NUR APPLICATION NOTE 04 (NUR AN004)

GETTING STARTED: SIMPLE INVENTORY AND TAG BUFFER FETCHING

Nordic ID Oy | Myllyojankatu 2 A | FI-24100 Salo | Finland
Office +358 2 727 7700 | Fax + 358 2 727 7720 | info@nordicid.com

SCOPE

This application note extends the NUR protocol documentation by showing a simple inventory command, its response, how to fetch the tags from the module's internal buffer and the ID buffer clear command. The inventory procedure with the NUR module is:

1. Inventory command
2. Retrieve the tag information from the internal buffer (can automatically clear the internal buffer as well).
3. Explicitly command the internal buffer to be cleared if needed

Scenario	Description
Basic inventory command	The basic inventory using specific values for Q, session and internal inventory rounds.
Inventory command's response	As the inventory command does not directly return the tags, it has an informative response about the inventory result.
Simple ID buffer reading	The simples cast of ID buffer read: the entries only contain the antenna ID and tag's EPC.
Advanced ID buffer reading: metadata included	More complex ID buffer reading: each tag also contains metadata as well as possible inventory + read data.
Fetching single tags from the module	These examples show how single tag's information is read from the module's internal buffer.
Single tag (or ID buffer) reading errors	A couple of error responses to ID buffer reads.
Clearing the ID buffer	The ID buffer clear command and its response.

INVENTORY COMMAND

COMMAND PACKET

This example command also contains the rounds, Q and session paraameters; they are not mandatory and if omitted, the default values from the module's internal settings are used. The application note AN0002 describing the module setup gives more information about the module setup.

```
A5 06 00 00 00 5C 31 05 00 02 2F 15
```

INVENTORY COMMAND CONTENTS

Field	Value	Description	
Header	A5060000005C (6 bytes)	Header consisting of:	
		A5	
		0600 = 0x0006	Payload + CRC length
		0x0000	Command flags
		0x5C	Header check sum
Command	0x31	Inventory command.	
Q	0x05	Use Q value of 5 (2^5 = 32 "slots").	
Session	0x00	Do inventory in session 0.	
Rounds	0x02	Value is 0x01FFFFFF. See all setup flags.	
Payload CRC	0x2F 0x15	Little endian; value is **0x152F**.	

C-STRUCTURE EXAMPLE OF THE INVENTORY COMMAND

```
/* Generic request structure */
struct __packed NURCMDHDR
{
  uint8_t start;    /* 0x0A5 */
  uint16_t payLen;  /* Following payload length including CRC-16.*/
  uint16_t flags;   /* Protocol/command flags. */
  uint8_t char cs;  /* Header checksum. */
};

struct __packed INVENTORYCMD
{
  struct NURCMDHDR hdr;
  uint8_t inventoryCmd; /* 0x31 */
  uint8_t Q;  /* Q as specified by the Gen2 protocol. */
  uint8_t session; /* Session as well. */
  uint8_t rounds; /* Internal rounds */
  uint16_t crc;
};
```

EXAMPLE INVENTORY RESPONSE

The example response tells that the module has found 2 tags during the last inventory and that there are total of 3 tags currently stored into the module's internal buffer.

RESPONSE PACKET

```
A5 0C 00 00 00 56 31 00 00 00 03 00 02 00 00 05 CA 50
```

RESPONSE CONTENTS

Byte(s)	Value	Description	
0…5	A50C00000056 (6 bytes)	Header consisting of:	
		A5	
		0C00 = 0x000C (12)	Payload + CRC length
		0x0000	Response flags
		0x56	Header check sum
6	31	Command echo.	
7	00	Status: 0 = OK.	
8…9	02 00	Unsigned 16-bit, little-endian: tags found during last inventory call = 0x0002.	
10…11	03 00	Unsigned 16-bit, little-endian: tags currently stored into the module's internal buffer (including the last found) = 0x0003.	
12	02	Byte, inventory rounds executed during last inventory call: 2.	
13…14	00 00	Unsigned 16-bit, little-endian: detected collisions during last inventory call: 0x0000.	
15	05	Byte, last Q-value that was used during last inventory round.	
Payload CRC	CA 50	Little endian; value is **0x50CA.**	

INVENTORY RESPONSE C-STRUCTURE EXAMPLE

```
/* Inventory response. */
struct __packed INVENTORYRESP
{
  uint8_t inventoryCmd; /* Command echo */
  uint16_t lastTags; /* Nr of tags during last execution. */
  uint16_t stored;  /* Tags currently stored into internal buffer. */
  uint8_t rounds; /* Rounds done during last execution. */
  uint16_t coll; /* Detected collision during last execution. */
  uint8_t lastQ; /* Last used Q. */
};
```

FETCHING TAG BUFFER

After an inventory has been executed it is then required that the tag buffer is read with a separate command. The command also includes instrction to the module whether to clear the internal buffer after it is being sent or to keep it in the memory. The command value is either

- **0x06**: fetch antenna ID and tag's EPC contents only. The tag's EPC contents may also contain inventory + read data (data only) if the inventory read was configured to execute in "data only" mode
- **0x07**: each tag entry is returned with this additional information:
  - RSSI in dBm
  - scaled RSSI (0...100%)
  - channel
  - frequency
  - time stamp (milliseconds from the beginning of the inventory execution)
  - PC (Protocol Control word) of the tag
  - possible inventory + read data (if the inventory read was configured to execute in EPC + data mode)

It is also possible to fetch the tag data one tag at a time, see Single tag data fetching.

SIMPLE BUFFER FETCHING

GET ID BUFFER PACKET

```
A5 04 00 00 00 5E 06 01 88 A7
```

GET ID BUFFER CONTENTS

Byte(s)	Value	Description	
0…5	A5040000005E (6 bytes)	Header consisting of:	
		A5	
		0400 = 0x0004	Payload + CRC length
		0x0000	Command flags
		0x5E	Header check sum
6	06	Get ID buffer command.	
7	01	Clear buffer after read: 1 = yes, 0= no.	
8…9	88 A7	Unsigned 16-bit, little-endian: packet CRC-16 = **0xA788**.	

SIMPLE BUFFER RESPONSE EXAMPLE

In this example an inventory has been executed. The module's internal buffer was empty before the inventory and during the inventory execution the module found 2 tags.

RESPONSE PACKET

```
A5 20 00 00 00 7A 06 00 0D 00 30 38 E5 11 C7 35 D2 C0 D9 C8 2A 25 0D 00
30 00 00 00 07 89 00 40 00 00 00 02 65 CD
```

RESPONSE PACKET CONTENTS

Byte(s)	Value	Description	
0…5	A5200000007A (6 bytes)	Header consisting of:	
		A5	
		2000 = 0x0020 (32)	Payload + CRC length
		0x0000	Response flags
		0x7A	Header check sum
6	06	Command echo.	
7	00	Status: 0 = OK.	
First tag entry			
8	0D	Bytes to follow (13); this tag entry's size – 1.	
9	00	Antenna ID (source antenna).	
10…21	30 38 E5 11 C7 35 D2 C0 D9 C8 2A 25	Tag's EPC.	
Second tag entry			
22	0D	Bytes to follow (13); this tag entry's size – 1.	
23	00	Antenna ID (source antenna).	
24…35	30 00 00 00 07 89 00 40 00 00 00 02	Tag's EPC.	
36…37	65 CD	Unsigned 16-bit, little-endian: packet CRC-16 = **0xCD65**.	

TAG BUFFER C-STRUCTURE EXAMPLE

```
/* Single tag information response. */
struct __packed SIMPLETAGINFO
{
  uint8_t length;  /* Antenna ID + EPC length. */
  uint8_t ant;     /* Source antenna */
  uint8_t epc[1];  /* variable length; EPC contents. */
};
```

Naturally the entries need to be handled as a variable length array with the number of entries matching the expected number of tags from the module.

BUFFER FETCHING WITH METADATA

In this example an inventory has been executed. The module's internal buffer was empty before the inventory and during the inventory execution the module found 2 tags + the ID buffer is read includeing the associated metadata.

GET ID BUFFER WITH METADATA PACKET

```
A5 04 00 00 00 5E 07 01 B9 94
```

GET ID BUFFER WITH METADATA PACKET CONTENTS

Byte(s)	Value	Description	
0…5	A5040000005E (6 bytes)	Header consisting of:	
		A5	
		0400 = 0x0004	Payload + CRC length
		0x0000	Command flags
		0x5E	Header check sum
6	07	Get ID buffer with metadata command.	
7	01	Clear buffer after read: 1 = yes, 0= no.	
8…9	B9 94	Unsigned 16-bit, little-endian: packet CRC-16 = **0x94B9.**	

The two byte command + parameter structure is the same as in the get ID buffer example only with the command value of 0x07.

METADATA BUFFER EXAMPLE

RESPONSE PACKET

```
A5 36 00 00 00 6C 07 00 18 C4 4C 1B 00 54 3A 0D 00 00 30 02 00 30 00 00
00 07 89 00 40 00 00 00 02 18 CC 64 0C 00 54 3A 0D 00 00 30 02 00 30 38
E5 11 C7 35 D2 C0 D9 C8 2A 25 6B 74
```

RESPONSE CONTENTS

Byte(s)	Value	Description	
0…5	A5360000006C (6 bytes)	Header consisting of:	
		A5	
		3600 = 0x0036 (52)	Payload + CRC length
		0x0000	Response flags
		0x6C	Header check sum
6	07	Command echo.	
7	00	Status: 0 = OK.	
First tag entry			
8	18	Bytes to follow (24); this tag entry's size – 1: 0x18 = 24.	
9	C4	Signed 8-bit, RSSI: 0xC4 = -60 dBm	
10	4C	Unsigned 8-bit, scaled RSSI: 0x4C = 76%.	
11…12	1B 00	Unsigned 16-bit, little-endian, timestamp in milliseconds: 0x001B = 27 (from the beginning of the inventory.	
13…16	54 3A 0D 00	Unsigned 32-it, frequency in kHz: 0x000D3A54 = 866900 kHz (866.9 MHz).	
17…18	00 30	Unsigned 16-bit, little-endian, PC word: 0x3000.	
19	02	Unsigned 8-bit, channel: 2	
20	00	Unsigned 8-bit, antenna ID: 0	
21…32	30 00 00 00 07 89 00 40 00 00 00 02	Tag's EPC.	
Second tag entry			
33	18	Bytes to follow: 0x18 = 24.	
34	CC	RSSI: 0xCC = -52	
35	64	Scaled RSSI: 0x64 = 100%	
36…37	0C 00	Timestamp: 0x000C = 12 milliseconds.	
38…41	54 3A 0D 00	Frequency in kHz: 0x000D3A54 = 866900 kHz (866.9 MHz).	
42…43	00 30	PC word = 0x3000.	
44	02	Channel = 2	
45	00	Antenna ID = 0	
46…57	30 38 E5 11 C7 35 D2 C0 D9 C8 2A 25	Tag's EPC.	
58…59	6B 74	Unsigned 16-bit, little-endian: packet CRC-16 = **0x746B.**	

METADATA TAG ENTRY C-STRUCTURE EXAMPLE

There is also one additional tag entry; the one that also contains the inventory + read data.

```
/* Metadata information fetch response.*/

struct __packed NUR_METADATAENTRY
{
  uint8_t length;       /* Number of bytes to follow. */
  int8_t rssi;          /* In dBm. */
  uint8_t scaledRssi;   /* Percentage. */
  uint16_t timeStamp;   /* From the beginning of the inventory. */
  uint32_t freq;        /* In kHz. */
  uint16_t pc;          /* Tag's PC field in EPC memory. */
  uint8_t channel;      /* Inventory channel number. */
  uint8_t antId;        /* Antenna ID. */
  uint16_t epcData[1]; /* Variable length EPC data. */
};
```

METADATA RESPONSE WHEN THE INVENTORY + READ DATA IS PRESENT

Response packet

```
A5 40 00 02 00 18 07 00 1D CC 64 1C 00 54 3A 0D 00 08 00 20 02 00 30 00
00 00 07 89 00 40 E2 80 11 30 20 00 29 3D 1D CA 64 10 00 54 3A 0D 00 08
00 20 02 00 30 38 E5 11 C7 35 D2 C0 E2 80 11 30 20 00 25 2A E9 F9
```

Response packet contents (inventory = inventory + read; 4 first words from the TID memory.)

Byte(s)	Value	Description	
0…5	A54000020018 (6 bytes)	Header consisting of:	
		A5	
		40 00 = 0x0040 (64)	Payload + CRC length
		02 00 = 0x0002	**Response flags: bit 1 is set = inventory + read data is present**
		0x18	Header check sum
6	07	Command echo.	
7	00	Status: 0 = OK.	
First tag entry			
8	1D	Bytes to follow (29); this tag entry's size – 1: 0x1D =	
9	CC	RSSI: 0xCC = -52 dBm	
10	64	Scaled RSSI: 0x64 = 100%	
11…12	1C 00	Timestamp: 0x001C = 28 milliseconds.	
13…16	54 3A 0D 00	Frequency in kHz: 0x000D3A54 = 866900 kHz (866.9 MHz).	
17	**08**	**EPC + data –field's data part length: 8 bytes (last 8).**	
18…19	00 20	PC word = 0x2000.	
20	02	Channel = 2	
21	00	Antenna ID = 0	
22…35	**30 00 00 00 07 89 00 40** *E2 80 11 30 20 00 29 3D*	8 bytes of EPC + 8 bytes of data: **EPC[0…7] = 3000000007890040** *Data[8…15] = E28011302000293D*	
Second tag entry			
36	1D	Bytes to follow (29); this tag entry's size – 1: 0x1D =	
37	CA	RSSI: 0xCA = -54 dBm	
38	64	Scaled RSSI: 0x64 = 100%	
39…40	10 00	Timestamp: 0x0010 = 16 milliseconds (note: first time seen)	
41…44	54 3A 0D 00	Frequency in kHz: 0x000D3A54 = 866900 kHz (866.9 MHz).	
45	**08**	**EPC + data –field's data part length: 8 bytes (last 8).**	
46…47	00 20	PC word = 0x2000.	
48	02	Channel = 2	
49	00	Antenna ID = 0	
50…65	**30 38 E5 11 C7 35 D2 C0** *E2 80 11 30 20 00 25 2A*	8 bytes of EPC + 8 bytes of data: **EPC[0…7] = 3038E511C735D2C0** *Data[8…15] = E28011302000252A*	
66…67	E9 F9	Unsigned 16-bit, little-endian: packet CRC-16 = **0xF9E9 .**	

Fetching single tags

There are many cases where in embedded systems, namely due to the small size of RAM memory, it is needed to get the tags from the NUR module's tag buffer one by one. This document shows how the tags can be read out in various ways.

THE FETCH TAG COMMAND

This part only describes the packet fetching packet contents. The various response types are shown in the examples.

The "fetch tag at" uses the get ID buffer command 0x06 (EPC only) or 0x07 (EPC + metadata) appended with index parameter. When the inventory command responds with how many tags there are in the internal buffer, this index parameter must be in range 0…tag count – 1. The index is not an array index; it merely represents the number that was stored along with the new tag information stored during the inventory. Thus, it is an order number telling at which point the tag was added into the internal buffer.

FETCH SINGLE TAG VARIATIONS

There are two ways to get the tag information: antenna ID appended with EPC and full metadata information (antenna, RSSI, timestamp, PC etc. + EPC and possible inventory + read data).

The packet when reading only the antenna ID + EPC of a single tag is:

Byte(s)	Is	Description
0…5	Command header	Protocol defined packet header.
6	Get ID buffer command.	The command value is 0x06: get ID buffer without metadata.
7…10	Tag "index"	Index in the range 0…number of tags in buffer – 1.
11…12	Mask length	Packet CRC.

The packet when reading a single tag information data with metadata is:

Byte(s)	Is	Description
0…5	Command header	Protocol defined packet header.
6	Get ID buffer command.	The command value is 0x07: get ID buffer with metadata.
7…10	Tag "index"	Unsigned 32-bit, little-endian: index in the range 0…number of tags in buffer – 1.
11…12	Packet CRC	Unsigned 16-bit, little-endian.

FETCH SINGLE TAG C-STRUCTURE EXAMPLE

The two command values are also included.

```
/* Block that how a single tag information is retrieved from the
module. */

#define FETCH_EPCONLY          0x06
#define FETCH_METADATA         0x07

struct __packed NUR_FETCHTAG
{
  uint8_t cmd;       /* 0x06 or 0x07. */
  uint32_t index;
};
```

SIMPLE ANTENNA ID AND EPC FETCHING

COMMAND PACKET

This packet fetches tag data at index 0:

A5 07 00 00 00 5D 06 00 00 00 00 89 DC

COMMAND PACKET CONTENTS

Byte(s)	Value(s) HEX	Description	
0…5	A5070000005D (6 bytes)	Header consisting of:	
		A5	
		0700 = 0x0007	Payload + CRC length
		0x0000	Command flags
		0x5D	Header check sum
6	06	Get ID buffer without metadata.	
7…10	00 00 00 00	32-bit unsigned, little-endian; index = 0. Note: the length of the command determines that this is a single tag information fetch.	
46…47	89 DC	Unsigned 16-bit, little-endian: packet CRC-16 = **0xDC89**.	

SIMPLE FETCH RESPONSE

The response packet contains only the antenna ID and EPC contents. Note that the EPC contents can also be inventory + read data if the inventory + read was configured to be “data only” in which case the EPC contents is not stored into the module’s internal buffer at all.

RESPONSE PACKET

A response example:

```
A5 12 00 00 00 48 06 00 0D 00 CC DD 44 30 31 32 33 34 00 00 00 00 89 62
```

RESPONSE CONTENTS

Byte(s)	Value(s) HEX	Description	
0…5	A51200000048 (6 bytes)	Header consisting of:	
		A5	
		1200 = 0x0012 (18)	Payload + CRC length
		0x0000	Command flags
		0x48	Header check sum
6	06	Command echo.	
7	00	Status: 0 = OK.	
8	0D	Bytes to follow: antenna ID + EPC length.	
9	00	Antenna ID.	
7…10	CC DD 44 30 31 32 33 34 00 00 00 00	Tag’s EPC; 12 bytes.	
46…47	89 62	Unsigned 16-bit, little-endian: packet CRC-16 = **0x6289**.	

SIMPLE FETCH C-STRUCTURE EXAMPLE

```
/* Simple tag information fetch response.*/

struct __packed NUR_SINGLETAGINFO
{
  uint8_t cmd;       /* Command echo */
  uint8_t status; /* 0 = OK, others = error. */
  uint8_t length; /* Length of following antenna ID and EPC. */
  uint8_t antId;  /* Antenna source. */
  uint8_t epc[0]; /* EPC contents. */
};
```

METADATA FETCH EXAMPLE

COMMAND PACKET

This packet fetches tag metadata at index 0:

```
A5 07 00 00 00 5D 07 00 00 00 00 D8 76
```

COMMAND PACKET CONTENTS

Byte(s)	Value(s) HEX	Description	
0…5	A5070000005D (6 bytes)	Header consisting of:	
		A5	
		0700 = 0x0007	Payload + CRC length
		0x0000	Command flags
		0x5D	Header check sum
6	07	Get ID buffer with metadata.	
7…10	00 00 00 00	32-bit unsigned, little-endian; index = 0. Note: the length of the command determines that this is a single tag information fetch.	
46…47	D8 76	Unsigned 16-bit, little-endian: packet CRC-16 = **0x76D8**.	

METADATA FETCH RESPONSE

RESPONSE PACKET

A response example:

```
A5 1D 00 00 00 47 07 00 18 CC 64 05 00 54 3A 0D 00 00 30 02 00 CC DD 44
30 31 32 33 34 00 00 00 00 63 B1
```

RESPONSE CONTENTS

Byte(s)	Value(s) HEX	Description	
0…5	A51D00000047 (6 bytes)	Header consisting of:	
		A5	
		1D00 = 0x001D (29)	Payload + CRC length
		0x0000	Response flags
		0x47	Header check sum
6	07	Command echo.	
7	00	Status: 0 = OK.	
8	18	Bytes to follow: 24	
9	CC	Signed 8-bit RSSI in dBm: 0xCC = -52.	
10	64	Unsigned 8-bit, scaled RSSI: 0x64 = 100%	
11…12	05 00	Unsigned 16-bit, timestamp in milliseconds: 0x0005 = 5.	
13…16	54 3A 0D 00	32-bit unsigned, frequency in kHz, little-endian: 0x000D3A54 = 866900kHz (866.9MHz)	
17…18	00 30	Unsigned 16-bit, little-endian, tag's PC: 0x3000.	
19	02	Byte, channel number = 2.	
20	00	Antenna ID: 0	
21…32	CC DD 44 30 31 32 33 34 00 00 00 00	EPC data.	
33…34	63 B1	Unsigned 16-bit, little-endian: packet CRC-16 = **0xB163**.	

METADATA FETCH C-STRUCTURE EXAMPLE

```
/* Metadata information fetch response.*/
struct __packed NUR_METADATAENTRY
{
  uint8_t cmd;          /* Command echo. */
  uint8_t status;       /* 0 = OK, others = error. */
  uint8_t length;       /* Number of bytes to follow. */
  int8_t rssi;          /* In dBm. */
  uint8_t scaledRssi;   /* Percentage. */
  uint16_t timeStamp;   /* From the beginning of the inventory. */
  uint32_t freq;        /* In kHz. */
  uint16_t pc;          /* Tag's PC field in EPC memory. */
  uint8_t channel;      /* Inventory channel number. */
  uint8_t antId;        /* Antenna ID. */
  uint16_t epcData[1]; /* Variable length EPC data. */
};
```

METADATA FETCH RESPONSE INCLUDING INVENTORY + READ DATA

This example assumes that the fetched metadata also includes inventory + read data as the inventory + read was configure to produce EPC + data. In the response packet's packet flag set the "contains inventory + read data" flag is set (bit 1, mask 0x0002).

RESPONSE PACKET

```
A5 26 00 02 00 7E 07 00 21 CC 64 07 00 54 3A 0D 00 08 00 30 02 00 CC DD
44 30 31 32 33 34 00 00 00 00 E2 80 68 10 20 00 00 01 33 AB
```

RESPONSE CONTENTS

Byte(s)	Value(s) HEX	Description	
0…5	A5260002007E (6 bytes)	Header consisting of:	
		A5	
		2600 = 0x0026 (38)	Payload + CRC length
		02 00 = 0x0002	**Response flags: inventory + read data is present, bit 2 (mask 0x0002) is set.**
		0x7E	Header check sum
6	07	Command echo.	
7	00	Status: 0 = OK.	
8	21	Bytes to follow: 33	
9	CC	Signed 8-bit RSSI in dBm: 0xCC = -52.	
10	64	Unsigned 8-bit, scaled RSSI: 0x64 = 100%	
11…12	07 00	Unsigned 16-bit, timestamp in milliseconds: 0x0007 = 7.	
13…16	54 3A 0D 00	32-bit unsigned, frequency in kHz, little-endian: 0x000D3A54 = 866900kHz (866.9MHz)	
17	08	Byte, length of the data part: 8 bytes. Tells how the EPC + data later on is split (EPC first, last 8 bytes are read data).	
18…19	00 30	Unsigned 16-bit, little-endian, tag's PC: 0x3000.	
19	02	Byte, channel number = 2.	
20	00	Antenna ID: 0	
21…40	*CC DD 44 30 31 32 33 34 00 00 00 00* **E2 80 68 10 20 00 00 01**	EPC + read data. *EPC[12] = CCDD4430313233400000000* **Data[8] = E280681020000001**	
41…42	33 AB	Unsigned 16-bit, little-endian: packet CRC-16 = **0xAB33**.	

METADATA FETCH C-STRUCTURE EXAMPLE (INCLUDING INVENTORY + READ DATA)

```
/* Metadataresponse including inventory + read data. */

struct __packed NUR_METADATAENTRY_EX
{
  uint8_t cmd;          /* Command echo. */
  uint8_t status;       /* 0 = OK, others = error. */
  uint8_t length;       /* Number of bytes to follow. */
  int8_t rssi;          /* In dBm. */
  uint8_t scaledRssi;   /* Percentage. */
  uint16_t timeStamp;   /* From the beginning of the inventory. */
  uint32_t freq;        /* In kHz. */
  uint8_t dataLen;      /* Number of inventory + read data bytes.*/
  uint16_t pc;          /* Tag's PC field in EPC memory. */
  uint8_t channel;      /* Inventory channel number. */
  uint8_t antId;        /* Antenna ID. */
  uint16_t epcData[1]; /* Variable length EPC + read data. */
};
```

ERROR RESPONSE CONTENTS

When the tag fetching end up with an error the content of the error packet is:

Byte(s)	Is	Description
0…5	Command header	Protocol defined packet header.
6	Get ID buffer command.	The command value is 0x06: get ID buffer without metadata.
7	Error code	Index in the range 0…number of tags in buffer – 1.
8…9	Packet CRC	Unsigned 16-bit, little-endian.

ERROR EXAMPLE: INVALID INDEX

This example packet is received from the module when the tag index is bigger than the module's internal tag count – 1:

```
A5 04 00 00 00 5E 06 05 0C E7
```

The response contents is

Byte(s)	Value(s) HEX	Description	
0…5	A5040000005E (6 bytes)	Header consisting of:	
		A5	
		0400 = 0x0004	Payload + CRC length
		00 00 = 0x0000	Response flags.
		0x5E	Header check sum
6	06	Command echo: get ID buffer, no metadata.	
7	05	Status = 0x05: invalid parameter caused by the index being invalid.	
8…9	0C E7	Unsigned 16-bit, little-endian: packet CRC-16 = **0xE70C**.	

ERROR EXAMPLE: NO TAGS IN THE MODULE

This example packet is received from the module when no tags are currently stored into the module's internal buffer.

```
A5 04 00 00 00 5E 06 20 CB 93
```

The response contents is

Byte(s)	Value(s) HEX	Description	
0…5	A5040000005E (6 bytes)	Header consisting of:	
		A5	
		0400 = 0x0004	Payload + CRC length
		00 00 = 0x0000	Response flags.
		0x5E	Header check sum
6	06	Command echo: get ID buffer, no metadata.	
7	20	Status = 0x20 (32): no tag(s).	
8…9	CB 93	Unsigned 16-bit, little-endian: packet CRC-16 = **0x93CB**.	

EXPLICIT CLEARING OF THE ID BUFFER

The module's internal ID buffer can also be cleared explicitly. Here's a command example.

COMMAND PACKET

```
A5 03 00 00 00 59 05 55 B1
```

COMMAND PACKET CONTENTS

Byte(s)	Value(s) HEX	Description	
0…5	A50300000059 (6 bytes)	Header consisting of:	
		A5	
		0300 = 0x0003	Payload + CRC length
		00 00 = 0x0000	Packet flags.
		0x59	Header checksum
6	05	Command value: 0x05, clear ID buffer.	
7…8	55 B1	Unsigned 16-bit, little-endian: packet CRC-16 = **0xB155**.	

RESPONSE PACKET

```
A5 04 00 00 00 5E 05 00 FA E2
```

RESPONSE PACKET CONTENTS

Byte(s)	Value(s) HEX	Description	
0…5	A5040000005E (6 bytes)	Header consisting of:	
		A5	
		0400 = 0x0004	Payload + CRC length
		0x0000	Response flags.
		0x5E	Header checksum
6	05	Command value: 0x05, clear ID buffer.	
7	00	Status: 0 = OK (always successful).	
8…9	FA E2	Unsigned 16-bit, little-endian: packet CRC-16 = **0xE2FA**.	