NUR APPLICATION NOTE 5 (NUR AN005)

GETTING STARTED: READING A TAG'S MEMORY CONTENTS

SCOPE

This document shows several cases of tag memory reading. This includes non-selected (no tag singulation) read and singulation based reading both without password and with password.

GENERAL READ PACKET CONTENTS

The read command consists of read command code that is followed by:

- Control flags and password (block is mandatory)
- Singulation (tag selection) block (optional, presence indicated by the control flags)
- Read instruction block: bank, address, word count (block is mandatory)

READ COMMAND BLOCK'S CONTENTS

The offsets are presented from each block's base i.e. starting from 0. The common block starts right after the command code which is `0x33 (51)`.

COMMON BLOCK

Common block is always present for any read or write (or alike command such as lock or kill).

COMMON BLOCK CONTENTS

Byte(s)	Is	Description	
0	Flags	Bit flags controlling the read operation:	
		Bit	**Is**
		0 (mask 0x01)	If '1', the operation uses tag accessing by the following password.
		1 (mask 0x02)	If '1', this part is then followed by a singulation block defining how the tag is selected.
		2 (mask 0x04)	If '1', then any operation that requires addressing (namely read/write) within a memory bank uses 64-bit addressing; '0' (default) means 32-bit addressing.
		3 (mask 0x08)	If '1', then the singulation uses 64-bit addressing; '0' (default) means 32-bit addressing.
		4…7	Not used; set to '0'.
1…4	Password.	32-bit unsigned: password in little-endian format. Always present whether used or not.	

COMMON BLOCK C-STRUCTURE EXAMPLE

NOTE: this example here also includes the command field. This structure is common for all read and write (and alike) commands.

```
struct __packed COMMON_RW_BLOCK
{
  /* Read/write/lock/kill command value (read=0x33) */
  unsigned char rwlkCmd;
  /* Control flags. */
  unsigned char flags;
  /* Password value fo secured tag accessing if used. */
  unsigned int password;
};
```

SINGULATION (TAG SELECTION) BLOCK

The application note NUR_AN010: tag selection explains the bit level operation of the tag singulation. The leftmost column shows the byte offsets when using 32-bit addressing and the following shows the byte offsets when using 64-bit addressing.

SINGULATION BLOCK CONTENTS

Byte(s) (32-bit addr)	Byte(s) (64-bit addr)	Is	Description
0	0	Bytes to follow	Number of bytes to follow = this block’s size in bytes - 1.
1	1	Bank	Selection bank: 1 (EPC), 2 (TID) or 3 (user memory).
2…5	2…9	Mask address	The *bit address* within the selection bank where the
6…7	10..11	Mask length	The *bit length* of the selection mask.
8…n	12…n	Mask data.	The bit mask data used for selection. Length must be the bit length divided by 8; if length *mod 8 is not zero* then 1 is added.

SELECTION BLOCK’S C-STRUCTURE EXAMPLE

```
/* Block that defines tag selection parameters: 32-bit */
struct __packed NUR_TAGSELBLOCK32
{
  uint8_t size;       /* Number of bytes to follow. */
  uint8_t bank;       /* Selection bank. */
  uint32_t bitAddr;   /* 32-bit address of the selection mask. */
  uint16_t bitLen;    /* Bit length of the selection mask. */
  uint8_t bitBuf[1]; /* Variable length bit buffer data. */
};

/* Block that defines tag selection parameters: 64-bit */
struct __packed NUR_TAGSELBLOCK64
{
  uint8_t size;       /* Number of bytes to follow. */
  uint8_t bank;       /* Selection bank. */
  uint64_t bitAddr;   /* 64-bit address of the selection mask. */
  uint16_t bitLen;    /* Bit length of the selection mask. */
  uint8_t bitBuf[1]; /* Variable length bit buffer data. */
};
```

THE READ BLOCK

READ BLOCK'S CONTENTS

Byte(s) (32-bit addr)	Byte(s) (64-bit addr)	Is	Description
0	0	Bytes to follow	Number of bytes to follow = this block's size in bytes - 1.
1	1	Bank	The bank to read from: 0 (password/reserved), 1 (EPC), 2 (TID) or 3 (user memory).
2…5	2…9	Word address	The word address to read from; 16-bit aligned as per Gen2 specification.
6	10	Word count	Number of 16-bit words to read.

READ BLOCK'S C-STRUCTURE EXAMPLE

```
/* Block that defines tag read parameters: 32-bit addressing */
struct __packed NUR_TAGREADBLOCK32
{
  uint8_t size;       /* Number of bytes to follow. */
  uint8_t bank;       /* Bank to read from. */
  uint32_t wordAddr;  /* 32-bit word address to read from. */
  uint16_t wordCount;   /* Word count. */
};

/* Block that defines tag read parameters: 64-bit addressing */
struct __packed NUR_TAGREADBLOCK64
{
  uint8_t size;       /* Number of bytes to follow. */
  uint8_t bank;       /* Bank to read from. */
  uint64_t wordAddr;  /* 64-bit word address to read from. */
  uint16_t wordCount;   /* Word count. */
};
```

THE READ COMMAND RESPONSE

A successful read response consists of command byte echo followed by

- status byte
- data from the tag

A SUCCESSFUL READ RESPONSE CONTENTS

Byte(s)	Is	Description
0	Command echo	Read command echo: 0x33.
1	Status	0 = OK
2…n	Read data	Length is the word count multiplied by 2.

A SUCCESSFUL READ RESPONSE C-STRUCTURE EXAMPLE

The example also includes command echo and status.

```
struct __packed NUR_TAGREADRESP
{
  uint8_t cmd;      /* Command echo. */
  uint8_t status;   /* Status (0=OK). */
  uint8_t data[1]; /* Variable length data. */
};
```

Such a response structure is assumed to be pointed via for example byte pointer in order to access the members correctly.

THE READ COMMAND ERROR RESPONSE

When the actual read execution fails (no parameter errors) the reader sends back the error code received from the tag if the error happened at the tag's side. Such a situation is for example memory overrun i.e. non-existing memory address is tried to read. The module's error code is then **`0x42`**: tag error.

READ ERROR: TAG ERROR PACKET CONTENTS

Byte(s)	Is	Description
0	0x33	Read command echo: 0x33.
1	0x42	Tag error, code follows.
2	<error>	If present, can be either error code as specified by the Gen2 protocol specification or module's error flag set.
3…4	CRC	The response CRC-16.

TAG ERROR EXAMPLE C-STRUCTURE

Following structure can be used for both error interpretations i.e. with tag error being sent back, error flag set being sent back and without error information.

```
struct __packed READ_TAGERROR
{
  /* Can also contain error flags if not tag error. */
  uint8_t error;
  uint16_t crc16;
};

struct __packed TAGERRORRESP
{
  uint8_t cmd;
  uint8_t error;
  union {
    struct READ_TAGERROR error;
    uint16_t crc16;
  } ext;
};
```

READ EXAMPLE: TID MEMORY, EPC SELECTION

In example's tag the 4 first word of the TIC memory are:

```
E2 80 68 10 20 00 00 01
```

READ PACKET

```
A5 23 00 00 00 79 33 02 00 00 00 00 13 01 20 00 00 00 60 00 CC DD 44 30
31 32 33 34 00 00 00 00 06 02 00 00 00 00 04 CC 46
```

READ PACKET CONTENTS

Byte(s)	Value(s) HEX	Description	
0…5	`A52300000079` (6 bytes)	Header consisting of:	
		A5	
		2300 = 0x0023 (35)	Payload + CRC length
		0x0000	Command flags
		0x79	Header check sum
Common block			
6	`33`	Read tag	
7	`02`	Byte, control flags: bit 1 (mask 0x02) is set: the common block is followed by singulation block	
8…11	`00 00 00 00`	Unsigned 32-bit, little-endian: password, not used, set to 0.	
Singulation block			
12	`13`	Bytes to follow (19): size of singulation block – 1.	
13	`01`	Bank where the selection mask is applied to: 1 = EPC	
14…17	`20 00 00 00`	Unsigned 32-bit, little-endian: mask bit address, 0x00000020 (32): the EPC's bit start address.	
18..19	`60 00`	Unsigned 16-bit, little-endian: selection mask's bit length: 0x0060 (96)	
20…31	`CC DD 44 30 31 32 33 34 00 00 00 00`	Selection mask, 12 bytes (96 bits): the tag's EPC.	
Read block			
32	`06`	Bytes to follow: size of read block – 1.	
33	`02`	Bank to read: 2 = TID	
33…36	`00 00 00 00`	Unsigned 32-bit, little-endian: word address to start read from.	
37	`04`	Byte: word count (Number of 16-bit words to read).	
38…39	`CC 46`	Unsigned 16-bit, little-endian: packet CRC-16 = **`0x46`CC**.	

READ EXAMPLE: TID MEMORY RESPONSE

RESPONSE PACKET

```
A5 0C 00 00 00 56 33 00 E2 80 68 10 20 00 00 01 64 D8
```

RESPONSE CONTENTS

Byte(s)	Value(s) HEX	Description	
0…5	`A50C00000056` (6 bytes)	Header consisting of:	
		A5	
		0C00 = 0x000C (12)	Payload + CRC length
		0x0000	Command flags
		0x56	Header check sum
6	`33`	Command echo	
7	`00`	Status: 0 = OK	
8…15	`E2 80 68 10 20 00 00 01`	Data from the tag: 4 words, 8 bytes.	
16…17	`64 D8`	Unsigned 16-bit, little-endian: packet CRC-16 = **`0xD864`**.	

READ EXAMPLE: PASSWORD MEMORY, TID SELECTION

The example's tag is singulated by

```
E2 80 68 10 20 00 00 01
```

located in the TID memory, bit address 0.

The password memory contents is all zeros (8 bytes).

READ PACKET

```
A5 1F 00 00 00 45 33 02 00 00 00 00 0F 02 00 00 00 00 40 00 E2 80 68 10
20 00 00 01 06 00 00 00 00 00 04 78 9D
```

READ PACKET CONTENTS

Byte(s)	Value(s) HEX	Description	
0…5	`A51F00000045` (6 bytes)	Header consisting of:	
		A5	
		1F00 = 0x001F (31)	Payload + CRC length
		0x0000	Command flags
		0x45	Header check sum
Common block			
6	`33`	Read tag	
7	`02`	Byte, control flags: bit 1 (mask 0x02) is set: the common block is followed by singulation block	
8…11	`00 00 00 00`	Unsigned 32-bit, little-endian: password, not used, set to 0.	
Singulation block			
12	`0F`	Bytes to follow (15): size of singulation block – 1.	
13	`02`	Bank where the selection mask is applied to: 1 = TID	
14…17	`00 00 00 00`	Unsigned 32-bit, little-endian: mask bit address = 0	
18..19	`40 00`	Unsigned 16-bit, little-endian: selection mask's bit length: 0x0040 (64)	
20…27	`E2 80 68 10 20 00 00 01`	Selection mask, 12 bytes (96 bits): the TID contents in word addresses 0…3.	
Read block			
28	`06`	Bytes to follow: size of read block – 1.	
29	`00`	Bank to read: 2 = password/reserved memory	
30…33	`00 00 00 00`	Unsigned 32-bit, little-endian: word address to start read from.	
34	`04`	Byte: word count (Number of 16-bit words to read).	
35…36	`78 9D`	Unsigned 16-bit, little-endian: packet CRC-16 = **`0x9D78`**.	

READ EXAMPLE: PASSWORD MEMORY RESPONSE

RESPONSE PACKET

```
A5 0C 00 00 00 56 33 00 00 00 00 00 00 00 00 00 6D FD
```

RESPONSE CONTENTS

Byte(s)	Value(s) HEX	Description	
0…5	`A50C00000056` (6 bytes)	Header consisting of:	
		A5	
		0C00 = 0x000C (12)	Payload + CRC length
		0x0000	Command flags
		0x56	Header check sum
6	`33`	Command echo	
7	`00`	Status: 0 = OK	
8…15	`00 00 00 00` `00 00 00 00`	Data from the tag: 4 words, 8 bytes.	
16…17	`6D FD`	Unsigned 16-bit, little-endian: packet CRC-16 = **`0xFD6D`**.	

READ EXAMPLE: PASSWORD MEMORY, EPC SELECTION USING PASSWORD

In this example the whole contents of password memory (both passwords locked) is read using the access password 0xACDCABBA. As the kill password is written to value 0xDEADBEEF, the expected read response in this case is `DE AD BE EF AC DC AB BA`.

The tag's EPC which the selection is based on is `30 00 00 00 07 89 00 40 00 00 00 02`.

READ PACKET

```
A5 23 00 00 00 79 33 03 BA AB DC AC 13 01 20 00 00 00 60 00 30 00 00 00
07 89 00 40 00 00 00 02 06 00 00 00 00 00 04 08 3C
```

READ PACKET CONTENTS

Byte(s)	Value(s) HEX	Description	
0…5	`A52300000079` (6 bytes)	Header consisting of:	
		A5	
		2300 = 0x0023 (35)	Payload + CRC length
		0x0000	Command flags
		0x79	Header check sum
Common block			
6	`33`	Read tag	
7	`03`	Byte, control flags: bits 0 and 1 are set thus the command includes the singulation block (bit 1) and the password is used to access the addressed memory (bit 0).	
8…11	`BA AB DC AC`	Unsigned 32-bit, little-endian: password to use for access = 0xACDCABBA.	
Singulation block			
12	`13`	Bytes to follow (19): size of singulation block – 1.	
13	`01`	Bank where the selection mask is applied to: 1 = EPC	
14…17	`20 00 00 00`	Unsigned 32-bit, little-endian: mask bit address, 0x00000020 (32): the EPC's bit start address.	
18..19	`60 00`	Unsigned 16-bit, little-endian: selection mask's bit length: 0x0060 (96)	
20…31	`30 00 00 00` `07 89 00 40` `00 00 00 02`	Selection mask, 12 bytes (96 bits): the tag's EPC.	
Read block			
32	`06`	Bytes to follow: size of read block – 1.	
33	`00`	Bank to read: 0 = password/reserved	
33…36	`00 00 00 00`	Unsigned 32-bit, little-endian: word address to start read from.	
37	`04`	Byte: word count (number of 16-bit words to read).	
38…39	`08 3C`	Unsigned 16-bit, little-endian: packet CRC-16 = **`0x3C08`**.	

READ EXAMPLE: PASSWORD MEMORY RESPONSE

RESPONSE PACKET

```
A5 0C 00 00 00 56 33 00 DE AD BE EF AC DC AB BA C2 2D
```

RESPONSE CONTENTS

Byte(s)	Value(s) HEX	Description	
0…5	A50C00000056 (6 bytes)	Header consisting of:	
		A5	
		0C00 = 0x000C (12)	Payload + CRC length
		0x0000	Command flags
		0x56	Header check sum
6	33	Command echo	
7	00	Status: 0 = OK	
8…15	DE AD BE EF AC DC AB BA	Data from the tag: 4 words, 8 bytes. First 4 is the kill password (32-bit, big-endian) and last 4 is the access password (32-bit, big-endian)	
16…17	C2 2D	Unsigned 16-bit, little-endian: packet CRC-16 = **0x2DC2**.	

READ EXAMPLE: USER MEMORY, NO TAG SELECTION

When tag is read without selection there can be only one tag in the field. Otherwise an error occurs as a single tag cannot be accessed.

The accessed tag's user memory word addresses 0…7 are programmed to (shown in 16-bit words):

```
0101 0202 0303 0404 0505 0606 0707 0808
```

READ PACKET

```
A5 0F 00 00 00 55 33 00 00 00 00 00 06 03 00 00 00 00 08 B8 32
```

READ PACKET CONTENTS

Byte(s)	Value(s) HEX	Description	
0…5	A50F00000055 (6 bytes)	Header consisting of:	
		A5	
		0F00 = 0x000F (15)	Payload + CRC length
		0x0000	Command flags
		0x55	Header check sum
		Common block	
6	33	Read tag	
7	00	Byte, control flags: no bits set = no singulation, 32-bit addressing.	
8…11	00 00 00 00	Unsigned 32-bit, little-endian: password, not used, set to 0.	
		Singulation block → not present	
		Read block	
12	06	Bytes to follow: size of read block – 1.	
13	03	Bank to read: 3 = user memory	
13…16	00 00 00 00	Unsigned 32-bit, little-endian: word address to start read from.	
17	04	Byte: word count (number of 16-bit words to read).	
18…19	B8 32	Unsigned 16-bit, little-endian: packet CRC-16 = **0x32B8**.	

READ EXAMPLE: USER MEMORY RESPONSE

RESPONSE PACKET

```
A5 14 00 00 00 4E 33 00 01 01 02 02 03 03 04 04 05 05 06 06 07 07 08 08
FB D9
```

RESPONSE PACKET CONTENTS

Byte(s)	Value(s) HEX	Description	
0…5	A5140000004E (6 bytes)	Header consisting of:	
		A5	
		1400 = 0x0014 (20)	Payload + CRC length
		0x0000	Command flags
		0x4E	Header check sum
6	33	Command echo	
7	00	Status: 0 = OK	
8…15	01 01 02 02 03 03 04 04 05 05 06 06 07 07 08 08	Data from the tag's user memory: 8 words, 16 bytes.	
16…17	FB D9	Unsigned 16-bit, little-endian: packet CRC-16 = **0xD9FB**.	

READ ERROR INFORMATION

There are three types of error information the module can send:

1. Error code only. For example tag access error is like this; it has no additional information.
2. Error flag set. When the error a command parameter error, then this flag set is present.
3. Tag's error code. When the error is `G2_TAG_ERROR_RESP (0x42, 66)` then the additional error information is the code that the tag backscattered.

ADDITIONAL ERROR INFORMATION

After the header, and excluding the CRC, the error response in case of parameter error is:

Byte(s)	Value(s) HEX	Description	
6	`33`	Command echo	
7	`05`	Status: 0x05 = parameter error(s).	
8	`<error_code>`	**Additional error information**	
		Value	**Is**
		1	No start (common) block. Command length is less than the size of common block (5 bytes).
		2	Selection mask length mismatch: the byte length calculated from the bit length does not match the actual byte length.
		3	Selection mask's bank error, not in range 1…3.
		4	Bank error (in read block), not in range 0…3.
		5	The read block length is not 6 (32-bit addressing) or 10 (64-bit addressing) byte long.
		7	Read length is 0: NUR module currently does not support "whole bank read" due to receiver restrictions.

READ ERROR FROM TAG: MEMORY OVERRUN

Memory overrun occurs when a memory location that doesn't exit is being read. Error response in such a case looks like:

Byte(s)	Value(s) HEX	Description	
0…5	A5050000005F (6 bytes)	Header consisting of:	
		A5	Start
		0500 = 0x0005	Payload + CRC length
		0x0000	Command flags
		0x5F	Header check sum
6	33	Command echo	
7	42	Status: 0x42 (66) = error code from tag.	
8	03	Gen2 specified error code (EPCGlobal specification v1.2, Annex I).	
9…10	FB D9	Unsigned 16-bit, little-endian: packet CRC-16 = **0xD9FB**.	

READ ERROR: PARAMETER ERROR

The following response is received with selection mask error:

Byte(s)	Value(s) HEX	Description	
0…5	A5050000005F (6 bytes)	Header consisting of:	
		A5	Start
		0500 = 0x0005	Payload + CRC length
		0x0000	Command flags
		0x5F	Header check sum
6	33	Command echo	
7	05	Status: 0x05 parameter error.	
8	02	Bit 1 is set: selection mask error.	
9…10	DE 8F	Unsigned 16-bit, little-endian: packet CRC-16 = **0x8FDE**.	