TECHNOLOGY SOLUTIONS UK LTD

APPLICATION NOTE:
MODIFYING THE EPC OF A TRANSPONDER USING THE ASCII 2 PROTOCOL

www.tsl.uk.com
Design • Development • Manufacture

CONTENT

History

Version	Date	Modifications
1.0	04/09/2015	First Release

INTRODUCTION

It is often necessary to change the existing EPC of a tag to a new value to support either standards-based or custom encoding formats. This document provides examples of how to modify the EPC identifier using the TSL ASCII 2 protocol. The ASCII 2 protocol is described in detail in the document *TSL ASCII Protocol 2.x* available from the Technology Solutions (UK) Ltd website[1].

EPC MEMORY BANK RECAP

Before looking at the EPC programming process it will be useful to review the layout of the tag's EPC memory bank. The EPC memory bank holds the Electronic Product Code (EPC), the Protocol Control (PC) word and the 16-bit cyclic redundancy check (CRC). The PC word is one sixteen bit word and defines the length as well as other properties of the EPC. The EPC is the actual identifier for the transponder (it can be many words in length). The CRC is a single word and is the checksum calculated across the PC and EPC. When stored in the EPC memory bank the first word is the CRC, the second word is the PC and the remainder of the memory bank stores the EPC. In an inventory response, the amount of the memory bank returned for the EPC is determined by the PC. Figure 1 illustrates the EPC memory banks logical structure.

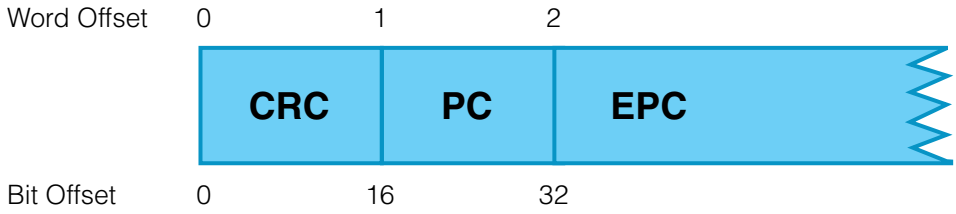

FIGURE 1 EPC MEMORY BANK - LOGICAL LAYOUT

PROCESS OVERVIEW

To modify the EPC of a specific, UHF tag requires writing the correct information to the tag's EPC memory bank. When multiple tags are in range of the reader (a common scenario) then it is essential to ensure that only the correct tag is modified. The following procedure is designed to work with any standards compliant UHF tag.

1. Isolate the target tag - where possible
2. Identify the existing EPC of the tag to be modified
3. Determine if the tag's Protocol Control (PC) word needs to be modified
   a. Calculate the new value of the PC word
4. Write the new data to the target tag's EPC memory bank

The steps are described in further detail below and will use an *example tag* with an initial 96-bit EPC value of `111122223333444455556666`.

[1]See the downloads page for the reader you are using e.g. https://www.tsl.com/downloads/rfid-readers/ultra-high-frequency/1128-bluetooth-uhf-rfid-reader/

ISOLATE THE TARGET TAG

Care needs to be taken when writing to a tag to ensure that only the desired tag is modified. Part of the solution is to physically isolate other, nearby tags from the Reader's field. This can be accomplished by placing the other tags within a closed metal box or tin. Ideally the target tag will now be the only tag within range of the Reader. However, this physical isolation is not always practical so we need to also take steps to ensure that the commands used will be directed only to the target tag. This selective writing to the target tag is described in the steps below.

IDENTIFYING THE EPC OF THE TARGET TAG

The existing EPC of the tag to be modified (the *target tag*) can be determined in one of two ways:

1. Some tags are supplied with the existing EPC already printed on the tag in hexadecimal text and often with the same information also in barcode form.
2. The Reader can be used to perform an inventory of nearby tags and obtain the target tag's current EPC. Using a reduced power inventory can help eliminate more distant tags but care must be taken to ensure that the correct is identified.

DETERMINE IF THE PROTOCOL WORD SHOULD BE MODIFIED

The EPC Global UHF Gen 2 Air Interface Protocol supports EPC identifiers of variable length with the Protocol Control word of the EPC memory bank being used to specify how many words (starting from offset 2) will be reported as the EPC value - 6 words (96-bits) is a typical value.

If the tags you are using already come with the same length EPC as your new EPC values then you will only need to modify the EPC value itself. This is fairly straight-forward and is described in detail in the next section.

If you wish to change the length of the reported EPC then a little more work needs to be done as the PC word (at offset 1) contains other information that needs to be preserved when the length bits are modified. The length bits are the 5 most significant bits of the PC word (as reported by the ASCII protocol) so for our example tag, which has a 96-bit EPC, a typical PC word might be (length bits in **bold**):

`3000` hex **`0011 0`**`000 0000 0000` binary **`00110`** = 6 words i.e. 96-bits

To change this tag to report a 64-bit EPC we would specify 4 words as the new length resulting in a new PC of:

`2000` hex **`0010 0`**`000 0000 0000` binary **`00100`** = 4 words i.e. 64-bits

If we were to write this new PC value into the EPC memory bank (at offset 1) of the *example tag* we would now see the EPC value as `1111222233334444`.

So, to change the length of the EPC requires the following actions

1. Read the existing PC word at offset 1 of the EPC memory bank
2. Modify the top 5 bits to encode the new number of words in the EPC value
3. Write the PC word back to offset 1 of the EPC memory bank – this can be combined with new data for the value of the EPC (as will be shown below)

WRITING THE NEW VALUE TO THE EPC MEMORY BANK

The ASCII 2 protocol provides a very robust command for writing to a single tag called *Write Single Transponder* (`.ws`). This command implements a writing algorithm which makes multiple retries to write to the tag so is highly recommended and will be used in all of the examples presented here.

So, the target tag has been isolated as best you can physically, the existing EPC value is known and, if needed, the new value for the PC word has been calculated. Now we need to construct the command to actually write the new data to the tag. As mentioned above, we also need to ensure the command is only received by the target tag and we do this by using the command's *Select* parameters.

SELECTING THE TARGET TAG

The *Write Single Transponder Command* requires sufficient data to *select* a single transponder and we need to provide:

- Data to compare with the tag memory
- The memory bank to compare against
- The offset into the memory bank where the comparison starts (specified in bits)
- The length of data to use in the comparison (specified in bits)

To ensure we only operate on the *target tag* we will provide the full, existing EPC value as follows:

`-sd 111122223333444455556666`	the full EPC value
`-sb epc`	compare against the EPC memory bank
`-so 0020`	start at 20h i.e. word 2 of the bank
`-sl 60`	match 60h data bits i.e. 96-bits

With these parameters only a tag with the EPC value of `111122223333444455556666` will respond. It is common with new tags that they are supplied with identical EPCs in which case the physical isolation of the target tag becomes crucial. If more than one tag responds then it is possible that the write command will succeed but you will not know which of the tags has been written without further testing. An alternative approach to selecting the target tag, that requires specific tag support, is detailed below and can be used with tags that have identical default EPCs.

MODIFYING THE EPC WITHOUT CHANGING ITS LENGTH

If the length of the EPC is not changing then we can simply write the new values into the appropriate memory locations. We do this by supplying:

- The data to be written
- The memory bank to write to
- The memory bank offset, in 16-bit words, at which to write
- The length of data to write in 16-bit words

Example 1: Modify the EPC to have the value `FFFFEEEEDDDDCCCCBBBBAAAA`.

The new EPC value is entirely different from the existing value so we will re-write the complete EPC specifying the data as follows:

`-da FFFFEEEEDDDDCCCCBBBBAAAA`	the new value for the EPC
`-db epc`	write to the EPC memory bank
`-do 0002`	start at **word** 2
`-dl 06`	write 6 **words**

To complete the command we will specify that the reader uses default values for all parameters that we do not explicitly specify by using the parameter reset option `-x`. The full command will then be:

```
.ws -x -da FFFFEEEEDDDDCCCCBBBBAAAA -db epc -do 0002 -dl 06 -sd
111122223333444455556666  -sb epc -so 0020 -sl 60
```

Executing this command using a terminal or via ASCII Protocol Explorer will give the following output when successful:

```
CS: .ws -x -da FFFFEEEEDDDDCCCCBBBBAAAA -db epc -do 0002 -dl 06 -sd
111122223333444455556666  -sb epc -so 0020 -sl 60
EP: 111122223333444455556666
WW: 6
ME: Write Success
OK:
```

Notice that the response includes the old value of the EPC - the new value is not available until the command has completed.

Example 2: Modify the EPC to have the value `111122223333444455550000`.

The new EPC value only requires that the last word be changed so we can do this by only writing the changed data - this will be slightly faster than re-writing the complete EPC:

`-da 0000`	the new value for the last word of the EPC
`-db epc`	write to the EPC memory bank
`-do 0007`	start at **word** 7
`-dl 01`	write 1 **word**

To complete the command we will specify that the reader uses default values for all parameters that we do not explicitly specify by using the parameter reset option `-x`. The full command will then be:

```
.ws -x -da 0000 -db epc -do 0007 -dl 01 -sd 111122223333444455556666  -sb epc -so 0020
-sl 60
```

Executing this command using a terminal or via ASCII Protocol Explorer will give the following output when successful:

```
CS: .ws -x -da 0000 -db epc -do 0007 -dl 01 -sd 111122223333444455556666  -sb epc -so
0020 -sl 60
EP: 111122223333444455556666
WW: 1
ME: Write Success
OK:
```

Example 3: Modify the EPC to have the value `FFFFEEEEDDDDCCCC`.

The new EPC value is shorter (64-bits) and different to the existing value so we need to determine the current PC word, modify it to include the new length and then rewrite the PC word and 4 words of the new EPC value in one command.

We can use a *Read Transponder Command* (`.rd`) to obtain the current value of the PC Word from EPC memory bank offset 1:

```
.rd -x -db epc -do 0001 -dl 01 -sd 111122223333444455556666 -sb epc -so 0020 -sl 60
```

The response will contain the PC word value in the `RD:` line:

```
CS: .rd -x -db epc -do 0001 -dl 01 -sd 111122223333444455556666 -sb epc -so 0020 -sl
60
EP: 111122223333444455556666
RD: 3000
OK:
```

Modifying the top 5 bits to represent 64-bits i.e. 4 words we get the new PC value of `2000` hex (See section *Determine if the Protocol Word Should be Modified* for more details.

We can now join this to the new EPC value to specify a single data block to be written as follows:

`-da 2000FFFFEEEEDDDDCCCC`	the new value for the PC word and EPC value
`-db epc`	write to the EPC memory bank
`-do 0001`	start at **word** 1
`-dl 05`	write 5 **words**

Notice that this write now starts at offset 1 where the PC word resides. To complete the command we will specify that the reader uses default values for all parameters that we do not explicitly specify by using the parameter reset option `-x`. The full command will then be:

```
.ws -x -da 2000FFFFEEEEDDDDCCCC-db epc -do 0001 -dl 05 -sd 111122223333444455556666
-sb epc -so 0020 -sl 60
```

Executing this command using a terminal or via ASCII Protocol Explorer will give the following output when successful:

```
CS: .ws -x -da 2000FFFFEEEEDDDDCCCC-db epc -do 0001 -dl 05 -sd
111122223333444455556666  -sb epc -so 0020 -sl 60
EP: 111122223333444455556666
WW: 5
ME: Write Success
OK:
```

The tag will now have the 64-bit EPC value of `FFFFEEEEDDDDCCCC`.

SELECTING THE TARGET TAG USING THE TID MEMORY BANK

Sometimes an alternative way of uniquely identifying a tag is possible - the TID memory bank of certain types of tag can contain a serialised id value. If the tags you are using have serialised TIDs then we recommend that these are used as the select criteria. This approach has the benefit that the tag is not being selected using the EPC so that if the write fails and (potentially) leaves the EPC in an unknown state we can simply reissue the same command to attempt the write again.

To use the TID memory bank you first need to consult the tag manufacturer's data sheet to determine how much TID memory the tag has then the following modified procedure can be used:

1. Identify the existing EPC of the tag to be modified
2. Read the TID memory
3. Determine if the tag's Protocol Control (PC) word needs to be modified
   a. Calculate the new value of the PC word
4. Write the new data to the target tag's EPC memory bank using the TID memory bank to select the target transponder

Example 4: Modify the EPC to the value `FFFFEEEEDDDDCCCCBBBBAAAA` using TID select method.

For this example we will simply modify the EPC value without changing its length.

We can use a *Read Transponder Command* (`.rd`) to obtain the TID value for our select parameters - we will read all of the TID bank using the length obtained from the tag data sheet - in this example it is 6 words long:

```
.rd -x -db tid -do 0000 -dl 06 -sd 111122223333444455556666 -sb epc -so 0020 -sl 60
```

The response will contain the TID value in the `RD:` line:

```
CS: .rd -x -db tid -do 0000 -dl 06 -sd 111122223333444455556666 -sb epc -so 0020 -sl
60
EP: 111122223333444455556666
RD: E2801130200029B36C40000B
OK:
```

Using this value as part of the select criteria we can write the new EPC value knowing that this uniquely serialised tag will be the only tag that responds to the command. Furthermore, in the event of the write failing we can simply re-issue the command to try again.

To ensure we only operate on the *target tag* we will select on the full TID value as follows:

`-sd E2801130200029B36C40000B`	the full TID value
`-sb tid`	compare against the TID memory bank
`-so 0000`	start at 0h
`-sl 60`	match 60h data bits i.e. 96-bits

The full command will then be:

```
.ws -x -da FFFFEEEEDDDDCCCCBBBBAAAA -db epc -do 0002 -dl 06 -sd
E2801130200029B36C40000B -sb tid -so 0000 -sl 60
```

Executing this command using a terminal or via ASCII Protocol Explorer will give the following output when successful:

```
CS: .ws -x -da FFFFEEEEDDDDCCCCBBBBAAAA -db epc -do 0002 -dl 06 -sd
E2801130200029B36C40000B -sb tid -so 0000 -sl 60
EP: 111122223333444455556666
WW: 6
ME: Write Success
OK:
```

The tag will now have the 64-bit EPC value of `FFFFEEEEDDDDCCCCBBBBAAAA`.

SUMMARY

A basic outline of the procedure for successfully writing to the EPC of a tag was presented. Also a detailed description of how to change the EPC, both with and without changing the length of the EPC, was provided. We have seen that a combination of physical isolation and use of appropriate *Select* criteria can ensure that only the desired tag responds to a write command. In addition, for some types of tag, an even more robust method can be used that provides better support should the write fail. Finally, all the examples were presented using just a few commands from the TSL ASCII 2 protocol, in particular it demonstrated the power of the *Write Single Transponder Command* to modify a tag's EPC value.

ABOUT TSL

ABOUT

TSL designs and manufactures both standard and custom embedded, snap on and standalone peripherals for handheld computer terminals. Embedded technologies include:

- RFID - Low Frequency, High Frequency & UHF
- *Bluetooth*® wireless technology
- Contact Smartcard
- Fingerprint Biometrics
- 1D and 2D Barcode Scanning
- Magnetic Card Readers
- OCR-B and ePassport

Utilizing class leading Industrial design, TSL develops products from concept through to high volume manufacture for Blue Chip companies around the world. Using the above technologies TSL develops innovative products in a timely and cost effective manner for a broad range of handheld devices.

CONTACT

Address:	Technology Solutions (UK) Limited, Suite A, Loughborough Technology Centre, Epinal Way, Loughborough, Leicestershire, LE11 3GE. United Kingdom.
Telephone:	+44 1509 238248
Fax:	+44 1509 214144
Email:	enquiries@tsl.uk.com
Website:	www.tsl.com

ISO 9001: 2008