used w/ USB table reader configuration)
- Module name
- FCC ID string
- HW version (if used)
- SW version
- Number of GPIOs, sensors, supported regions and antennas.

Get reader information command

Offset	Field	Note
0	Get reader information.	Value is 9.

Get reader information response

The string members of the response are pre-pended with a length byte. After the command echo (0x09) and the status byte (0) the response is:

Byte(s)	Are
0...3	DWORD, version information version (1).
4	Length of the following serial number, <a> bytes
5...5+a-1	Serial number (ASCII)
5+a	Length of the following alternate serial number, <b> bytes.
6+a...6+b-1	Alternate serial (ASCII)
6+b	Length of the following reader name, <c> bytes.
7+b…7+c-1	Reader name (ASCII).
7+c	Length of the following FCC ID, <d> bytes.
8+c…8+d-1	FCC ID (ASCII)
8+d	Length of the following HW version, <e> bytes.
9+d…9+e-1	HW version (ASCII).
9+e…11+e	SW version e.g. 2, 5, ‘A’. 3 bytes.
12+e	Number of GPIO available GPIOs.
13+e	Number of available sensors.
14+e	Number of supported regions.
15+e	Number of available antennas.

5.10 Get versions

Note: this applies to NUR L2 module and devices that use the L2 module.

Command value for get versions is 0x0C (12). The command returns primary and secondary SW versions depending on whether the module is running bootloader or application.

Command

Offset	Field	Note
0	Get versions.	Value is 0x0C (12).

Response when running application:

Byte	Is
0	Application major version number
1	Application minor version number
2	Application build (ASCII, A – Z).
3	Bootloader major version number
4	Bootloader minor version number
5	Bootloader build (ASCII, A – Z).

Response when running bootloader:

Byte	Is
0	Bootloader major version number
1	Bootloader minor version number
2	Bootloader build (ASCII, A – Z).
3	Application major version number
4	Application minor version number
5	Application build (ASCII, A – Z).

5.11 Get FW information

This command returns an ASCII block giving information about the FW, version and module HW combination.

Command

Offset	Field	Note
0	Get FW info.	Value is 0x1E (30).

Example FW information response, NUR05W module bootloader:

Offset	Field	Note
0	Command echo	Value is 0x1E (30).
1	Status	0 (OK).
2…	ASCII string	<FWINFO>MODULE=NUR-05W;TYPE=LOADER;VER=1.9-A;DATE=Sep 27 2013 13:27:02</FWINFO>

Example FW information response, NUR05WL2 module application:

Offset	Field	Note
0	Command echo	Value is 0x1E (30).
1	Status	0 (OK).
2…	ASCII string	<FWINFO>MODULE=NUR-05WL2;TYPE=APP;VER=4.0-A;DATE=Sep 27 2013 13:28:52</FWINFO>

5.12 Beep

This command is available with the USB table reader. The call is always successful; values that are not present are set to defaults and values that are out of range are set either to minimum or maximum depending on whether they are too low (→ min) or too high (→ max). The command's parameter length is expected to be 9 bytes containing frequency, duration in ms and duty cycle. The default values are:

- Frequency = 1000Hz (1kHz)
- Duration = 200ms
- Duty cycle = 50%.

Beep command

Offset	Field	Note
0	Beep.	Value is 0x0D (13).
1	Frequency in Hz.	DWORD in range 500…3000.
5	Duration in ms.	DWORD in range 20…1000.
9	Duty cycle.	BYTE in range 5…95.

Beep response

Offset	Field	Note
0	Command echo.	Value is 0x0D (13).
1	Status.	Always 0 (OK).

5.13 Stop all continuous commands

This command is used to stop any ongoing continuous operations.

Stop all continuous commands command

Offset	Field	Note
0	Stop all continuous operations.	Value is 0x0E (14).

Stop all continuous commands response

Offset	Field	Note
0	Command echo.	Value is 0x0E (14).
1	Status.	Always 0 (OK).

5.14 Configure GPIO

The available GPIO pins are configured based on a given mask where each bit represent a GPIO pin. When the setup is written the command is expected to have the GPIO settings following the mask. Each GPIO pin has these attributes:

- Enabled / disabled
- Input / output
- Action
- Edge; rising or falling (affects action when input)

The GPIO configuration per pin is:

Byte	Field	Note	
0	Enabled / disabled	1 / 0.	
1	Type	**Value**	**Is**
		0	Output
		1	Input
		2	RF on indicator.
		3	RFID read indicator (scan single / inventory).
		4	Beeper (expects external beep device; active = '1').
		5	Antenna 1 control, if external switch connected; active = '1'
		6	Antenna 2 control, if external switch connected; active = '1'
2	Edge	**Value**	**Is**
		0	Negative (falling) edge
		1	Positive (rising) edge.
		2	Both edges.
3	Action	**Value**	**Is**
		0	NOP
		1	Notify
		2	Scan single tag
		3	Inventory (see also setup).

5.15 GPIO configuration command structure

Byte	Field	Note
0	Command	Value is 0x0E (15)
1	Flags	Each bit corresponds to a GPIO pin. Example: 0x05 = GPIO pin 1 (bit 0) + GPIO pin 3 (bit 2) • N= 2
2…2+4*N-1	GPIO configuration field.	Length is N * 4 bytes. Description above.

5.16 GPIO configuration example

The example sets GPIO pin 1, 2 and 4 as follows:

- GPIO 1: input, sends notification on both edges.
- GPIO 2: RF on indicator
- GPIO 4: external beeper

The flag field is then bit0 + bit1 + bit3 = 1 + 2 + 8 = 11 = 0x0B.

Example command:

Byte	Field	Note
0	Command	Value is 0x0F (15)
1	Flags	0x0B
GPIO 1 configuration block		
2	Enable	1
3	Type	1, input
4	Edge	2, both edges
5	Action	2
GPIO 2 configuration block		
6	Enable	1
7	Type	0, output
8	Edge	0
9	Action	2
GPIO 4 configuration block		
11	Enable	1
12	Type	0, output
13	Edge	0
14	Action	4

5.17 GPIO configuration response

The GPIO configuration can be queried with zero length command. The configuration response has the same structure except that the response always includes all the pin information i.e. the flag field indicates “all GPIO pins”.

5.18 GPIO configuration errors

Error	Value	Reason
INVALID_COMMAND	1	The module is configured as a USB table reader; the GPIO functionality is not supported.
INVALID_PARAMETER	5	When setting the flag field is 0.
INVALID_LENGTH	2	The GPIO configuration blocks' length is not as expected (nr GPIO multiplied by 4).

5.19 Get GPIO

This command returns the status of specified GPIO pins. The pins are defined as in the configuration command. Command length is expected to be 1; otherwise the error INVALID_LENGTH (0x05) is returned. If the mask field is 0 then error INVALID_PARAMETER (0x05) is returned.

Command:

Byte	Field	Note
0	Command	Value is 0x10 (16)
1	Mask	Flags as in configuration

Response:

Byte	Field	Note
0	Mask	As given.
GPIO 1 state		
1	Number	GPIO pin number
2	Enabled	1 / 0
3	Type	As in given configuration (0…6)
4	State	0 / 1 (always 0 if disabled)
GPIO <n> state		
…	…	Number of entries depends on the given mask's bit count.

5.20 Set GPIO

This command sets the states for those GPIO pins that have their type set to output. Again the mask field defines the pins and in this command the following bytes define the state, either 0 or 1. If the outputs given do not match the module's settings then error INVALID_PARAMETER (0x05) is returned. No other parameters are included thus the command length is expected to match the GPIO count + 1. Otherwise the error INVALID_LENGTH (0x02) is returned.

Command:

Byte	Field	Note
0	Command	Value is 0x11 (17)
1	Mask	Flags as in configuration
2	State 1	0 / 1
N	State <n>	As many as required.

Response:

Byte	Field	Note
0	Mask	Mask corresponding to output pins.
1	Pin number	GPIO number
2	State	0 / 1.
…	…	…byte pairs indicating the GPIO pin states if present.

5.21 Sensors

When the module is configured as a USB table reader then the sensor (tap and light) can be configured with this command. If the command is not supported then the INVALID_COMMAND error (0x01) is returned. If one sensor is defined with the mask field then the expected command length is 3; if both then the expected length is 5. The current configuration can be queried with parameter length 0. Length error (=length mismatch) is indicated with INVALID_PARAMETER (0x05).

Command:

Byte(s)	Field	Note
0	Sensor command.	Value is 0x12 (18).
1	Sensor mask	Bit 0 = '1' = tap sensor Bit 1 = '1' = light sensor
2	Enabled	0 / 1, 1 = enable
3	Action	Action as in GPIO except for the tap sensor; it does not have an edge configuration.
(4)	Enabled	0 / 1 (omitted if only one sensor defined with mask)
(5)	Action	Action as in GPIO except for the tap sensor; it does not have an edge configuration.

Response:

This response is appended also in the case if INVALID_PARAMETER.

Byte(s)	Field	Note
0	Tap sensor enabled	0 / 1
1	Tap sensor action	Action as in GPIO except for the tap sensor; it does not have an edge configuration.
2	Light sensor enabled	0 / 1.
3	Light sensor action	Action as in GPIO except for the tap sensor; it does not have an edge configuration.

5.22 Restart

This command causes the system to reboot thus restoring all startup defaults.

Restart command

Offset	Field	Note
0	Restart.	Value is 0x14 (20).

Restart response

Offset	Field	Note
0	Command echo.	Value is 0x14 (20).
1	Status	Always 0.

Note that the response is sent before the actual re-initialization.

5.23 Get device capabilities

The device capabilities query returns several reader specific capabilities that can be used to separate older generation HW from the new L2-family modules. The query is simple command having the value of **0x0B** and no parameters.

Device capabilities response

The device capabilities query has a 128-byte response. Current version is filled as follows:

Offset	Field	Note
0	Size.	DWORD, size reserved for the structure. Currently 128.
4	Flag set 1.	DWORD. 32 flag bits indicating various capabilities the module has. See "Device capabilities flag set 1".
8	Flag set 2.	DWORD. Currently not used.
12	Maximum TX power in dBm.	32-bit integer. For NUR05W and NUR05WL2 this value is 27 and for NUR10W module this value is 30.
16	TX attenuation value.	32-bit integer. Tells the attenuation step size when to power level is set in dBm. This value is 3.
20	Maximum TX level in mW.	16-bit unsigned integer. For NUR05W and NUR05WL2 this value is 500 and for NUR10W module this value is 1000.
22	TX attenuation steps.	16-unsigned integer that tells how many steps there are available with TX attenuation. This value is 20 (0…19).
24	Tag buffer size.	Number of 96-bit EPC data that the module can currently hold.
26	Antenna count.	Maximum number of antennas that the current configuration can have.
28	GPIO count	Maximum number of GPIOs that the current configuration can have.
30	Reader chip version.	1 = AS3992 2 = AS3993
32	Module type.	WORD. The type of the module: 1 = NUR 05W 2 = NUR05WL ("XNUR") 3 = NUR05WL2 ("NUR L2") 4 = NUR10W ("NUR L2 1W")
34	Configuration flags.	DWORD. Defines how the module is configured for use, see configuration bits.
32…127	Reserved.	Currently not used.

NOTE: when the module runs bootloader then the first valid field is reader chip version. Other are set o zero.

5.24 Device capabilities: flag set 1

The device capabilities' flag set holds the following information:

Bit mask	Meaning
0x00000001 (bit 0)	If '1' then the RX decoding of FM0 is upported.
0x00000002 (bit 1)	If '1' then the RX decoding of Miller-2 is upported.
0x00000004 (bit 2)	If '1' then the RX decoding of Miller-4 is upported.
0x00000008 (bit 3)	If '1' then the RX decoding of Miller-8 is upported.
0x00000010 (bit 4)	If '1' then the link frequency of 40kHz is supported.
0x00000020 (bit 5)	If '1' then the link frequency of 80kHz is supported.
0x00000040 (bit 6)	If '1' then the link frequency of 160kHz is supported.
0x00000080 (bit 7)	If '1' then the link frequency of 256kHz is supported.
0x00000100 (bit 8)	If '1' then the link frequency of 320kHz is supported.
0x00000200 (bit 9)	If '1' then the link frequency of 640kHz is supported.
0x00000400, 0x00000800 (bits 10 and 11)	Reserved.
0x00001000 (bit 12)	If '1' then the device is configured as a Sampo table reader that has a beeper present.
0x00002000 (bit 13)	If '1' then the device is configured as a Sampo table reader that has a light sensor present.
0x00004000 (bit 14)	If '1' then the device is configured as a Sampo table reader that has a tap sensor present.
0x00008000 (bit 15)	If '1' then the device is capable of performing an antenna tuning i.e. to better match an antenna attached to the device.
0x00010000 (bit 16)	If '1' then the device is capable of performing an channel scan operation.
0x00020000 (bit 17)	If '1' then the device supports inventory + read.
0x00040000 (bit 18)	If '1' then the device support per antenna power settings.
0x00080000 (bit 19)	If '1' then the device is configured to use beam forming antennas (area reader).

5.25 Device capabilities: configuration bits

Bit mask	Meaning
0x00000001 (bit 0)	Module is used in a USB table reader.
0x00000002 (bit 1)	Module is used in an Ethernet table reader.
0x00000004 (bit 2)	Module is used in a Stix mini reader.
0x00000008 (bit 3)	NOT USED
0x00000010 (bit 4)	NOT USED
0x00000020 (bit 5)	The module HW is configured to operate in 1W mode.
0x00000040 (bit 6)	The module can control area reader's beam forming antennas.

6 Control commands

6.1 Set baudrate

When connected with RS232 port this command sets the baudrate. Note that the response to this command is sent before changing the baudrate thus the next command is expected with the new baudrate. The baudrate is given in a single byte. Valid values and the corresponding baudrates are:

Value	Baudrate
0	115200
1	230400
2	500000 (500k)
3	1000000 (1M)
4	1500000 (1M5)
5	38400
6	9600

Set baudrate command

Note that if the parameter length is not 1, then the module only echoes back the value. Such an approach can be used to detect the baudrate.

Offset	Field	Note
0	Set baudrate.	Value is 0x20 (32).
1	Baudrate to set.	Range is 0...4.

Set baudrate response

Offset	Field	Note
0	Command echo.	Value is 0x20 (32).
1	Status	0 if successful. 5 (INVALID_PARAMETER) if the baudrate is not in range.
2	Current setting.	In range 0…6 if successful.

6.2 Load / read setup

! WARNING !

By changing the link frequency it is possible to create a region / link frequency combination that does not meet the requirements of your local UHF ISM band radio regulations.

In the command (value is 0x22) there is a DWORD (32-bit) length flag field that tells the module which ones of the given parameters are to be modified or echoed back. Several values are modified or read by combining the flag values with an OR-operation. The flag values are:

Flag value	Indicates presence of
0x0001 (bit 0)	Link frequency (DWORD, 160k, 256k or 320k).
0x0002 (bit 1)	Receiver decoding (BYTE, 1..3 = M2-M8)
0x0004 (bit 2)	TX level (BYTE, range = 0…19 = 27…8dBm).
0x0008 (bit 3)	TX modulation (BYTE, 0 = ASK, 1 = PRASK)
0x0010 (bit 4)	Region (BYTE, 0…6)
0x0020 (bit 5)	Inventory Q value (BYTE, 0…15).
0x0040 (bit 6)	Inventory session value (BYTE, 0…3).
0x0080 (bit 7)	Inventory rounds (BYTE, 1…10).
0x0100 (bit 8)	Antenna mask (BYTE, each bit denotes one antenna where '1' = enabled).
0x0200 (bit 9)	Single scan timeout in ms (WORD, 50…500)
0x0400 (bit 10)	Inventory timeout (WORD, max = 60000ms = 60s). Applied to the inventory run by a GPIO trigger.
0x0800 (bit 11)	Selected antenna (BYTE, 0xFF or antenna number 0…7).
0x1000 (bit 12)	Operation flags, DWORD: Bit 0: when '1', hop events are enabled. Bit 1: when '1' the inventory stream sends also "zero counts" i.e. rounds where no tags were found. Bit 12: when '1' the autotune will send events (**L2 family only**).
0x2000 (bit 13)	Inventory target, BYTE value: - 0 = A - 1 = B - 2 = AB
0x4000 (bit 14)	EPC length during inventory, BYTE and in bytes. Set to 0xFF to allow all EPC lengths.
0x8000 (bit 15)	Read RSSI filter values that are signed 8-bit values giving the range where the tag is still read: minimum followed by maximum value. Set both to 0 to allow all.
0x10000 (bit 16)	Write RSSI filter values that are signed 8-bit values giving the range where the tag is still written. Values as in read.
0x20000 (bit 17)	Inventory RSSI filter values indicating in which range the found tag is to be stored. Setting as in read and write.

Load / read setup flags continues

These flags and values are available in L2-family of modules only.

Flag value	Indicates presence of
0x40000 (bit 18)	Read timeout field is present (WORD, range is 50…1000 ms).
0x80000 (bit 19)	Write timeout field is present (WORD, range is 200...2000 ms).
0x100000 (bit 20)	Lock timeout field is present (WORD, range is 200...2000 ms).
0x200000 (bit 21)	Kill timeout field is present (WORD, range is 200…2000 ms).
0x400000 (bit 22)	Power save setup for continuous inventory. Range is 0…3 where: 0 = not in use (factory default for a module) 1 = a maximum of 1000ms of quiet time between inventories 2 = a maximum of 500ms of quiet time between inventories (factory default for STIX mini-reader) 3 = a maximum of 100ms of quiet time between inventories
0x800000 (bit 23)	Per antenna power setting. There are 4 bytes that define specific maximum TX level for each antenna; this setting overrides the TX level setting when it is in range 0…19.
0x1000000 (bit 24)	Antenna power offset. There are four bytes reserved for power offset. The offset is -1…1. The first value is used and the rest 3 are RFU.
0x2000000 (bit 25)	Extended antenna mask. Introduced for area / multiport readers. DWORD value makes it possible to control up to 32 antennas in total. **L2-family with FW 5.0-A or higher.**
0x4000000 (bit 26)	Automatic tuning setup is present. Consists of two bytes, see automatic tuning and related information. **L2-family with FW 5.0-A or higher.**
0x8000000 (bit 27)	Extended per antenna power setting. **L2-family with FW 5.0-A or higher.**
0x10000000 (bit 28)	Receiver sensitivity setting. **L2-family with FW 5.0-A or higher.**

6.3 Module setup: region numbers

See "Get region information" command for more region information.

Number	Is	Channels	Max. channel time in ms
0	EU	4	1000
1	North-America	52	383
2	People's Republic of China (upper band)	16	400
3	Malaysia	7	400
4	Brazil	26	400
5	Australia	11	400
6	New Zealand	16	400
7	Japan (250mW max power, NOTE: LBT)	19	1000
8	Japan (500mW max power)	4	1000
9	Korean	6	1000
10	India	3	1000
11	Russia	3	1000
12	Vietnam	10	400
13	Singapore	10	400
14	Thailand	10	400
15	Philippines	4	400

6.4 Load setup example

Present = link frequency, TX level, scan timeout, OP flags, read timeout and write timeout:

Byte(s)	Field	Note
0	Load setup.	Value is 0x22 (34).
1…4	Flag field.	DWORD: + 0x00000001 : LF flag + 0x00000004 : TX level flag + 0x00000200 : scan timeout + 0x00001000 : OP flags + 0x00020000 : inventory RSSI limits. + 0x00040000 : read timeout value. + 0x00080000 : write timeout value. = 0x000E1205
5…8	Link frequency	DWORD, unit is Hz. Valid values are: • 160000 (160k) • 256000 (256k) • 320000 (320k) Set if the link frequency flag (bit 1) is set in the flag field.
9	TX level	0 = 27dBm (~500mW). 19 = minimum (27-19)dBm = 8dBm
10…11	Scan timeout	Range is 50…500. Unit is ms.
12	TX level.	Transmission level. This value represents the attenuation (attn) from the maximum level (27dBm) in dBm. Range is 0…19 thus the TX level can be set from 8 dBm (attn=19, ~6 mW) to 27 dBm (attn=0, ~500mW). Set if the TX level flag (bit 3) is set in the flag field.
13…16	OP flags	E.g. value 0x00000002 would mean Bit 0 = ‘0’ = hop events disabled Bit 1 = ‘1‘ = inventory stream also notifies of empty rounds.
17…18	Inventory RSSI limits	Byte 1 = RSSI low (signed 8-bit) e.g. -65 (0xBF) Byte 2 = RSSI high (signed 8-bit) e.g. -55 (0xC9)
19…20	Read timeout value	WORD 0x01F4 (=500, in ms).
21…22	Write timeout value	WORD 0x03E8 (=1000 ms).

The response that follows the command and status byte is in the same format as the example shown above.

6.5 Module setup: field summary

When all the module setup flags are present:

Offset	Field	Is
0	Link frequency.	DWORD: 160000, 256000 or 320000.
4	RX decoding	0…3 corresponding to 2^{RXDec} = FM0, M2, M4 and M8.
5	TX level	Range is 0…19 giving dBm 27 - <*tx_level*>. In 1W module the range dBm range is 30 - <tx_level>.
6	TX modulation	0 = ASK, 1 = PR-ASK.
7	Region.	0…15, see "Setup's region numbers".
8	Inventory Q.	Inventory's default Q parameter, range is 0…15. Zero mean automatic and is default.
9	Session.	Range is 0…3 and default is 0.
10	Number of inventory rounds.	Range is 0…10: zero is automatic and default.
11	Antenna mask.	Range is 1..0x0F. Each bit marks an enabled antenna (bit 0…3).
12	Triggered scan single timeout.	WORD. Timeout when scan single is triggered by a sensor or GPIO, in milliseconds. Range is 50…500 (default).
14	Triggered inventory timeout.	WORD. Timeout when inventory is triggered by a sensor or GPIO, in milliseconds. Maximum value is 60000 (one minute). Zero means that the channel timeout is used. Default is 1000.
16	Selected antenna.	0xFF means automatic selected from the enabled antennas. Otherwise setting must match to the selected antenna mask.
17	Operation flags.	DWORD. Controls the reader behavior in various situations. Bit 0: when set to '1' causes the module to send an event when the frequency changes. Bit 1: when set to '1' the module also sends notification during inventory stream when no tags were found during last inventory operation.
21	Default inventory target.	Valid values: 0 = A, 1 = B, 2 = toggle between A and B.
22	Forced EPC length in inventory.	0xFF means all. Otherwise the range is 0…62 (bytes).
23	Read RSSI filter low limit	Filters out tag for reading if the RSSI is not between these limits. Default is 0 = not used.
24	Read RSSI filter high limit	
25	Write RSSI filter low limit	Filters out tag for writing if the RSSI is not between these limits. Default is 0 = not used.
26	Write RSSI filter high limit	
27	Inventory RSSI filter low limit	Filters out the tags during inventory that are not between these values. Default is 0 = not used.
28	Inventory RSSI filter high limit	

When all the module setup flags are present continues:

Offset	Field	Is
29	Default read timeout value.	WORD. Tag read timeout in milliseconds. Range is 50…500 (default).
31	Default write timeout value.	WORD. Tag write timeout in milliseconds. Range is 200…2000, default is 1000.
33	Default lock timeout value.	WORD. Tag lock timeout in milliseconds. Range is 200…2000, default is 1000.
35	Default kill timeout value.	WORD. Tag kill timeout in milliseconds. Range is 200…2000, default is 1000.
37	Auto-period setup.	WORD. Defines the duty cycle during continuous inventory and tag tracing. This value can be used to significantly reduce the current consumption in mobile applications. 0 = not used. 1 = maximum of 1000ms off time 2 = maximum of 500ms off time (Stix mini-reader default) 3 = maximum of 100ms off time (default for all others) Note: with Stix mini-reader this value cannot be set to 0.
39	Per antenna maximum TX level setting.	Per antenna offsets 39, 40, 41 and 42: 0xFF = not used (TX level overrides) Range 1…19: this setting per antenna overrides the TX level setting. Default value is 0xFF (not used).
43	Power offset value.	4 bytes. First value is used, other are reserved and ignored. Signed char, valid values: -1, 0 and 1. This offset is added to the TX level. When the value is -1 then the offset is added to the TX level meaning that is decremented (as the value is negative) and vice versa with value 1. Default value is 0.
47	Extended antenna mask.	DWORD. Allows up to 32 antennas' separate enabling. Current maximum used is bit7 (18 antennas, area reader).
51	Autotune setup.	Two bytes. First is control byte, second one is 8-bit signed threshold value. Explained in detail here.
53	Extended per antenna power.	32 bytes. Each byte states per antenna power as the TX level setting does. Set to 0xFF if the antenna specific power is not in use. Overrides the TX level setting for corresponding antenna.
85	Receiver sensitivity.	One byte. The settings are: 0: nominal; no special changes in reception 1: low; the receiver attenuated below the nominal level 2: high; the receiver uses high gain settings

6.6 Get region information

This command returns the region's information. The region can be currently selected (no parameters) or the information can be queried for a region. See the list of regions.

Get region information command

Offset	Field	Note
0	Get region information.	Value is 0x24 (36).
1	Region number	Optional. Range is 0…15. See list.

Get region information response

Offset	Field	Note
0	Command echo.	Value is 0x24 (36).
1	Status	0 if successful. 5 (INVALID_PARAMETER) if queried region number is not in range (0…15). In this case no additional data is appended.
2	Id	Number of the region / hop table.
3	Channel 0 center frequency.	DWORD, in kHz.
7	Channel width.	DWORD, in kHz.
10	Last index.	BYTE, last index i.e. number of channels-1.
11	Channel time.	DWORD, maximum channel time in ms.
15	Table name length	Number of bytes (N) to follow (name length).
16	Hop table name.	ASCII characters, N characters.

6.7 Store setup

This command saves the current setup to the module's non-volatile memory. The command also takes 4 possible flag bits as parameters.

Store setup command

Offset	Field	Note	
0	Store setup.	Value is 0x28 (40).	
1	What to store	Bits having the meaning	
		Bit	**Stored group**
		0x01, bit 0	Store group 1 if set
		0x02, bit 1	Store group 2 if set
		0x04, bit 2	Store group 3 if set
		0x08, bit 3	Store group 4 if set

Corresponding groups

NOTE: no saving is done in a group where no changes are detected when compared to a previously stored setup.

Group	Flag value	Includes
1	0x01	When bit 0 is set then the saved settings are: 1. Region 2. Link frequency 3. Receiver decoding 4. TX level 5. Modulation 6. Selected antenna 7. Antenna configuration 8. Default inventory Q 9. Default inventory session 10. Default inventory target 11. Expected EPC length in inventory 12. Triggered inventory timeout 13. Single tag scan timeout 14. Read, write, lock and kill timeout. 15. STIX reader auto inventory period.
2	0x02	When bit 1 is set then the saved settings consists of (if change is detected) GPIO and sensor configuration.
3	0x04	When bit 2 is set then default baudrate is stored.
4	0x08	When bit 3 is set then operation flags are stored.

Store setup response is always OK.

7 RF behavior and performance

7.1 Tune antenna (L2-family)

The antenna tune command requires the L2-family hardware in order to work properly. There are several ways to control how the tuning is done. Physically the module has two digitally controlled capacitors that are able to find the best match for the currently connected antenna. In RF terms this means that a setting with the lowest leakage from the TX branch to the RX branch is iterated by controlling the capacitor values.

The tuning areas are divided into 6 frequency bands from which the closest tuning value is always retrieved when the module hops to next frequency. The bands' center frequencies are 850, 870, 890, 910, 930 and 950MHz.

7.2 The tune command

Basic tuning takes the following parameters:

Offset	Field	Note
0	Tune antenna.	Value is 0x66 (102).
1	Tuning type.	DWORD. Valid values are: 0 = fast (narrowest capacitor adjustment) 1 = medium 2 = wide 3 = full (all possible values are iterated = 32 x 32 values = 1024 in total). Can cause error 5 (invalid parameter).
5	Antenna	DWORD. Antenna to use. Range is 0…3. Can cause error 5 (invalid parameter). This error may also occur if the antenna given is not present i.e. not configured to be in use.
9	Band.	DWORD Range is 0…5 (see above). A DWORD value of 0xFFFFFFFF causes the tuning to be done for all bands. Can cause error 5 (invalid parameter).
13	Save result.	If set to 1 then the result is saved to the user tune region of the module's non-volatile memory. Otherwise the result is just taken into use and will not be available after next boot.
17	Result that is "good enough",	32-signed integer. Typically any result that is negative can be considered as "good". This is the value that determines that the tuning round is to be stopped if the result is below this limit. Typically set this value to e.g. -50 so that the tuning does not stop i.e. the best possible value for that antenna + band combination is found.
18…26	Reserved.	8 bytes that have to be set to all 0's. Reserved for future extensions.

7.3 Antenna tune response

Tuning response with all the bands selected is:

Bytes(s)	Field	Note
0	Command echo	0x66
1	Status	0 = OK
2…5	Antenna	DWORD. The antenna that was used in the tuning.
6...17	3 x DWORD; reserved.	Set to 0.
18…29	Band 0 (850 MHz) results.	Each of the results contains 3 32-signed integer values: I, Q and dBm values. The dBm value is multiplied by 1000. Therefore, to get the actual value it needs to be converted to a floating point value and then be divided by 1000.0. Contents for each band with an example is shown below.
30…41	Band 1 (870MHz) results (“EU band”).	
42…53	Band 2 (890MHz) results.	
54…65	Band 3 (910MHz) results (“FCC lower band half”).	
66…77	Band 4 (930MHz) results (“FCC upper band half”).	
78…89	Band 5 (950MHz) results.	
90...91	CRC-16	Over bytes 0...89.

Tuning response with one band selected is:

Offset	Field	Note
0	Band number (0…5)	DWORD. The band that was used in the tuning.
4…7	I value of the best result.	32-bit signed integer.
8…11	Q part of the best result.	32-bit signed integer.
12…15	dBm value for the best result.	32-bit signed integer, multiplied by 1000.

Important note on tuning: some of the combinations may take very long time to complete. For such cases the user must take care that the communication timeout is long enough. The longest tuning rounds may take up to 20 seconds.

Example result. The bytes are represented within the result, see the actual offsets above. The I- and Q-values are present for development purposes; the useful value in the structure is the reflected power field (P_{ref})

Bytes	Contents	Values, little-endian (DEC)	Meaning
0...3	06 00 00 00	0x00000006 (6)	Best result’s I-value
4…7	FC FF FF FF	0xFFFFFFFC (-4)	Best result’s Q-value
8...11	67 B2 FF FF	0xFFFFB267 (-19865)	P_{ref}=-19.865 dBm.

7.4 Restoring tune from factory defaults or user saved values

The tuning can be reverted to either the one that was saved during production or to the one that were saved by user. Command structure is:

Offset	Field	Note
0	Tune antenna.	Value is 0x66 (102).
1…4	From what region the tuning is restored from.	DWORD. 0 = take the last user stored tuning into use 1 = take the factory saved tuning into use 2 = override everything with factory default (both user area and the currently used values) Can cause error 5, invalid parameter.
5…16	12 reserved bytes.	Set to 0.

7.5 Saving the currently used tuning values

If e.g. for test purposes the tune command was called without setting the save field to 1 the currently used values can be set as user defaults. In such a case the command structure is like:

Offset	Field	Note
0	Tune antenna.	Value is 0x66 (102).
1…4	What to save.	When set to 0 then all the tuning values are saved into the user area. Otherwise this parameter denotes antenna and valid range is 1…4 corresponding to antenna number 0…3. Can cause error 5, invalid parameter.
5…8	Reserved.	Set to 0.

This command is responded with a simple OK response if the page programming was successful. If the page programming failed, then the command responds with not ready error, code 0x0D (13).

7.6 Automatic tuning

In the L2-family of modules, from FW version 5.0-A and above, it is possible to set the module into automatic tuning mode. This means that the module, by itself, periodically checks the reflected power during e.g. inventory stream and tries to find better internal match for the connected antenna.

7.7 Automatic tuning: enabling and disabling

The autotune requires the FW version 5.0-A or above (as well as NUR API version 1.7.9.1 or above). It is enabled and disabled via module setup setting. The autotune can also be set up to send notifications giving information (namely behavior monitoring) about the last performed on-the-fly tuning operation.

In the modules setup's autotune field there are two bytes: first one is for control and second one is a signed 8-bit integer representing the threshold that triggers the autotune if it is used. If the autotune is enabled, threshold is used and the calculated reflected power in dBm is higher than the threshold value then the automatic tuning is performed. The use of the threshold can also be disabled when the autotune itself is enabled. In this case the autotune is done every time the reflected power is measured.

7.8 Autotune control bytes

Byte	Meaning
0	Control byte. The used bits are: • bit0: enable (1) or disable (0) the automatic tuning • bit1: enable (1) or disable (0) the use of the following threshold value
1	Signed 8-bit integer (char) representing the autotune threshold value in dBm. The second byte (char) can have value -127…0. Typically the value could be e.g. in range -10…-5. Finding the best performing value is matter of trial and error as it depends heavily on the environment the reader is used in.

7.9 Autotune notification

Through the operation flag field it is possible to set the autotune to send a notification each time the autotune is performed. Contents of the autotune notification packet is:

Offset	Field	Note
0	Notification code.	0x8D (autotune event)
1	0	(status)
2	Tuning capacitor 1 value.	Range is 0…31. *Usable namely for NID R&D personnel in case of anomalies.*
3	Tuning capacitor 2 value.	
4	Signed 32-bit little-endian integer: reflected power.	Represents the calculated reflected power in dBm multiplied by 1000. For example -11500 = -11.5 dBm.
8	BYTE: antenna.	The antenna number for which the tuning was done for.
9	DWORD: frequency	The tuning frequency in kHz.

7.10 Carrier time extension (L2-family)

In L2-family modules from FW 5.0-A onwards it is possible to extend the carrier ON-time in order to for example perform two custom exchanges where the latter depends on the first and thus requires the carrier to stay on over both of the operations.

7.11 Carrier time command

Offset	Field	Note
0	Carrier command	Value is 0x62 (98).
1	ON/OFF	BYTE, If 1 carrier is set ON.

Bytes	Field	Note
0	Carrier command	Value is 0x62 (98).
1...4	ON/OFF	DWORD. If 1 then the carrier time is extended starting from the current time + current region's channel time. With 0 the carrier is set OFF. Use only values 0 or 1.

7.12 Carrier time response

Offset	Field	Note
0	Carrier command echo	Value is 0x62 (98).
1	Status	0 = OK. Error 0x05 (INVALID_PARAMETER) if the DWORD parameter is not 0 or 1.

7.13 Getting reflected power

Getting the reflected power provides means to check how much the transmission branch in the module leaks to the receiver side.

The reflected power is either measured in certain frequency (given as a parameter) or in the current region's middle channel (no parameters given).

7.14 Get reflected power command

Byte	Field	Note
0	Command code	0x61 (get reflected power)
1...4	Frequency	DWORD, optional. Frequency in kHz. Being out of the allowed range 840000... 960000kHz causes error 0x05 (INVALID PARAMETER).

7.15 Get reflected power response

The response consists of the receiver's I- and Q-parts, and a special divisor. How to calculate the actual reflected power in dBm is shown below the response structure.

Byte	Field	Note
0	Command code	0x61 (get reflected power)
1	Status	0 when no error. With error the rest is not included.
2...5	I-part	32-bit signed integer.
6...9	Q-part	32-bit signed integer.
10...13	Divisor.	32-bit signed integer.

7.16 Calculating the reflected power

The following formula is used to calculate the reflected power in dBm (RP_{dBm}) using I-part (I), Q-part (Q) and divisor (DIV):

$$RP_{dBm} = 20 \times \log_{10}\left(\frac{\sqrt{I^2 + Q^2}}{DIV}\right)$$

8 Gen2 commands

8.1 Scan single tag

This command tries to scan single tag by doing a one round inventory i.e. using the Q value of 0. The command can be instructed to use timeout. If the timeout is not specified then the command uses default timeout. For the module alone the default timeout is 100 ms and for the USB table reader the default timeout is 1000 ms.

Scan single tag command

Offset	Field	Note
0	Scan single tag.	Value is 0x30 (48).
1…2	Timeout.	WORD, in milliseconds. The range is 50…500 ms. If out of the range then the timeout is either set to minimum (too small) or maximum (too high).

Scan single tag response (tag scanned)

Offset	Field	Note
0	Command echo.	Value is 0x30 (48).
1	Status	0
2	Antenna ID	0…3: the source antenna.
3	RSSI	Signed 8-value for the RSSI.
4	Scaled RSSI.	Scaled RSSI value, 1…100.
5…N	Tag's EPC.	EPC bytes, rest of the packet excluding the CRC-16.

Scan single tag error

Offset	Field	Note
0	Command echo.	Value is 0x30 (48).
1	Error	0x20 (ERROR_NO_TAG): no tag was scanned within the given time.

8.2 Reset to target

With this command it is possible to reset the tags' inventoried flags into known state either A or B. This command is done per each antenna and in given session.

Reset to target command

Offset	Field	Note
0	Reset to target	Value is 0x3A (58)
1	Session	0…3.
2	Target	0 = B, others = A.

When successful the command sends a simple OK response without parameters.

Two errors can occur with this command:

Offset	Field	Note
0	Reset to target	Value is 0x3A (58)
1	Error	INVALID_LENGTH (0x02): command too short i.e. less than 2 bytes. G2_ERROR_WRITE (0x40): undefined error during the air operation.

8.3 Simple inventory

For the simple inventory command it is possible to give 3 parameters. Length of the command parameters is required to be 0, 2 or 3 bytes depending on what parameters are used. Otherwise the command will return error 5 (INVALID_PARAMETER) as the status byte followed by the error flags. Parameter length explained:

Length	Meaning
0	None: use default settings stored into the module (load / read setup).
2	Parameters are Q and session.
3	Parameters are Q, session and number of inventory rounds.

Simple inventory command

Offset	Field	Note	
0	Inventory.	Value is 0x31 (49).	
1	Q	Q-parameter: 0…15.	Can be left out if the number of rounds parameter is also left out.
2	Session	Session: 0…3	
3	Number of rounds	Maximum number of rounds for the inventory. Can be omitted.	

8.4 Inventory with select parameters

Inventory with select parameters adds the possibility to use bit mask applied to EPC/UII, TID or the user memory banks thus making only certain part of the tag population to participate in the inventory round. This part can be for example tags that have a common part of the EPC memory the same like e.g. SSCC-96 or SGTIN-96 company prefix. If the parameter length for this command is less than 13 bytes then the command returns with value of 5 (INVALID_PARAMTER) as the status byte followed by the error flags.

Inventory select command

Offset	Field	Note
0	Inventory w/ select.	Value is 0x32 (50).
1	Q	Q-parameter, must be present.
2	Session	Session parameter, must be present.
3	Number of rounds	Maximum number of rounds for the inventory.
4	Selection block's size.	Number of bytes to follow.
5	Select bank.	Bank that the selection mask is applied to. Range is 1…3. Error causes the bank error flag to be set (bit 2).
6	Flags.	Option bits: 0x01 (bit 0): the selection is done as inverted i.e. the tags that do not match the selection criteria will participate in this inventory round. 0x02 (bit 1): if 1 then use extended addressing. This is to say that the following mask bit address is 64-bit (QWORD) instead of 32 bits (DWORD).

8.5 Inventory select using 32-bit addressing

Byte(s)	Field	Note
7	Bit address.	DWORD, this is the bit address of the mask.
11	Bit length.	This is the bit length of the mask (BL). It is expected that this field is followed by a number of mask bytes that is in terms of C given by (using integers) *N = (BL / 8) + ((BL & 7) != 0).* See example.
12…12+N-1	Mask.	Data for the bit mask to be used during the inventory round.

8.6 Inventory select using 64-bit addressing

Byte(s)	Field	Note
7	Bit address.	QWORD, this is the bit address of the mask.
15	Bit length.	This is the bit length of the mask (BL). It is expected that this field is followed by a number of mask bytes that is in terms of C given by (using integers) *N = (BL / 8) + ((BL & 7) != 0).* See example.
16…16+N-1	Mask.	Data for the bit mask to be used during the inventory round.

8.7 Selection mask example

Assuming that the given selection length is 34 bits it is then required that the selection mask byte length is (using integers)

N = (34 / 8) + ((34 & 7) != 0) = 4 + 1 = 5 bytes.

8.8 Inventory response

Offset	Field	Note
0	Command echo.	For inventory value is 0x31 (49). May also be 0x3A for extended inventory.
1	Status.	0: OK 0x1F, error: tag buffer full
2	Tags found.	WORD, number of tags found during last inventory execution.
4	Tags in memory.	WORD, total number of tags currently stored into the module's memory.
6	Rounds.	Number of inventory rounds executed.
7	Collisions.	WORD, number of events interpreted as collisions.
9	Current Q.	Q-parameter: when automatic Q-adjusting is in use this presents the last used Q. Otherwise it is the same as the given value for the Q-parameter.

8.9 Inventory parameter errors

This is the response that will always be sent if there is parameter or parameter block length error(s).

Offset	Field	Note	
0	Command echo.	Value is 0x31 (49).	
1	Status.	5 (INVALID_PARAMETER)	
2	Error flags.	A set of bits explaining what was interpreted as an error	
		Flag	**Is**
		0x01 (bit 0)	Q is greater than 15.
		0x02 (bit 1)	Session is greater than 3.
		0x04 (bit 2)	Inventory parameter length error.
		0x08 (bit 3)	Given selection mask length does not match to expected or it is 0.

For example with incorrect Q and session parameters then the error flag field would have the value 0x02 + 0x04 = 0x06.

8.10 Configure inventory + read (L2-family)

The L2-family modules implement inventory and read in a single operation. Once configure the inventory + read applies to all inventories until canceled. Also all other restrictions apply to this operation: antenna selection, power save configuration, read RSSI level etc.

The inventory + read is configured as follows. Note that this configuration does nothing by itself; this configuration tells the module how the various inventories after this configuration should behave.

Offset	Field	Note
0	Configure inventory + read command.	Value is 0x41 (65).
1	On / off.	If this byte is set to non-zero value then the inventory + read is taken into use. NOTE: if the command's length is only one byte then it can be used as a simple on/off control. If the inventory + read is turned on and the parameters currently stored into the module are not valid then the command result in error 5, invalid parameter.
2	Type	Valid values are: 0 : the module stores EPC + related data. Response in this case differs from the one received when querying metadata buffer. 1 : the module stores the read data only. This is useful e.g. when implementing a TID-inventory where the tag's hard-coded TID serial is the matter of interest. In this case the read data is returned in the place of EPC when getting either the ID buffer or meta buffer. Other values currently result in error 5, invalid parameter.
3	Bank	Bank to read from. Valid values are 1 (PEC), 2 (TID) and 3 (user memory). Other values cause error 5, invalid parameter.
4	Word address.	DWORD. Configures the read's word address within the bank.
8	Word count.	Range is 1…32 (2…64 bytes). Other values cause error 5, invalid parameter.

See: response structure with inventory + read.

9 Tag singulation

When a specific tag is to be addressed for a read, write etc. operation the payload's generic structure is like this:

Common block	Singulation block (optional)	Information block (read, write, lock etc.)
Flags + password	Select information	Address, data, lock payload etc.

9.1 Common block

The common block specifies the flag field and the password. This block is always present and the flag field specifies the following data structure and its format. Common block's structure is:

Offset	Field	Note
0	Flags	
1…4	Password	Password

The flag bit set construction:

Bit	7	6	5	4	3	2	1	0
Is	RFU	RFU	RFU	RFU	EA2	EA1	SBP	SEC

Bits explained:

- SEC: operation is secured. With read and write the password that follows the flags is used ignoring the contents of the password. With kill or lock commands the SEC bit is ignored and the password is used as is. Value used in a mask: 1.
- SBP: Singulation Block Present. The common header is followed by the singulation data. Value used in a mask: 2.
- EA1: Extended Address 1. In the singulation data the address is extended i.e. 64-bit instead of 32-bit. Value used in a mask: 4.
- EA2: Extended Address 2. In the read and write the address is extended i.e. 64-bit instead of 32-bit. Value used in a mask: 8.

Example: constructing a flag set that indicate secured operation with singulation block and extended R/W address:

SEC (1) + SBP (2) + EA2 (8) = 11 (0x0B).

9.2 Singulation block: 32-bit addressing

The singulation block that is optionally present in the command is constructed as follows when the EA1 flag is '0' i.e. the singulation uses 32-bit bit address:

Offset	Field	Is
0	Block size	Number of bytes to follow; size of this block excluding the size byte. Block's integrity is checked against this length byte.
1	Bank	Bank: range is 1…3.
2	Bit address	DWORD, 32-bit bit address.
6	Mask's bit length	WORD, number of bits used in the mask.
8…n	Mask	Mask data. Mask length must match the bit length field requirement in terms of C: Number of mask bytes = (BL/8) + ((BL%8)!=0)

9.3 Singulation block: 64-bit addressing

The singulation block that is optionally present in the command is constructed as follows when the EA1 flag is '1' i.e. the singulation uses 64-bit bit address:

Offset	Field	Is
0	Block size	Number of bytes to follow; size of this block excluding the size byte. Block's integrity is checked against this length byte.
1	Bank	Bank: range is 1…3.
2	Bit address	QWORD, 64-bit bit address.
10	Mask's bit length	WORD, number of bits used in the mask.
12…n	Mask	Mask data. Mask length must match the bit length field requirement in terms of C: Number of mask bytes = (BL/8) + ((BL%8)!=0)

9.4 Information block errors

When the common start block, singulation block or the read/write/etc. information block parsing failed the response is:

Offset	Field		
0	Command echo	Applicable read, write etc. command value.	
1	Status / error	Error code: 5 (INVALID_PARAMETER)	
2	Error type	Specifies what was wrong with parameters.	
		Code	**Is**
		1	Singulation block not present or invalid (SBP='1').
		2	Singulation mask data not present or invalid.
		3	Singulation data bank is 0 or greater than 3.
		4	Read, write or lock bank error.
		5	Read information block size error.
		6	Write information block size error.
		7	Read length is 0.
		8	Write length is 0 or data length and word count mismatch.
		9	Lock parameter / information block error.
		10	Not supported; command was recognized but not currently supported.

9.5 Read: 32-bit addressing

When the EA2 flag in the common block's flag field is '0' then the read information block uses 32-bit addressing as follows:

Offset	Field	Is
0	Block size	Number of bytes to follow.
1	Bank	Bank: range is 0…3.
2	Word address	DWORD, 32-bit word address as specified by Gen2.
6	Word count	Word count, range is 1…255, 0 causes error 5 (INVALID_PARAMETER).

9.6 Read: 64-bit addressing

When the EA2 flag in the common block's flag field is '1' then the read information block uses 64-bit addressing as follows:

Offset	Field	Is
0	Block size	Number of bytes to follow.
1	Bank	Bank: range is 0…3.
2	Word address	QWORD, 64-bit word address as specified by Gen2.
10	Word count	Word count, range is 1…255, 0 causes error 5 (INVALID_PARAMETER).

9.7 Write: 32-bit addressing

When the EA2 flag in the common block's flag field is '0' then the write information block uses 32-bit addressing as follows:

Offset	Field	Is
0	Block size	Number of bytes to follow; size of this block excluding the size byte. Block's integrity is checked against this length byte.
1	Bank	Bank: range is 0…3.
2	Word address	DWORD, 32-bit word address as specified in Gen2.
6	Word count	Word count: range starts from 1 and is limited by the byte starting this block. Must not be zero (error 5, INVALID_PARAMETER).
7…n	Bytes	Words to write. Byte length must: • match the word count multiplied by two (error 5, INVALID_PARAMETER) • must not be zero (error 5, INVALID_PARAMETER) • size must match the block length subtracted with the size of the preceding parameters (error 2, INVALID_LENGTH)

9.8 Write: 64-bit addressing

When the EA2 flag in the common block's flag field is '1' then the write information block uses 64-bit addressing as follows:

Offset	Field	Is
0	Block size	Number of bytes to follow; size of this block excluding the size byte. Block's integrity is checked against this length byte.
1	Bank	Bank: range is 0…3 (error 5, INVALID_PARAMETER).
2	Word address	QWORD, 64-bit word address as specified in Gen2.
10	Word count	Word count: range starts from 1 and is limited by the byte starting this block. Must not be zero (error 5, INVALID_PARAMETER).
11…n	Bytes	Words to write. Byte length must: • match the word count multiplied by two (error 5, INVALID_PARAMETER) • must not be zero (error 5, INVALID_PARAMETER) • size must match the block length subtracted with the size of the preceding parameters (error 2, INVALID_LENGTH)

9.9 Block write (L2-family)

The parameterized block write command works the same way as the normal write command except that it uses the Gen2 BlockWrite command. However, since there are currently very few tags that support the Gen2 BlockWrite command exactly as it is specified, it has been seen useful to have a specifically parameterized BlockWrite command in order to tell the module how to perform the command. Block write is distinguished from a normal write command with its command value that is **0x42**.

9.10 BlockWrite information block, 32-bit addressing

When the EA2 flag in the common block's flag field is '0' then the block write information block uses 32-bit addressing as follows:

Offset	Field	Is
0	Command	0x42 (66).
1	Block size	Number of bytes to follow; size of this block excluding the size byte. Block's integrity is checked against this length byte.
2	Bank	Bank: range is 0…3.
3	Word address	DWORD, 32-bit word address as specified in Gen2.
7	Word count	Word count: range starts from 1 and is limited by the byte starting this block. Must not be zero (error 5, INVALID_PARAMETER).
8	Block length	Defines the block size in 16-bit words. If this value is 0 then the block size is set equal to word count. Otherwise, if the word count is not divisible by the block size an error occurs (5, INVALID_PARAMETER). If the block size is more than word count, then the used block size is also set equal to word count.
9…n	Bytes	Words to write. Byte length must: • match the word count multiplied by two (error 5, INVALID_PARAMETER) • must not be zero (error 5, INVALID_PARAMETER) • size must match the block length subtracted with the size of the preceding parameters (error 2, INVALID_LENGTH)

9.11 BlockWrite information block, 64-bit addressing

When the EA2 flag in the common block's flag field is '0' then the block write information block uses 32-bit addressing as follows:

Offset	Field	Is
0	Command	0x42 (66)
1	Block size	Number of bytes to follow; size of this block excluding the size byte. Block's integrity is checked against this length byte.
2	Bank	Bank: range is 0…3.
3	Word address	QWORD, 64-bit word address as specified in Gen2.
11	Word count	Word count: range starts from 1 and is limited by the byte starting this block. Must not be zero (error 5, INVALID_PARAMETER).
12	Block length	Defines the block size in 16-bit words. If this value is 0 then the block size is set equal to word count. Otherwise, if the word count is not divisible by the block size an error occurs (5, INVALID_PARAMETER). If the block size is more than word count, then the used block size is also set equal to word count.
13…n	Bytes	Words to write. Byte length must: • match the word count multiplied by two (error 5, INVALID_PARAMETER) • must not be zero (error 5, INVALID_PARAMETER) • size must match the block length subtracted with the size of the preceding parameters (error 2, INVALID_LENGTH)

9.12 Read BlockPermalock information block (L2-family)

Offset	Field	Is
0	Lock command	Set to 0 (read permalock).
1	Bank	1...3 (in practice the bank is always 3, user memory).
2	Block address	DWORD, 0…0xFFFFFFFF. The address of 16-word block.
6	Range.	Number of 16-word blocks to acquire the lock information from.

9.13 Set BlockPermalock information block (L2-family)

Offset	Field	Is
0	Lock command	Set to 1 (=set permalock).
1	Bank	1...3 (in practice the bank is always 3, user memory).
2	Block address	DWORD, 0…0xFFFFFFFF. The address of 16-word block.
6	Range.	Number of 16-word blocks to acquire the lock information from.
7…n	Lock words	Lock words as specified in the Gen2 protocol specification. Each bit corresponds to a 16-word block. Number of words must match the *range* parameter.

9.14 Block erase (L2-family)

The block erase command's value is **0x40**. Otherwise its structure is similar to the read command. The same kind of structure holds the following information.

9.15 Block erase information block, 32-bit addressing

Offset	Field	Is
0	Block size	Number of bytes to follow.
1	Bank	Bank: range is 0…3.
2	Word address	DWORD, 32-bit word address as specified by Gen2.
6	Word count	Word count (=number of 16-bit words to erase), range is 1…255, 0 causes error 5 (INVALID_PARAMETER).

9.16 Block erase information block, 64-bit addressing

Offset	Field	Is
0	Block size	Number of bytes to follow.
1	Bank	Bank: range is 0…3.
2	Word address	QWORD, 64-bit word address as specified by Gen2.
10	Word count	Word count (=number of 16-bit words to erase), range is 1…255, 0 causes error 5 (INVALID_PARAMETER).

9.17 Lock information block

With lock command there is no addressing information and the EA2 flag in the flag field is ignored.

Offset	Field	
0	Block size	Always 4.
1	Mask	WORD, mask as specified in the Gen2.
3…5	Action	WORD, action as specified in the Gen2.

9.18 Kill information block

Kill information block's presence is currently ignored.

9.19 Read command structure

Offset	Field	
0	Read tag	Value is 0x33 (51).
Common block		
1	Flags	The flag field.
2	Password	DWORD, 32-bit password value.
Singulation block		
6	Singulation data	N bytes, size depends on the flag field. Optional.
Read information block		
6+N…M-1	Address, count.	M bytes, size depends on the flag field. Mandatory.

9.20 Successful read response

Offset	Field	
0	Command echo.	Value is 0x33.
1	Status	O (OK).
2…N+2-1	Bytes	Data read from the tag. Note: if for some reason the tag responded with a data shorter than expected then no error is indicated. Therefore the host should always check the received data length.

9.21 Read error response

Offset	Field	
0	Command echo	Value is 0x33.
1	Status / error	0x22 (G2_ERROR_SELECT): error during singulation. 0x24 (G2_ERROR_ACCESS): accessing with given password failed. 0x42 (G2_TAG_ERROR_RESP): tag responded with an error 0x30 (G2_ERROR_READ): unspecified error. May be e.g. communication error due to interference, the tag left the field before finishing the read or the tag simply does not support error codes and did not answer.
2	Tag's error code.	Present if the error was 0x42. This byte value is what the tag backscattered to the reader.

9.22 Write / BlockWrite command structure

Offset	Field	
0	Write tag	Value is 0x34 (52) for write command. Value is 0x35 (53) for BlockWrite command. Value is 0x42 (66) for BlockWrite command appended with block size definition.
Common block		
1	Flags	The flag field.
2	Password	DWORD, 32-bit password value.
Singulation block		
6	Singulation data	N bytes, size depends on the flag field. Optional.
Write information block		
6+N…M-1	Address, count, data.	M bytes, size depends on the flag field. Mandatory.

9.23 Successful write response

Offset	Field	
0	Command echo.	Value is 0x34 or 0x35 (BlockWrite).
1	Status	O (OK).
2	Count.	Number of words written successfully.

9.24 Write error response

Offset	Field	
0	Command echo.	Value is 0x34 or 0x35 (BlockWrite).
1	Status / error	0x22 (G2_ERROR_SELECT): error during singulation. 0x24 (G2_ERROR_ACCESS): accessing with given password failed. 0x42 (G2_TAG_ERROR_RESP): tag responded with an error 0x40 (G2_ERROR_WRITE): unspecified error. May be e.g. communication error due to interference, the tag left the field before finishing the read or the tag simply does not support error codes and did not answer. 0x41 (G2_ERROR_WR_PART): Partially successful write without an indication what went wrong.
If error is 0x42 (G2_TAG_ERROR_RESP)		
2	Tag's error.	The error code the tag backscattered.
3	Count.	Number of words written until error.
If error is 0x41 (G2_ERROR_WR_PART)		
2	Count	Number of words written until error.

9.25 Lock command structure

Offset	Field	
0	Lock tag	Value is 0x36 (54).
Common block		
1	Flags	The flag field. SEC bit is ignored; lock is always secured.
2	Password	DWORD, 32-bit password value.
Singulation block		
6	Singulation data	N bytes, size depends on the flag field. Optional.
Lock information block		
6+N	Block size.	Value is 4.
6+N+1	Mask	2 WORD parameters as specified in Gen2. Only the lowest 10 bits are used producing the 20-bit lock parameter.
6+N+3	Action	

9.26 Lock response

Offset	Field	
0	Command echo.	Value is 0x36 (54).
1	Status / error	0 (OK) 0x22 (G2_ERROR_SELECT): error during singulation. 0x24 (G2_ERROR_ACCESS): accessing with given password failed. 0x42 (G2_TAG_ERROR_RESP): tag responded with an error 0x40 (G2_ERROR_WRITE): unspecified error. May be e.g. communication error due to interference, the tag left the field before finishing the read or the tag simply does not support error codes and did not answer.
2	Tag's error	If error is 0x42 (G2_TAG_ERROR_RESP) this is the tag's backscattered error code otherwise this byte is not present.

9.27 Kill command structure

Offset	Field	
0	Kill tag	Value is 0x37 (55).
Common block		
1	Flags	The flag field. SEC bit is ignored; kill is always secured.
2	Password	DWORD, 32-bit password value. This is the kill password value.
Singulation block		
6	Singulation data	N bytes, size depends on the flag field. Optional.
Kill information block		Currently not used.

9.28 Kill response

Offset	Field	
0	Command echo.	Value is 0x37 (55).
1	Status / error	0 (OK) 0x22 (G2_ERROR_SELECT): error during singulation. 0x24 (G2_ERROR_ACCESS): accessing with given password failed. 0x42 (G2_TAG_ERROR_RESP): tag responded with an error 0x40 (G2_ERROR_WRITE): unspecified error. May be e.g. communication error due to interference, the tag left the field before finishing the read or the tag simply does not support error codes and did not answer.
2	Tag's error	If error is 0x42 (G2_TAG_ERROR_RESP) this is the tag's backscattered error code otherwise this byte is not present.

9.29 Block erase command structure

Offset	Field	
0	Block erase	Value is 0x40 (64).
Common block		
1	Flags	The flag field.
2	Password	DWORD, 32-bit password value.
Singulation block		
6	Singulation data	N bytes, size depends on the flag field. Optional.
Read information (block erase) block		
6+N…M-1	Address, count.	M bytes, size depends on the flag field. Mandatory. This command uses the read information block. The number of words parameter number of block to erase.

9.30 Block erase response (L2-family)

Offset	Field	
0	Command echo.	Value is 0x40 (64).
1	Status / error	0 (OK) 0x22 (G2_ERROR_SELECT): error during singulation. 0x24 (G2_ERROR_ACCESS): accessing with given password failed. 0x42 (G2_TAG_ERROR_RESP): tag responded with an error 0x40 (G2_ERROR_WRITE): unspecified error. May be e.g. communication error due to interference, the tag left the field before finishing the read or the tag simply does not support error codes and did not answer.
2	Number of words locked OR tag's error code.	If error is 0x42 (G2_TAG_ERROR_RESP) this is the tag's backscattered error code otherwise this byte is not present. If the status is OK, this byte echoes the number of words that were locked. In all other cases this byte is 0.

9.31 Read block PermaLock command structure (L2-family)

Offset	Field	
0	Tag permalock read / set	Value is 0x44 (68)
Common block		
1	Flags	The flag field.
2	Password	DWORD, 32-bit password value.
Singulation block		
6	Singulation data	N bytes, size depends on the flag field. Optional.
Block PermaLock block		The lock parameter is 0 so this is read.

9.32 Read block PermaLock response (L2-family)

Offset	Field	
0	Commend echo	Value is 0x44 (68)
1	Status	0 when successful.
2	Bank	The bank where the lock status was read from.
3	Address.	DWORD. 16-word block address where the reading started.
7	Number of words	Number of lock status words to follow.
8…	Lock status words	Lock status words as specified in the Gen2 specification. **NOTE**: the words are as they are received from the tag i.e. in big-endian format.

9.33 Set block PermaLock command structure (L2-family)

Offset	Field	
0	Tag permalock read / set	Value is 0x44 (68)
Common block		
1	Flags	The flag field.
2	Password	DWORD, 32-bit password value.
Singulation block		
6	Singulation data	N bytes, size depends on the flag field. Optional.
Block PermaLock block		The lock parameter is 1 so this is permalock set operation.

9.34 Set block PermaLock response (L2-family)

When successful, this command will simply respond with a OK result i.e. command echo and 0 as status byte.

9.35 Block PermaLock (set / read) errors (L2-family)

Offset	Field	
0	Commend echo	Value is 0x44 (68)
1	Status	0x22 (G2_ERROR_SELECT): error during singulation. 0x24 (G2_ERROR_ACCESS): accessing with given password failed. 0x42 (G2_TAG_ERROR_RESP): tag responded with an error 0x30 (G2_ERROR_READ): unspecified error. May be e.g. communication error due to interference, the tag left the field before finishing the read or the tag simply does not support error codes and did not answer.
2		Tag's backscattered error code, if the status is 0x42. In other cases this byte is 0.

9.36 Tag tracing

This command causes a continuous trace of a specific tag. It is useful e.g. tracing a single item in a larger population of a tag. Since the trace notification when the tag is in the field also sends the tag's RSSI the command can be used as a "homing operation". It is also possible to instruct the command to only send the RSSI. Like tag singulation also this operation can be performed with 32-bit as well as with 64-bit addressing. The command takes singulation parameters in order to find tag that exactly matches the bit pattern in its specific bank.

The operation can be cancelled with the "Stop all continuous commands" command or specifying the stop flag in the flags field if it was run continuously. It is also possible to run the trace procedure only once by setting the continuous flag in the flags field to 0.

Offset	Field		
0	Trace tag	Value is 0x38 (56).	
1	Flags	Control flags	
		Flag bit	**Is**
		0x01 (bit 0)	No EPC. Causes the command to only send the RSSI and antenna fields.
		0x02 (bit 1)	Start continuous tracing. Set to 0 to run only once.
		0x04 (bit (2)	32/64-bit addressing selection
		0x08 (bit 3)	Stop continuous tracing
2	Bank	Range is 1…3.	
If flag bit 0 is '0': 32-bit addressing.			
3	Bit address.	DWORD, 32-bit selection bit address.	
7	Mask bit length	This is the bit length of the mask (BL). It is expected that this field is followed by a number of mask bytes that is in terms of C given by (using integers) *N = (BL / 8) + ((BL & 7) != 0).* See example.	
8…8+N-1	Mask data	Byte length must match as stated above.	
If flag bit 0 is '1': 64-bit addressing.			
3	Bit address.	QWORD, 64-bit selection bit address.	
11	Mask bit length	This is the bit length of the mask (BL). It is expected that this field is followed by a number of mask bytes that is in terms of C given by (using integers) *N = (BL / 8) + ((BL & 7) != 0).* See example.	
12…12+N-1	Mask data	Byte length must match as stated above.	

Trace tag response is a simple OK response if the flags specified the trace to be stopped.

10 Customizable commands

There are three customizable commands: custom read, custom write and custom exchange. The first two act as read and write commands with only the command value, bank and their corresponding bit length modifiable. The custom exchange command has several flags to be used and it also expects the transmitted bit buffer to be implemented by the host.

All of the commands can be pre-pended with the singulation block.

10.1 Custom read

Command value is 0x3C (60).

For the 32-bit addressing the custom read command appended after possible singulation block is:

Offset	Field	
0	Custom read command.	Value is 32-bit. The length field defines how many bits of the command DWORD is used.
4	Read command bit length.	Defines how many bits from the given command is used. Range is 1…32 and invalid value causes invalid parameter error.
5	Bank value for the command.	32-bit value to be used for the custom read command.
9	Bank bit length	Defines how many bits from the bank parameter are used. Range is 0…32. Value larger than 32 causes invalid parameter error.
10	Word address.	32-bit word address value.
14	Word count.	Number of words to read. Must be greater than 0.

For the 64-bit addressing the custom read command appended after possible singulation block is:

Offset	Field	
0	Custom read command.	Value is 32-bit. The length field defines how many bits of the command DWORD is used.
4	Read command bit length.	Defines how many bits from the given command is used. Range is 1…32 and invalid value causes invalid parameter error.
5	Bank value for the command.	32-bit value to be used for the custom read command.
9	Bank bit length	Defines how many bits from the bank parameter are used. Range is 0…32. Value larger than 32 causes invalid parameter error.
10	Word address.	64-bit word address value.
18	Word count.	Number of words to read. Must be greater than 0.

10.2 Custom write

Command value is 0x3D (61).

For the 32-bit addressing the custom write command appended after possible singulation block is:

Offset	Field	
0	Bytes to follow.	Number of bytes to follow including the control part and the data; excluding bytes to follow.
1	Custom write command.	Value is 32-bit. The length field defines how many bits of the command DWORD is used.
5	Write command bit length.	Defines how many bits from the given command is used. Range is 1…32 and invalid value causes invalid parameter error.
6	Bank value for the command.	32-bit value to be used for the custom read command.
10	Bank bit length	Defines how many bits from the bank parameter are used. Range is 0…32. Value larger than 32 causes invalid parameter error.
11	Word address.	32-bit word address value.
12	Word count.	Number of bytes in the following data bytes. Must be greater than 0 and divisible by 2.
16…	Data bytes.	Number of words to read. Must be greater than 0.

For the 64-bit addressing the custom write command appended after possible singulation block is:

Offset	Field	
0	Bytes to follow.	Number of bytes to follow including the control part and the data; excluding bytes to follow.
1	Custom write command.	Value is 32-bit. The length field defines how many bits of the command DWORD is used.
5	Write command bit length.	Defines how many bits from the given command is used. Range is 1…32 and invalid value causes invalid parameter error.
6	Bank value for the command.	32-bit value to be used for the custom read command.
10	Bank bit length	Defines how many bits from the bank parameter are used. Range is 0…32. Value larger than 32 causes invalid parameter error.
11	Word address.	64-bit word address value.
16	Word count.	Number of bytes in the following data bytes. Must be greater than 0 and divisible by 2.
17…	Data bytes.	Number of words to read. Must be greater than 0.

10.3 Custom exchange

Command value is 0x3E (62). For this command the host is expected to build the transmitted bit buffer by itself as well as to decode the response. Custom exchange block appended after possible singulation block is:

Offset	Field	Is
0	Control flags WORD.	Various bits that control the transmission and reception. See custom exchange control bits below.
2	TX length WORD.	Number of bits in the transmission buffer. Range is 1…1023.
4	RX length WORD.	Number of bits expected in the response. Can be 0 in case of transmission only or when the reception length is unknown
5	Reception timeout.	The reception timeout in milliseconds. Must always be present and the range is 20…100.
6…	Bit buffer.	The bit buffer to be transmitted. Length of this buffer (bufLen) in bytes is determined by TX length parameter (txLen) in terms of C like: bufLen = (txLen/8) + ((txLen%8)!=0) ;

10.4 Custom exchange control flags

Mask value	Name	Is
0x0001	asWrite	Perform the operation as if it was a write operation; this control the receiver when the response is read. NOTE: setting this flag will also ignore the response length; the module expects response that is like a write's response (length may not be known).
0x0002	useHandle	When set then the handle received from the tag's singulation is appended to the bit stream.
0x0004	xorRn16	When set then after the singulation, a ReqRN is performed and the resulting new RN16 is XORed with the last 16 bits in the given bit buffer. Causes invalid parameter error if the bit buffer is less than 16 bits.
0x0008	txOnly	Causes the module to only send the given bit buffer and no response is expected. CRC-16 is appended.
0x0010	noTxCRC	Causes the module to send without appending the CRC-16. Also in this case it is not possible to expect any answer.
0x0020	noRxCRC	Causes the receiver not to decode the received CRC. The response is then returned "as is".
0x0030	CRC5	Causes the module to use CRC-5 instead of CRC-16 when transmitting.
0x0080	noRxLen	The RX length parameter shall not be used. Useful when the expected reception length may vary.
0x0100	stripHandle	The last 2 bytes in a successful reception are considered to be the tag's handle and those bytes are left out from the response.
0x0200	No re-select	Causes this specific command not to do any tag selection (singulation) procedure (L2-family).

10.5 Custom exchange errors

As a complex command as the custom exchange is it can run into several errors. The list explains the possible ones:

Error	Can be caused by
0x22	Tag select error. • Singulation block error • Invalid response from the singulated / non-singulated tag • Tag not in range. • The "xorRn16"-flag is set, but the tag did not respond or gave erroneous response to the ReqRN command.
0x24	Tag access error. • Invalid password • Invalid response from the singulated / non-singulated tag • Tag not in range.
0x42	Tag responded with an error; caused by invalid command. NOTE: the tag's error response byte is returned as a response parameter.
0x05	Invalid parameter. • Bit buffer length mismatch • The "xorRn16"-flag is set but TX bit buffer is shorter than 16 bits • RX length is greater than 1023 bits. • Response is expected ("noRxLen" is 0) but RX length is set to 0 • RX timeout is out of range (20…100)
0x30	Generic error response to the command (Gen 2 read error). Indicates air IF error or no response.
0x40	Generic error response in a situation where the "txOnly"-flag was set but the transmission ended unexpectedly.

11 Inventory stream

The inventory stream tells the module to continuously do an inventory and after each inventory send the results to the host with the inventory notification.

The inventory command structure is:

Case 1: no parameters (length=0)

Offset	Field	
0	Inventory stream	Value is 0x39 (57).

This format stops the ongoing inventory stream.

Case 2: use default settings (length=1

Offset	Field	
0	Inventory stream	Value is 0x39 (57).
1	Ignored.	-

With this command format the inventory stream takes its parameters from the current default settings i.e. what was set using the Load setup command's rounds, session and Q fields.

Case 3: set parameters (length=3)

Offset	Field	
0	Inventory stream	Value is 0x39 (57).
1	Q	0…15
2	Session	0…3
3	Rounds	1…10.

(Inventory select continued)

Case 4: add select parameters

In this case the inventory select parameters are added to the request:

Offset	Field	Note
0	Inventory stream	Value is 0x39 (57).
1	Q	0…15
2	Session	0…3
3	Rounds	1…10.
4	Selection block's size.	Number of bytes to follow.
5	Select bank.	Bank that the selection mask is applied to. Range is 1…3. Error causes the bank error flag to be set (bit 2).
6	Flags.	Option bits: 0x01 (bit 0): the selection is done as inverted i.e. the tags that do not match the selection criteria will participate in this inventory round. 0x02 (bit 1): if 1 then use extended addressing. This is to say that the following mask bit address is 64-bit (QWORD) instead of 32 bits (DWORD).
Continues with other select parameters…		

11.1 Inventory stream response / notification

Inventory stream notification has the same contents as the get ID buffer added with this pre-pended header:

Offset	Field	
0	Stopped?	BYTE, 1 if stream was stopped.
1	Rounds done	BYTE, rounds done during last inventory.
2…3	Collisions	WORD, Count of events that the reader interpreted as collisions.
4	Last Q	BYTE, last used Q value.
Continues with get ID buffer with meta response…		

12 Extended inventory

Extended inventory's command value is 0x3A. It is a continuous inventory or once run operation that can be stopped by giving the command without any parameters. The idea in extended inventory is that the inventory can take in specific parameters for the either continuously or once run inventory operation. Parameters include Q, session, number of inventory rounds, channel transition time, target, select state and one or more filters that are able to return a number of different tag populations that meet the requirements that the filter or filters specify.

Command structure when started or run once is:

Offset	Field	Is
0	Flags	BYTE. Bit 0 is used. When set to '1' the operation is run continuously. When the bit 0 is '0' then the operation is run once and then stopped.
1	Q	BYTE. Q parameter for extended inventory 0…15.
2	Session	BYTE. Session parameter for extended inventory 0…3.
3	Rounds	BYTE. Number of inventory rounds.
4…5	Transition time	WORD. Channel change time in milliseconds. Set to 0 to disable and use current region's default channel time.
6	Target	BYTE. Target value; A, B, or AB = 0, 1, and 2 respectively.
7	Query select	BYTE. This is the parameter that the query command uses: all (0, 1), SL (2) and ~SL (3).
8	Filter count	BYTE. Number of filters to follow. Range is 0…8. Too many causes invalid parameter error.
Filter example		If present; the filter tables
9	Truncate	BYTE. Truncate bit for the select command, 0/1. **NOTE: not supported in the current FW version.**
10	Target	BYTE. 3-bit target parameter for select (inventoried S0…S3) or SL.
11	Action	BYTE. 3-bit action parameter for select (specified in EPCGlobal's specification).
12	Bank	BYTE. Bank into which the mask is applied to; range is 1…3).
13….16	Mask bit address	DWORD value for the select mask's bit address.
17	Mask length	BYTE. This tells the bit length of the select command's mask.
18…n	Mask bytes	The actual mask bytes
Mask example		Example: bit length (see above) is 20 → 3 bytes (24 bits in total) are required.
18	0xAA	Mask bytes that fill the mask length requirement; mask = 0xAABBC.
19	0xBB	
20	0xC0	

(Extended inventory continues...)

Parameter errors that cause the invalid parameter error followed by the error flag byte:

Bit mask	Bit	Caused by
0x01	0	Q greater than 15.
0x02	1	Session greater than 3.
0x04	2	Bank out of range: [1…3].
0x08	3	Parameter length error: - Not enough parameters (command too short) - filter data length mismatch - too many filters (more than 8)

12.1 Extended inventory response

The extended inventory starts notifying the host in the same manner as continuous inventory when run continuously. When the operation is run once the operation's response is similar to the inventory that is run once.

13 Simple EPC enumeration stream (L2-family)

It is possible to enumerate tags all that all have the same EPC. The enumeration is simple done by copying a part of the TID memory to the EPC. The module itself keeps track of the last 64 TID contents that have been written. Upon each successful EPC write the module notifies about the last result.

Note that interrupting the EPC enumeration and starting inventory stream right after resets the current EPC enumeration setup. Also trying to start the EPC enumeration while continuous inventory is running will cause an error response 13 (0x0D, NOT_READY).

13.1 Starting the EPC enumeration function

The command format is:

Offset	Field	Is
0	Command	Value is 0x43 (67).
1	Antenna	The antenna that should perform the enumeration: 0…3.
2	TID source address	Word address where the copied TID data is read from.
3	TID source length	Number of words to read from the TID bank. Range is 1…4 (2…8 bytes).
4	Use of block write	Valid values are: • 0 none: the words are written with a standard write command • 1 one word: the words are written one by one but using the BlockWrite Command. • 2 two words: the EPC data is written by using BlockWrite command writing 2 words (32 bits) at a time
5…12	Running number's start value	This is 64-bit unsigned integer. This value is taken as a start value for the enumeration. It is incremented after each written EPC.
13	EPC length	The EPC words length. Range is 4…8 (8…16 bytes).
14	Modification address	This is the bit address where the modified bit pattern is placed in the EPC memory. **IMPORTANT**: this address is within the EPC data i.e. 32 is added to this by the module.
15	Number's bit length	This is the number of bits that is used in the running number's value. Range is 1…64.
16	Reset	If zero, then possibly earlier saved TID data is saved. Otherwise TIDs are cleared. **NOTE**: a reset occurs and stored TIDs are lost when a streaming inventory has taken place after EPC enumeration has been stopped.
17…32	Base EPC	16 bytes that contains the "base EPC". This field contains the base value that is written to the EPC memory and modified according to the parameters related to TID and running number.

13.2 Errors when starting an EPC enumeration

When the start is successful the module replies with a simple OK response.

In case of parameter error or errors then module replies with INVALID_PARMAETER (0x05) error. The error reply is appended with an error flag byte. The response's flag field can contain several error flags and the content is:

Offset	Field	Is	
0	Command echo	Value is 0x43 (67).	
1	Error flags	**Bit / mask**	**Error is**
		0 / 0x01	The given antenna is not available or enabled.
		1 / 0x02	BlockWrite parameter is greater than 2.
		2 / 0x04	TID word length is out of range [1…4].
		3 / 0x08	EPC length is out of range [4…8].
		4 / 0x10	Running number's bit length is out of range [1…64].
		5 / 0x20	Modified bit address is out of
		6 / 0x40	The modified bit address added with running number's bit length is larger than the EPC bit length.

Note about the start: starting this continuous function will interrupt all other continuous functions except inventory stream.

13.3 Stopping the EPC enumeration

The EPC enumeration is stopped with the same command structure by setting all the fields to 0xFF:

Offset	Field	Is
0	Command	Value is 0x43 (67).
1…32	0xFF	Stop indication.

13.4 EPC enumeration stream notification (L2-family)

This notification occurs after a successful start of EPC enumeration stream. The notification content is:

Offset	Field	Note
0	Notification code.	0x8A (new EPC was written)
1	EPC length	BYTE. 2…16. Length of the last written EPC.
2	TID length	BYTE. Length of the written TID data.
3…18	EPC	Last written EPC. Number of valid bytes is stated by <*EPC length*> .
19…34	TID	Last TID data used in write. Number of valid bytes is stated by <*TID length*> .

14 Special commands

14.1 Read / write scratch area

From FW version 5.0-A (3.8-A in NUR05W) onwards the module provides 512-byte non-volatile scratch memory area divided into two 256-byte pages. This area can freely be used for any type of user information and it is addressable per byte basis like in EEPROM memory.

14.2 Read scratch command

Byte	Field	Note
0	Command code	0x79 Read/write scratch area.
1	Page	Page is either 0 or 1.
2	Byte address	Byte address is in range 0...255
3..4	Byte count (WORD)	Number of bytes to read. Care must be taken that (byte address + byte count -1) <= 255

14.3 Scratch read response

Byte	Field	Note
0	Command code	0x79
1	Status	0 on success. Error 0x05 (INVALID_PARAMETER) occurs with parameter range exceeding or mismatches (invalid page, address overrun).
2...2+count - 1	Byte address	Read data.

Error 0x05 (INVALID_PARAMETER) occurs with parameter range exceeding or mismatches (invalid page, address overrun).

14.4 Write scratch command structure

Byte	Field	Note
0	Command code	0x79 Read/write scratch area.
1	Page	Page is either 0 or 1.
2	Byte address	Byte address is in range 0...255.
3...4	Byte count (WORD)	Number of bytes to read. Care must be taken that (byte address + byte count - 1) <= 255.
5...5 + count - 1	Data to write.	Care must be taken that the data length matches to byte count.

14.5 Scratch write response

Byte	Field	Note
0	Command code	0x79
1	Status	0 on success. Error 0x05 (INVALID_PARAMETER) occurs with parameter range exceeding or mismatches (invalid page, address overrun, data length mismatch). Error 0x0D (13, NOT_READY) when the write failed internally.

14.6 Antenna mapping

From FW version 5.0-A (L2 family) and version 3.8-A (NUR05W) the module also provides information about antenna mapping. This means that the module internally stores names for the antennas.

An antenna map consists of the physical antenna ID (index number / logical antenna) that is assigned to an antenna name. In a module that is configured as STIX reader for example there is only one antenna present having the name “Internal”.

In terms of the protocol, a single antenna map entry looks like (STIX example):

ID	Length	ASCII characters							
0	8	I	n	t	e	r	n	a	l

14.7 Reading the antenna map

The command code for antenna mapping is 025. The command does not take any parameters.

Byte	Field	Note
0	Command code	0x25 (antenna map)

The error 0x01 (INVALID_COMMAND) is sent when the FW does not support this operation.

Example: 4-port reader response to antenna mapping

A NUR module with four antenna ports responds to the command like this:

Byte(s)	Field	Note
0	Command echo	0x25 (antenna map)
1	Status	0
2	Number of entries to follow.	4 (depends on the module configuration).
3	Logical number 1	0
4	Name 1 length	4
5...8	Name 1	“AUX0”
9	Logical number 2	1
10	Name 2 length	4
11...14	Name 2	“AUX1”
15	Logical number 3	2
16	Name 3 length	4
17...20	Name 3	“AUX2”
21	Logical number 4	3
22	Name 3 length	4
23...26	Name 4	“AUX3”

14.8 Channel scanning

The channel scanning provides a means to “listen” to either a single channels or scan through the whole currently selected region’s channels. To put it simply, it provides a frequency spectrum over the currently selected region’s channels.

The scan channel either takes one byte as a parameter defining the channel number (0…max – 1) or no parameters. When the parameter length is 0, the module scans through all available channels in the currently selected region and outputs multiple results as show below.

The command is executed using the currently selected antenna thus if information from all antennas is required the antenna then needs to be changed before each scan.

14.9 Scan channel command

Offset	Field	Note
0	Command code	0x63 (scan channel(s))
1	Channel	Optional. If currently selected region’s channel count is denoted “MAX” then this parameter needs to be in range 0…MAX-1.

14.10 Scan channel response (per channel)

Offset	Field	Note
0	Command echo	0x63 (scan channel(s))
1	Status	0 when no error (Older FW version may indicate 0x01 (INVALID _COMMAND) error. Error 0x02 (INVALID_LENGTH) is returned if the parameter length is other than 0 or 1. No information is present with either of these errors.
2	Frequency	DWORD. Frequency in kHz of this result.
6	Level.	32-bit signed integer indicating level in dBm.
10	Raw I/Q.	The I/Q values from the receiver. Ignore, or contact Nordic ID R&D via support if information considered necessary.

The above response can be repeated several times if the scan channel was executed with no parameters. For example in EU region the response has 4 entries and in FCC region there are 52 entries.

15 Proprietary commands

15.1 NXP read protect

The NXP read protect command value is 0x50 (80).

The NXP read protect command consists of the same common and optional singulation blocks as e.g. read or write command. In addition there is the NXP block that defines:

Offset	Field	Is
0	Bytes to follow	Always 1.
1	Set / reset	1 = set read protect, 0 = reset read protect.

15.2 NXP EAS

The NXP EAS set / reset command value is 0x51 (81).

The NXP EAS command consists of the same common and optional singulation blocks as e.g. read or write command. In addition there is the NXP block that defines:

Offset	Field	Is
0	Bytes to follow	Always 1.
1	Set / reset	1 = set EAS, 0 = reset EAS.

15.3 Password in the NXP read protect and EAS set/get commands

The used password is placed in to the common block's password field for all NXP commands.

15.4 Get NXP EAS alarm

The get NXP EAS alarm value is 0x52 (82).

The NXP EAS command consists of the same common and optional singulation blocks as e.g. read or write command. The command takes no parameters. The response is:

Offset	Field	Is
0	Command echo	0x52 (82).
1	Status	0 when an alarmed tag was in the field. ERROR_NO_TAG (0x20) when the alarm was not detected or it was corrupt.

15.5 Start or stop NXP EAS alarm stream

It is also possible to run the EAS alarm detection continuously. Then the response is sent through the EAS alarm notification.

Offset	Field	Is
0	Start / stop NXP EAS alarm stream.	0x54 (84).
1	Start / stop.	0 = stop 1 = start

This command can cause INVALID_LENGTH (0x02) error if the parameter length is other than 1.

The response to start or stop is simple OK response with no parameters.

15.6 NXP EAS alarm stream notification

This notification is sent whenever the NXP EAS alarm stream detects a tag in the field that indicates the EAS alarm bit being set.

Offset	Field	Note
0	Notification code.	0x89 (NXP EAS alarm)
1	Status.	This byte consists of flags: Bit 0 (0x01) is set if the alarm was detected in the last round Bit 1 (0x02) is set if the alarm stream was stopped

15.7 Monza 4 QT command

The Monza 4 QT command value is 0x53.

NOTE: Monza 4 write target is always non-volatile memory.

The Monza 4 QT command consists of the same common and optional singulation blocks as e.g. read or write command. In addition there is the Monza 4 block that defines:

Offset	Field	
0	Bytes to follow	Always 1 (read) or 3 (write).
1	Reserved.	Set to 0.
2…3	QT command bits.	(Write) WORD parameter As specified in the Monza 4 QT command specification: Bit 15 = QT_SR (1=reduce) Bit 14 = QT_MEM (1=public profile).

15.8 Password usage in Monza 4 command

The used password is placed in to the common block's password field.

16 Set custom hop table

With this command a custom hoptable can be stored into the module's volatile memory. The command structure is:

Offset	Field	
0	Command	Value is 0x29 (41).
1	Channel count	DWORD. Range is 1…100.
5	Channel time	DWORD, time before hopping to next frequency in ms. Minimum is 100ms.
9	Silent time	DWORD, time to pause when the channel is changed in ms. Maximum is 1000ms.
13	LF	DWORD, maximum LF setting. Valid values are 160000, 256000 and 320000.
17	Tari	DWORD, 1 = 12.5µs and 2 = 25µs.
21…24	Frequency entry[0]	Frequency values are DWORDs, unit is kHz and the range is 840000…960000kHz.
25…n	*<frequency entries>*	…

Rules:

- The hoptable is applied immediately if valid and the module's setup is set to use custom hoptable
- If the module's setup is not set to use custom hoptable then the (valid) hoptable is only stored
- Channel count is in range 1…100
- Channel time has a minimum setting of 100ms
- Silent time has a maximum value of 1000ms
- LF is restricted as in other cases (160k, 256k and320k)
- Tari setting must be either 1 (12.5µs) or 2 (25µs)
- Each given frequency (in kHz) has to be divisible by 25
- Frequency range is 840000…960000kHz
- The frequencies are hopped in the order they are sent to the module
- Any occurring error prevents the module from storing the given hoptable.

Command length: as the parameters above require 20 bytes then the command length is expected to be 20 + 4 * channel count. If the parameter length is 0 then the currently stored hoptable is echoed back, see "Custom hoptable response".

NOTE: this command is replaced with the extended version of the command and the response, when setting, is "Custom hoptable extended response".

16.1 Custom hoptable errors

The error response consists of the protocol specific error code (INVALID_PARAMETER, 0x05) and flag field (one byte) giving specific error information.

Offset	Field	Error bits	
0	Error flags.	**Bit**	**Meaning**
		0 (1)	Channel count is either 0 or > 100.
		1 (2)	Channel time is < 100 ms.
		2 (4)	Silent time is > 1000ms.
		3 (8)	Link frequency is other than 160k, 256k or 320k.
		4 (0x10)	Tari setting is other than 1 or 2.
		5 (0x20)	Length mismatch; the following frequency table has invalid length.
		6 (0x40)	The module encountered an invalid frequency and stopped parsing: either the frequency was out of range or it was not divisible by 25 (kHz).

16.2 Custom hoptable response

When no custom hoptable is currently stored the module responses with a DWORD values set to 0.

When a custom hoptable is stored then the response is:

Offset	Field	
0	Channel count	DWORD. Range is 1…100.
4	Channel time	DWORD, time before hopping to next frequency in ms. Minimum is 100ms.
8	Silent time	DWORD, time to pause when then channel is changed in ms. Maximum is 1000ms.
12	LF	DWORD, maximum LF setting. Valid values are 160000, 256000 and 320000.
16	Tari	DWORD, 1 = 12.5µs and 2 = 25µs.
20…23	Frequency entry[0]	Frequency values are DWORDs, unit is kHz and the range is 840000…960000kHz.
24…n	*<frequency entries>*	…

16.3 Set custom hop table extended

With this command a custom hoptable can be stored into the module's volatile memory. This command includes two additional parameters: the maximum TX level and LBT threshold. The command structure is:

Offset	Field	
0	Command	Value is 0x2A (42).
1	Channel count	DWORD. Range is 1…100.
5	Channel time	DWORD, time before hopping to next frequency in ms. Minimum is 100ms.
9	Silent time	DWORD, time to pause when the channel is changed in ms. Maximum is 1000ms.
13	LF	DWORD, maximum LF setting. Valid values are 160000, 256000 and 320000.
17	Tari	DWORD, 1 = 12.5µs and 2 = 25µs.
21…24	LBT threshold	32-bit signed integer. Minimum value is -90. Zero value means "not used".
25…28	Maximum TX level.	DWORD. Range is 0…19. Value represents 27 - <*tx_level*> dBm.
29…32	Frequency entry[0]	Frequency values are DWORDs, unit is kHz and the range is 840000…960000kHz.
25…n	<*frequency entries*>	…

Rules:

- The hoptable is applied immediately if valid and the module's setup is set to use custom hoptable
- If the module's setup is not set to use custom hoptable then the (valid) hoptable is only stored
- Channel count is in range 1…100
- Channel time has a minimum setting of 100ms
- Silent time has a maximum value of 1000ms
- LF is restricted as in other cases (160k, 256k and320k)
- Tari setting must be either 1 (12.5µs) or 2 (25µs)
- LBT threshold minimum value is -90. Zero means that it is not used.
- Maximum TX level is in range 0…19. It gives dBm value 27 - <*tx_level*>.
- Each given frequency (in kHz) has to be divisible by 25
- Frequency range is 840000…960000kHz
- The frequencies are hopped in the order they are sent to the module
- Any occurring error prevents the module from storing the given hoptable.

Command length: as the parameters above require 28 bytes then the command length is expected to be 28 + 4 * channel count. If the parameter length is 0 then the currently stored hoptable is echoed back, see "Custom hoptable extended response".

16.4 Custom hoptable extended response

When no custom hoptable is currently stored the module responses with a DWORD values set to 0.

When a custom hoptable is stored then the response is (this response shall replace the response shown in "Custom hoptable response"):

Offset	Field	
0	Channel count	DWORD. Range is 1…100.
4	Channel time	DWORD, time before hopping to next frequency in ms. Minimum is 100ms.
8	Silent time	DWORD, time to pause when then channel is changed in ms. Maximum is 1000ms.
12	LF	DWORD, maximum LF setting. Valid values are 160000, 256000 and 320000.
16	Tari	DWORD, 1 = 12.5µs and 2 = 25µs.
20	LBT threshold	32-bit signed integer.
24	Maximum TX level.	DWORD. Range is 0…19 representing 27 - <*tx_level*> dBm.
28…31	Frequency entry[0]	Frequency values are DWORDs, unit is kHz and the range is 840000…960000kHz.
32…n	<*frequency entries*>	…

16.5 Extended custom hoptable errors

The error response consists of the protocol specific error code (INVALID_PARAMETER, 0x05) and flag field (one byte) giving specific error information.

Offset	Field	Error bits	
0	Error flags.	**Bit**	**Meaning**
		0 (1)	Channel count is either 0 or > 100.
		1 (2)	Channel time is < 100 ms.
		2 (4)	Silent time is > 1000ms.
		3 (8)	Link frequency is other than 160k, 256k or 320k.
		4 (0x10)	Tari setting is other than 1 or 2.
		5 (0x20)	Length mismatch; the following frequency table has invalid length.
		6 (0x40)	The module encountered an invalid frequency and stopped parsing: either the frequency was out of range or it was not divisible by 25 (kHz).
		7 (0x80)	Either the LBT threshold is too low (< -90) or maximum TX level is out of range (0…19).

17 CRC-16 calculation

This is the C-implementation of the CRC-16:

```
#define CRC16_START       0xFFFF
#define CRC16_POLYNOMIAL  0x1021 /* CCITT */

static WORD crc16table[256];
static BOOL crc16Init = FALSE;

static void CRC16Init()
{
  int i, j;
  ULONG c;
  for (i = 0; i < 256; i++) {
    c = i << 8;
    for ( j = 0; j < 8; j++ ) {
      c = (c & 0x8000) ? CRC16_POLYNOMIAL ^ (c << 1) : (c << 1);
    }
    crc16table[i] = (WORD)c;
  }
}

WORD CRC16(WORD crc, const BYTE *buf, DWORD len)
{
  if (!crc16Init) {
    crc16Init = TRUE;
    CRC16Init();
  }
  while (len--) {
    crc = (crc << 8) ^ crc16table[(crc >> 8) ^ *buf++];
  }
  return crc;
}
```

18 Summary of command error codes

The list of typically needed error codes. In case of other error codes (not Gen2 protocol / operation codes) contact Nordic ID's support.

Code	Is
0	No error.
1	Invalid command: command or its format is not recognized.
2	Invalid length: typically command is expected to have certain length and to condition was not satisfied.
3	Parameter out of range: one or more command's parameters are out of range.
5	Invalid parameter: for example invalid setup flags.
11	CRC check error: communication error that is caused by either an incorrectly received or built CRC.
14	Application not present: the module was commanded to boot in the bootloader mode but no valid application was found.

19 Other documentation

There are application notes available either from the support@nordicid.com or within the distribution package in which this document is included in. The application notes illustrate, byte by byte basic, how the various command packets are formed.

AN	Contains
001	Basic commands including: ping, get mode, get reader information and get device capabilities.
002	Module setup including: get all setup parameters, set basic setup (TX level, RX decoding, link frequency, TX modulation, antenna enable/disable, antenna selection), setting default inventory parameters, storing setup to module and resetting to factory defaults.

There are some differences in the new NUR2 family regarding supported and not supported features. The details can be found in

NUR2_MigrationDoc_vXX.pdf and
NUR2-0W1_MigrationDoc_vXX.pdf

both located in NordicID/sdk/docs.

20 Version information

Revision	Date	Notes
14	December 5th 2019	Added reference to NUR2 migration documents
13	October 18th 2019	Added missing number in GET GPIO response description
12	October 18th 2019	Cosmetic changes and minor corrections
11	August 21st 2015 – PRE-RELEASE 1	Updated / added: - baudrates - operation flags in module setup - new module setup members: - extended antenna mask - autotune setup - extended per antenna power settings - receiver sensitivity - device capabilities - scratch area command - antenna mapping - channel scanning - reflected power reading - single tag fetching
10	-	-
9	October 7th 2013	Added: - extended hoptable - get FW info - get versions - new fields in module setup - whole module setup description API DLL version 1.6.0 (.NET 1.6.0.0), NUR/XNUR FW version 2.4 and above, NUR L2 FW version 4.0 and above.
8		Fixed reader information response. Fixed single tag scan response. API DLL version 1.4.3, FW 2.9 and on.
7	November 27th 2012	Added region numbers for Russia, Vietnam, Singapore, Philippines and Thailand. No changes to API DLL.
6	August 28th 2012	Custom exchange. API DLL applied is 1.4.1.
5	March 15th 2012	Additions: - Read, write and inventory RSSI filtering - Extended inventory - Japanese hop tables - API DLL version applied is 1.3.2
4	January 31st 2012	- added command typical command (not Gen2 related) error code list - fixed load setup description - API DLL version applied is 1.2.3.
3	November 11th 2011	API DLL version 1.2.0.

		Module SW version 2.1-A.
2	September 30th 2011	First “official”. Matches to the API DLL version 1.0.2.0.