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 RFID

Custom exchange example: simple read command

Version 1

NORDICID

1 Concept of custom exchange

The ”custom exchange” –concept in Nordic ID’s UHF RFID module means implementing a freely formed command as well as defining the RF front end behavior upon the execution of the command. Having such a concept is necessary in order to implement custom commands introduced in the class 1 generation 2 specification. The term “exchange” implies that the operation is a data exchange between the reader and the tag although the custom exchange can also be instructed to perform single or multiple transmissions only.

The class 1 generation 2 read command that is used in this example consists of these parts:

The read command is chosen for this example as the command itself is simple thus making it possible to illustrate the idea of the custom exchange and how its parameters relate to the requirements of the gen2 protocol.

The example read command’s “human readable” format is:

“Read 8 bytes from password memory address 0 from tag with 12 bytes as 200002010100000000006DB in its EPC bank at word address 2 (bit address 32)”

2 About notation

All the values presented like signed 32-bit integer, unsigned 16-bit integer and so on are all in little-endian format. The only data that is read from “left-to-right” is the actual read command’s contents, explained here.

3 Singulation

The singulation as a term means a way to target a specific tag by using a specific selection command that the target tag, and the target tag only, reacts to. In this example the tag is selected by its (in this case unique) EPC contents. The tag's EPC (appearing in e.g. an inventory) used in this example is shown below along with the word and bit address information (indexing from 0). EPC is in hexadecimal format:

EPC: `20 00 02 01 01 00 00 00 00 00 06 DB`

Address information	←bit address 32 word address 2		word address 2		word address 2		word address 2		word address 2		bit address 127→ word address 7	
EPC byte	0	1	2	3	4	5	6	7	8	9	10	11
Contents	20	00	02	01	01	00	00	00	00	00	06	DB

The singulation part then has the parameters set to bank = 1 (EPC), bit mask address = 32 (beginning of the EPC contents) and length of 96 bits (=12 bytes, 6 words). This part is also given some flags (stating that the singulation block is present) and a password (set to 0, always present, but the flags state that it is not used)

Common part (for read, write, lock, kill, custom exchange, etc., etc.) that follows the custom exchange command byte **0x3F**:

Byte(s)	Meaning	More	
0	Flags	The flag bits are defined like:	
		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 integer, little-endian, here: not used, set to 0.	

When the tag replies, accessed with password or without it, it backscatters handle to the reader. This handle is usually required for the following command to execute.

For the whole packet including specifics for the singulation, see "Final exchange protocol packet".

4 Controlling custom exchange

Followed by the singulation part that is used in this example is the actual contents of the custom exchange. It contains these parts:

- control flags
- transmission length in bits
- expected reception length in bits (if known)
- response timeout in milliseconds

<table>
<tr><th>Byte(s)</th><th>Meaning</th><th colspan="2">More</th></tr>
<tr><td>0…1</td><td>Control flags</td><td colspan="2">Unsigned 16-bit integer.
Exchange's instructions and behavior control.
The bits are more thoroughly explained here.</td></tr>
<tr><td rowspan="11"></td><td rowspan="11"></td><th>Bit (mask)</th><th>Name/meaning</th></tr>
<tr><td>0 (0x0001)</td><td>Do "as write"</td></tr>
<tr><td>1 (0x0002)</td><td>Use handle from singulation</td></tr>
<tr><td>2 (0x0004)</td><td>XOR RN16.</td></tr>
<tr><td>3 (0x0008)</td><td>Transmit only; no reply expected.</td></tr>
<tr><td>4 (0x0010)</td><td>No TX CRC.</td></tr>
<tr><td>5 (0x0020)</td><td>No RX CRC.</td></tr>
<tr><td>6 (0x0040)</td><td>Use TX with CRC-5.</td></tr>
<tr><td>7 (0x0080)</td><td>No RX length; unknown.</td></tr>
<tr><td>8 (0x0100)</td><td>Strip handle from response packet.</td></tr>
<tr><td>9 (0x0200)</td><td>No tag re-selection (NUR05WL2, NUR10W).</td></tr>
<tr><td>2…3</td><td>Transmission (TX) length</td><td colspan="2">Unsigned 16-bit integer. The length is in bits.</td></tr>
<tr><td>4…5</td><td>Reception (RX) length</td><td colspan="2">Unsigned 16-bit integer. Tells the reader how many bits the command is expected to return.</td></tr>
<tr><td>6</td><td>RX timeout</td><td colspan="2">Time to wait for the tag's response in milliseconds. Range is 20…100. This applies to the actual custom command's RX time.</td></tr>
</table>

5 Read command's exchange control

In this example the required exchange control looks like this:

Byte(s)	Value HEX (actual)	More
0…1	02 01 (0x0102)	Flags: bit1 (0x0002) = use handle. Required by the read command specification thus handle is appended to the read command. bit8 (0x0100): = strip handle i.e. when the module sends back what the tag replied, the 2-byte (RN16) handle is removed from it.
2…3	1A 00 (0x001A)	TX length: 26 bits. See read command's structure.
4…5	50 00 (0x0050)	RX length: 80 bits. As per gen2 specification, the tag replies with the read contents appended with the 16-bit handle. Thus the length is: 8 bytes/4 words = 64 bits + 16 bits = 80 bits (0x0050).
6	14	Response timeout in milliseconds: 20.

6 Exchange data (read command explained)

As per Gen2 specification the read command consists of following parts which are presented per bit basis like this (*module handles these parts*):

	Command	Memory bank	Word address	Word count	*RN16 (handle)*	*CRC-16*
Number of bits	8	2	EBV	8	16	16

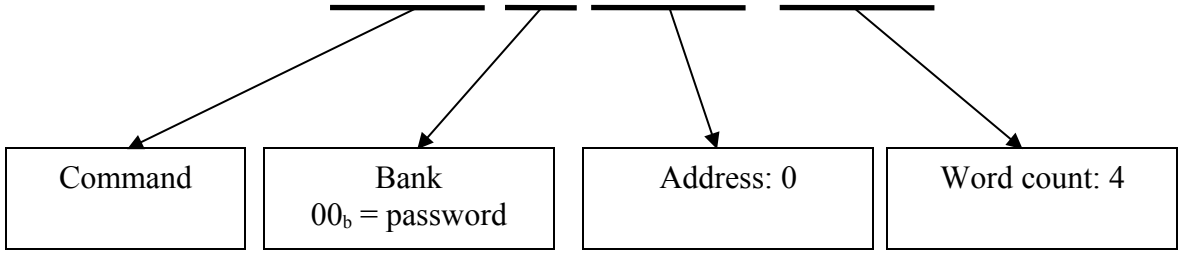
Where

Command	8-bit value 11000010_b (0xC2): "read"
Memory bank	2 bits: 00_b = password (reserved) memory 01_b = EPC memory 10_b = TID memory 11_b = user memory
Word address	Variable number of bits (Extensible Bit Vector, EBV). Here 8 bits: 00000000_b (0x00)
Word count	Number of 16-bit words to read. We read 8 (=4 words) bytes thus this value is 00000100_b (0x04).
RN16 (handle)	This is the handle the tag backscattered during its singulation. This is not appended during command build phase at the host side as it is a random number that the tag generates; the control flag "use handle" instructs the module to append it to the final transmitted bit stream.
CRC-16	This field is also appended by the modules TX part so this does not need to be taken care of otherwise.

7 Read command's bit level encoding

Like stated in the beginning, the bits are encoded from left to right. The split command is shown and explained below. As the read command's length is 26-bits then the required number of bytes to allocate the bit stream is 4 (4 x 8 = 32). These bytes are what is set to the transmitted data part of the custom exchange:

Actually transmitted read command				
Bytes HEX	C2	00	01	00
Bits	11000010	00000000	00000001	00000000
Combined	11000010 00 00000000 00000100 000000			



The 6 bits at the end are padding to byte alignment and are not transmitted.

8 Exchange part in whole

Combining the exchange control and its data part we get the exchange part in whole:

Byte(s)	Value HEX (actual)	More
0…1	02 01 (0x0102)	Flags: bit1 (0x0002) = use handle. Required by the read command specification thus handle is appended to the read command. bit8 (0x0100): = strip handle i.e. when the module sends back what the tag replied, the 2-byte (RN16) handle is removed from it.
2…3	1A 00 (0x001A)	TX length: 26 bits. See read command's structure.
4…5	50 00 (0x0050)	RX length: 80 bits. As per gen2 specification, the tag replies with the read contents appended with the 16-bit handle. Thus the length is: 8 bytes/4 words = 64 bits + 16 bits = 80 bits (0x0050).
6	14	Response timeout in milliseconds: 20.
7…10	C2 00 01 00	The transmitted bit buffer.

9 Final exchange protocol packet

Combining command, singulation and control parts the final protocol packet with actual byte addresses is (values in HEX, little-endian format):

Byte(s)	Value(s)	Meaning	More
0…5	A5 2700 0000 7D	Header	As specified by the NUR protocol. Packet length = 0x0027 (39, starting from command including CRC) CS = 0x7D
6	3F	Command	Command code for custom exchange (63).
	Common part (5 bytes total)		
7	02	Flags	Singulation and addressing flags. bit1 is set: common part is followed by singulation block using 32-bit addressing (as bit3 is not set).
8…11	00 00 00 00	Password	32-bit unsigned integer, not used as flags' bit0 is '0'.
	Singulation block (20 bytes total)		
12	13	Block length	Singulation block length (bytes to follow, 19).
13	01	Bank	Bank where the select is applied to; 01 = EPC.
14…17	20 00 00 00	Address	Unsigned 32-bit integer, bit address of the selection mask: 0x00000020 (32)
18…19	60 00	Length	Unsigned 16-bit integer, length of the selection mask in bits.
20…31	20 00 02 01 01 00 00 00 00 00 06 DB	Mask	12 bytes/96 bits, the selection mask data. In this case the tag's EPC.
	Custom exchange part (11 bytes total)		
32…33	02 01	Flags	The custom exchange control flags: 0x0102.
34…35	1A 00	TX length	Transmission length in bits: 0x001A (26).
36…37	50 00	RX length	Expected response length in bits: 0x0050 (80).
38	14	Timeout	Response wait timeout: 0x14 (20ms).
39…42	C2 00 01 00	TX buffer	Transmitted bit buffer, 4 bytes as explained in bit level encoding.
	Packet CRC (2 bytes)		
43…44	5B D1	CRC-16	16-bit unsigned integer, packet CRC starting from command byte and calculated to TX buffer end.

The command in 4-byte pieces is:

```
A5 27 00 00   00 7D 3F 02   00 00 00 00   13 01 20 00
00 00 60 00   20 00 02 01   01 00 00 00   00 00 06 DB
02 01 1A 00   50 00 14 C2   00 01 00 5B   D1
```

10 Response to the exchange command

The password memory of the tag used in this example was pre-programmed to this (HEX):

```
AC DC AB BA DE AD BE EF
```

The read command implemented as custom exchange then produces the following reply:

Byte(s)	Value(s)	Meaning	More
0…5	A5 0C00 0000 56	Header	As specified by the NUR protocol. Packet length = 0x000C (12, starting from command including CRC) CS = 0x56
6	3F	Command echo	Command code for custom exchange (63).
7	00	Status	0 = OK
8…15	AC DC AB BA DE AD BE EF	Data	The memory bank contents excluding tag appended the handle (control flag “strip handle” = ‘1’).
16…17	05 10	CRC	Unsigned 16-bit integer, packet’s CRC-16: 0x1005.

The response in 4-byte pieces is:

```
A5 0C 00 00   00 56 3F 00   AC DC AB BA   DE AD BE EF
05 10
```

11 Custom exchange control flags explained

11.1 Bit 0, mask 0x0001: “as write”

This flag control the receiver behavior when the response is received from the tag. Typically write or “write like” operations take a little longer to execute. This flag causes some attenuation the receiver end in order to be less susceptible to possible interferences. Due to this it must be taken into account that the distance from which the reader can still reliably execute the operation reduces.

11.2 Bit 1, mask 0x0002: use/append handle

This flag is required to be ‘1’ in practically every command that expects a response. As the tag is only singulated by the singulation block there is then no other means to communicate with the tag; the handle sort of “addresses” the tag that currently is ready to communicate with the reader.

11.3 Bit 2, mask 0x0004: XRX RN16

In the Gen2 protocol definition, the write command requires the reader to get a new RN16 from the tag in prior to the actual write operation. Setting this flag to ‘1’ mimics that behavior in other “write like” operations too. The requirement is that the actual transmitted data is 16 bits in length so that an exclusive OR operation can be done with the received 16-bit RN.

11.4 Bit 3, mask 0x0008: TX only

Like the name states: transmit the buffer only, but don’t wait for a response. Usable in some multi-part command sequences where the tag may needs additional instructions between commands.

11.5 Bit 4, mask 0x0010: no TX CRC

When set to ‘1’, this flag causes the reader to leave the CRC out in the transmission phase.

11.6 Bit 5, mask 0x0020: no RX CRC

When set to ‘1’, the reader does not decode the (possibly) received CRC-16; it is placed into the reception buffer as a part of the “payload” that the response has.

Caution: if, by any reason, the module FW needs to locate the response handle from the bytes it receives, this may cause some very irrational behavior.

11.7 Bit 6, mask 0x0040: TX with CRC-5

If CRC is used in TX phase then setting this bit to '1' causes the transmitter to use CRC-5 instead of CRC-16.

11.8 Bit 7, mask 0x0080: no RX length

If the reception length is unknown, then this bit should be set to '1' as well as the RX length field in the control block to 0.

Caution I: as the module's receiver does not understand the concept of "receive 0 bits" it is then set to maximum value of 1023 bits. This then causes a situation where the FW wait for the response and the tries to locate the handle that the tag appends to the response. As the uncertainties are obvious, this may cause irrational behavior and unexpected responses to the host from he module.

Caution II: if you are a UHF RFID chip designer and plan to ignore the tag handle in cases where the response length can vary then don't. Just don't. Such an approach to work correctly would always require special HW and SW in order to work correctly and as such does not serve any Greater Purpose™ of UHF readers to be more generically applied to practice.

11.9 Bit 8, mask 0x0100: strip handle

As the commands are appended with the handle/RN16 value last used to execute the command, this bit controls whether the handle is kept ('0') and sent back to the host or not ('1'). In this example's read command it is tripped as it is not of interest.

Example: the tag backscatters data `AA BB CC DD` and handle **`11 22`**. If this flag is '1' then the data part in the custom exchange command is `AA BB CC DD`. If the flag is '0' then the data is `AA BB CC DD` **`11 22`**.

11.10 Bit 9, mask 0x0200: no re-select (NUR05WL2/NUR10W)

Setting this flag to '1' causes the exchange operation to skip tag singulation. This is usable when the carrier of the module is commanded to be on between multi-part command's different parts where the parts depend on each other and as such no-reselection is required (communication is based on the existing handle). Usually this flag is combined with the "strip handle" flag (being set to '0') so that the handle is received by the host and then appended to following command's bit stream. In such a case the "use handle" in the latter is set to '0' (as the handle already is there).

12 Command parts as sample C-structures

In these samples it is expected that the host can handle memory addressing with 16- or 32-bit variables from addresses not aligned 2 or 4. Thus, the structures are packed (one byte alignment). The packing depends naturally on compiler. Some approaches are the "__packed" keyword, __attribute__((packed)) or Microsoft's #pragma pack(push, 1)/#pragma pack(pop) –pair.

For clarity, the "_packed" keyword is assumed to be available. The command byte is inserted before the common block and the variable length data is taken into account when accessing the packet byte buffer.

```
/* Protocol defined command header. */
struct __packed NUR_CMDHDR
{
  uint8_t start;    /* 0xA5 */
  uint16_t length; /* Command + parameters + CRC-16. */
  uint16_t flags;  /* From host, not used. Set to 0.*/
  uint8_t cs;     /* Over previous 5 bytes. */
};

/* Block that defines the addressing modes and password
usage. */
struct __packed NUR_COMMONBLOCK
{
  uint8_t flags;  /* Addressing and password usage. */
  uint32_t pwd;   /* Password value. */
};

/* Block that defines tag selection parameters. */
struct __packed NUR_TAGSELBLOCK
{
  uint8_t size;       /* Number of bytes to follow. */
  uint8_t bank;       /* Selection bank. */
  uint32_t bitAddr;  /* 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 the custom exchange (CX) command
behavior. */
struct __packet NUR_CXBLOCK
{
  uint16_t flags;     /* Control flags. */
  uint16_t txLen;     /* TX length in bits. */
  uint16_t rxLen;     /* RX length in bits. */
  uint8_t tmo;        /* Timeout in ms, 20.100.. */
  uint8_t txData[1]; /* Variable length TX bit buffer. */
};
```