Technology
Solutions (UK) Ltd

APPLICATION NOTE:
TSL ASCII 2 PROTOCOL - SELECTING, READING AND WRITING TRANSPONDERS

CONTENT

History

Version	Date	Modifications
1.0	25/08/2013	First Release for ASCII Protocol 2.1
1.1	21/08/2013	Revised Memory bank description, Added command breakdown to Write example. Spelling corrections and formatting
1.2	28/11/2013	Added notes section and note on large memory access
1.3	16/12/2014	Expanded the Examples section to include reading, writing and locking of transponders

INTRODUCTION

The aim of this document is to recap the behaviour and operation of UHF transponders with respect to selecting, reading and writing and to provide some examples of how to achieve this with the TSL ASCII 2 Protocol.

For more information about the behaviour of the transponders refer to the standard describing their behaviour. The latest UHF Class 1 Gen 2 Standard is available for download from http://www.gs1.org/gsmp/kc/epcglobal/uhfc1g2

“The RF in RFID” by Daniel M. Dobkin (ISBN 978-0-7506-8209-1) is another useful reference.

UHF TRANSPONDERS RECAP

MEMORY BANKS

A UHF Transponder has up to four memory banks; *Reserved, EPC, TID* and *User*. Transponder memory is accessed as sixteen bit words addressed from the start of the memory bank.

The *Reserved* memory bank is 64 bits long and contains the kill password (bits 0x00 to 0x1f) and access password (bits 0x20 to 0x3f)

The *EPC* memory bank holds the *Electronic Product Code (EPC)*, the *Protocol Control (PC)* word and the 16-bit cyclic redundancy check *(CRC)*. A typical transponder inventory response contains the *PC* word, the *EPC* and the *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*. Refer to section 6.3.2.1.2 of the standard for more information.

The *Transponder identifier* bank *(TID)* holds 64 bits of information to help identify the type of transponder. The standard defines a number of schemes how this memory can be formatted (refer to section 6.3.2.1.3 of the standard). The most useful format for uniquely identifying transponders is where the transponder type is serialized in this bank to provide the silicon identifier and a unique serial number.

The *User* memory bank is used for storing application specific information.

INVENTORY AND TAG ACCESS

A *tag access* operation is a read / write / lock / kill / block read / block write etc… and may require an access and or kill password.

When the reader attempts to communicate to a field of transponders it performs an *inventory*. An *inventory* involves issuing a request to all transponders and then providing time slots for the transponders to reply in. The number of slots is determined by the Q value (2^Q).

When performing an *inventory* the transponder only replies with its inventory response. When performing a *tag access* the reader performs a further set of operations to access the transponder in each slot before issuing the next slot request for the next transponder.

Whether the transponder responds at all (assuming it is in range) is determined by the query issued by the reader for the *inventory*. This identifies the required transponders based on the states of a number of flags maintained by each transponder.

Before performing an *inventory* the reader can perform a *select*. This is a command to all transponders within range of the reader and instructs transponders that match and not-match a particular criteria to set their flags into a specific known state.

The *select* separates the field of transponders into distinct groups which is then followed by an inventory or tag access which operates on a particular group.

SESSIONS AND SELECTED FLAGS

Each transponder maintains five flags; the *selected* flag and four *session* flags (*session 0, session 1 session 2, session 3*). As described above these flags are set into particular states by issuing a *select* and then used to identify the transponders of interest for an *inventory* or *tag access* (refer to section 6.3.2.2 of the standard).

The *selected* (SL) flag is either *selected* or *not selected*.

The *session* flags have a value or either *A* or *B*. As a transponder responds to an *inventory* (which may be part of a *tag access*) the flag in the transponder, used to identify the transponder as part of this *inventory*, toggles state. For example if the *inventory* was for transponders in *session 0* flag state *A,* each transponder that responds to the inventory will change its *session 0* flag to *B* from *A* as it responds.

Each *session* and *selected* flag has a particular persistence, that is the amount of time the transponder can maintain the selected state while either energized (in the reader field) or not energized (outside the reader field). These are summarised in the table below:

Flag	Action
S0 inventoried flag	Tag energized: Indefinite Tag not energized: None
S1 inventoried flag	Tag energized: 500ms < persistence < 5s Tag not energized: 500ms < persistence < 5s
S2 inventoried flag	Tag energized: Indefinite Tag not energized: 2s < persistence
S3 inventoried flag	Tag energized: Indefinite Tag not energized: 2s < persistence
Selected (SL) flag	Tag energized: Indefinite Tag not energized: 2s < persistence

SESSIONS AND INVENTORY

The following are provided as examples of how the different persistence of the session may be useful. Note that the ASCII 2 protocol provides an 'inventory only' flag for the inventory command that means that a select does not have to be performed for every inventory command executed:

Session 0 is only persistent while the reader field is energized. The field is reset for each reader command. Transponders revert to the *A* state for sessions after the persistence expires. Therefore performing an *inventory* of *session 0*, flag state *A* transponders provides the best chance that all transponders will respond to every *inventory*.

Session 1 has a short persistence. Performing a session 1 *inventory* will return and temporarily silence transponders. This is useful with large populations of transponders to temporarily silence transponders, once seen a transponder won't respond again for a short while. An analogy may be telling the loud ones to be quiet to allow the quiet ones to speak.

Session 2 can be used for counting situations as a transponder remains in its new state for some time. The transponders are selected into a known state (e.g. *session 2, A*). Perform inventories for this query target (e.g. *session 2, A*) until no more transponder respond. All transponders have toggled state (e.g. to *session 2, B*) when no more transponders respond. Perform more inventories this time for the opposite query target (e.g. *session 2, B*) until no more transponders respond. The result is two lists of transponders that should be the same, each transponder only responding to an inventory twice.

TRANSPONDER ACCESS

To specify the transponder(s) to *inventory* or perform a *tag access* operation requires two things; Firstly to specify and execute a *select* to group the required transponder(s) separately from the rest of those in the field, secondly to perform the *inventory* or *tag access* querying for the appropriate group of transponders.

SPECIFYING THE GROUP OF TRANSPONDERS TO *SELECT*

Transponders are grouped based on the contents of their memory banks. For each of the commands where *select* is used a *Select Bank*, *Select Offset, Select Data* and *Select Length* are specified.

The *Select Bank* specifies which memory bank of the transponder is used for the comparison.

The *Select Offset* specifies the offset in bits into the memory bank at which to start matching the *Select Data*.

The *Select Data* specifies the actual bit pattern to match to the transponders memory starting at the specified offset. The data must be specified as a whole number of bytes. The *Select Length* permits partial bytes.

The *Select Length* specifies the number of bits in the *Select Data* to match to the memory in the specified transponder bank starting from the *Select Offset*.

The above defines two distinct groups of transponders; the transponders where the memory matches the specified pattern at the specified location in its memory and those transponders that do not. The *Select Action* and *Select Target* parameters specify how to change the state of the transponders so that the two groups (matching and non-matching) may be identified.

The *Select Target* specifies which of the transponder's session or selected flags to modify according to the *Select Action.*

The *Select Action* specifies how to modify the specified flag. The actions are summarised in the table below and describes how the selected or session flag (based on the *Select Target*) is modified depending on whether the transponder matches or does not match the specified pattern.

Parameter n=	Matching Action		Non Matching Action	
	SL Flag	Session Flag	SL Flag	Session Flag
0	assert	Set A	deassert	Set B
1	assert	Set A	nothing	nothing
2	nothing	nothing	deassert	Set B
3	toggle	toggle	nothing	nothing
4	deassert	Set B	assert	Set A
5	deassert	Set B	nothing	nothing
6	nothing	nothing	assert	Set A
7	nothing	nothing	toggle	toggle

SPECIFYING THE GROUP OF TRANSPONDERS TO RETURN

With the transponders separated into groups (via known *session* and selected flag states) the *query* parameters (*Query Select, Query Session* and *Query Target)* specify which group to return.

A transponder will respond to an *inventory* (which may be part of a *tag access*) if both of the following are true:

- its selected flag matches the *Query Select* value
- the session flag specified by *Query Session* matches the state specified in *Query Target*

The *Query Select* can be set to *all* which effectively allows the state of the *selected* flag to be ignored.

NOTE: When a transponder responds to an *inventory* it toggles the flag for the specified *session* (i.e. *B* to *A* or *A* to *B* for the query *session*).

SELECTING THE Q VALUE AND ALGORITHM

When a reader performs an *inventory* it sends out the *query* and then provides a number of time slots for the transponders to respond within. The number of time slots used is determined by the *Q* value, there are 2^Q timeslots which gives a range from 1 to 32768 time slots. Providing too many time slots for the number of expected transponders does not cause any problems except to slow down the command as the reader will issue many timeslots where there are no transponders to reply.

The *fixed Q* algorithm uses the *Q* value directly and does not alter it for subsequent inventories. The *dynamic Q* algorithm uses the *Q* value parameter as a starting value and adapts the Q for subsequent operations based on the number of transponders that replied in the previous inventory.

EXAMPLES

MODIFYING A SINGLE TRANSPONDER

These examples demonstrate writing to, reading from and locking a single, unique transponder. To identify a single transponder, typically, the *EPC* returned from an *inventory* is used and is assumed to be unique. *Select* parameters are used that specify the full *EPC* of the target transponder to restrict the operation to just that target tag. These examples modify the transponder's User memory bank.

Transponder memory is address by word or bit. Each word has 16 bits. The *EPC* is stored from word 2 of the *EPC* memory bank (bit address 32). So, for a typical 96 bit *EPC* the value is 6 words long. Transponders are likely to be in the default state A for a session, unless they have been recently energised, so it is recommended to use state B for writing.

WRITING

The ASCII 2 protocol provides two commands for writing to transponders. This example uses the more flexible of the two the Write Transponder Command - *.wr*. This command gives the developer full control over the query and selection criteria, the Q algorithm and Q Value.

Task:

Write words 0x1111222233334444 to the *User* bank at word address 5 of a transponder with *EPC* 0x0123456789abba9876543210 using *session 1*.

Command:

```
.wr -db usr -da 1111222233334444 -dl 04 -do 0005 -ql all -qs s1 -qt b -sa 4 -sb
epc -sd 0123456789abba9876543210 -sl 60 -so 0020 -st s1
```

Command Breakdown:

.wr	Execute the Write transponder command
–db usr	Data will be written to the User memory bank
–da 1111222233334444	The data to write in 4-character, ASCII-hex words
–dl 04	4 Words will be written
–do 0005	The data will be written starting at offset 5
–ql all	Transponders with any *selected* flag state can respond
–qs s1	Transponders will respond based on the value of the *session 1* flag
–qt b	Transponders with (*session 1*) state *B* will respond
–sa 4	Transponders that *match* the *Select* criteria will clear the *selected* flag and set the *session 1* flag to *B*. Transponders that *do not match* the *Select* criteria will set the *selected* flag and set the *session 1* flag to *A*.
–sb epc	Select transponders based on the *EPC* memory

–sd 0123456789abba9876543210	Select using this data i.e. the *EPC* value
–sl 60	Select using 96 bits of the data (length is in hex, 0x60 = 96)
–so 0020	Select from memory bit offset 32 (0x20 = 32 bits)
–st s1	Select will affect the *session 1* flag state

Response:

```
CS: .wr -qlall -qss1 -qtb -sa4 -sbepc -sd0123456789abba9876543210 -sl60 -so0020
-sts1 -dbusr -dl04 -do0005 -da1111222233334444
EP: 0123456789ABBA9876543210
WW: 4
OK:
```

The command's response provides the command executed after the CS: header and then for each responding transponder provides the EPC (after the EP: header) and the number of words written to that transponder (following the WW: header). The above response shows a single transponder responding and all 4 words were written - a successful operation.

The *.wr* command can write to multiple transponders simultaneously. If the *select* criteria can match multiple transponder EPCs or the EPC specified is not unique then the command will attempt to write the data to all matching transponders. To ensure that only a single transponder is written to use the *.ws* command. This command is very robust and makes multiple attempts to write the specified data before failing. It also guarantees that only one transponder is written to. If the *select* criteria specifies a match for more than one transponder in range then the *.ws* command is likely to succeed but be aware that there is no simple way of knowing which of the transponders was written to.

READING

Having successfully written to the transponder the contents of the User memory can be verified using the Read Transponder Command - *.rd*.

Task:

Read 4 words from the *User* bank at word address 5 of a transponder with *EPC* 0x0123456789abba9876543210 using *session 1*.

Command:

```
.rd –db usr–dl 04 –do 0005 –ql all –qs s1 –qt b –sa 4 –sb epc –sd
0123456789abba9876543210 –sl 60 –so 0020 –st s1
```

Command Breakdown:

.rd	Execute the Read transponder command
–db usr	Data will be read from the User memory bank
–dl 04	4 Words will be read
–do 0005	The data will be read starting at offset 5
–ql all	Transponders with any *selected* flag state can respond
–qs s1	Transponders will respond based on the value of the *session 1* flag
–qt b	Transponders with *(session 1*) state *B* will respond
–sa 4	Transponders that *match* the *Select* criteria will clear the *selected* flag and set the *session 1* flag to *B*. Transponders that do not *match* the *select* criteria will set the *selected* flag and set the *session 1* flag to *A*.
–sb epc	Select transponders based on the *EPC* memory
–sd 0123456789abba9876543210	Select using this data i.e. the *EPC* value
–sl 60	Select using 96 bits of the data (length is in hex, 0x60 = 96)
–so 0020	Select from memory bit offset 32 (0x20 = 32 bits)
–st s1	Select will affect the *session 1* flag state

Response:

```
CS: .rd -qlall -qss1 -qtb -sa4 -sbepc -sd0123456789abba9876543210 -sl60 -so0020
-sts1 -dbusr -dl04 -do0005
EP: 0123456789ABBA9876543210
RD: 1111222233334444
OK:
```

The command's response provides the command executed after the CS: header and then for each responding transponder provides the EPC (after the EP: header) and the data read from that transponder (following the RD: header). The above response shows a single transponder responding with the 4 hex words 1111222233334444 - a successful operation.

LOCKING

When a transponder has been prepared with the desired data then it can be secured to restrict further modification. In this example we will use the Lock Command - *.lo* to require an access password to make further modifications to the 'locked' User memory bank. Care must be taken when experimenting with the lock command as it is possible to make a transponder permanently read-only. The use of this command requires a full understanding of the 20-bit lock payload as described in the UHF Class 1 Gen 2 Standard .

Before using the lock command the access password needs to be set – this uses a write operation as described in the 'WRITING' section above. We will use *ddddcccc* as the access password.

Task:

Set an access password of *ddddcccc*.

Command:

```
.wr -db res -dl 02 -do 0002 -da ddddcccc -ql all -qs s1 -qt b -sa 4 -sb epc -sd
0123456789abba9876543210 -sl 60 -so 0020 -st s1
```

Task:

Lock the User memory bank to prevent further modification unless the access password is specified.

Command:

```
.lo -lp 00c02 -ap ddddcccc -ql all -qs s1 -qt b -sa 4 -sb epc -sd
0123456789abba9876543210 -sl 60 -so 0020 -st s1
```

Command Breakdown:

.lo	Execute the Lock command
–lp 00c02	20-bit Lock payload specifying that the User memory bank requires an access password to be written to.
–ap ddddcccc	The access password
–ql all	Transponders with any *selected* flag state can respond
–qs s1	Transponders will respond based on the value of the *session 1* flag
–qt b	Transponders with (*session 1)* state *B* will respond
–sa 4	Transponders that *match* the *Select* criteria will clear the *selected* flag and set the *session 1* flag to *B*. Transponders that *do not match* the select criteria will set the *selected* flag and set the *session 1* flag to *A*.
–sb epc	Select transponders based on the *EPC* memory
–sd 0123456789abba9876543210	Select using this data i.e. the *EPC* value
–sl 60	Select using 96 bits of the data (length is in hex, 0x60 = 96)
–so 0020	Select from memory bit offset 32 (0x20 = 32 bits)
–st s1	Select will affect the *session 1* flag state

Response:

```
CS: .lo -lp00C02 -qlall -qss1 -qtb -sa4 -sbepc -sd0123456789abba9876543210 -sl60
-so0020 -sts1 -apdddcccc
EP: 0123456789ABBA9876543210
LS: Lock Success
OK:
```

The command's response provides the command executed after the CS: header and then for each responding transponder provides the outcome of the lock request. The above response shows a single transponder responding with *Lock Success* - a successful operation.

Task:

Attempt to write to the User memory with an incorrect access password.
Beware: as the reader stores parameters, once the access password has been specified for a command it will continue to be used even if not explicitly entered on the command line. For this reason the access password is re-specified as 00000000 in the following command as if you are following through these examples the reader will currently have it set to ddddcccc which it will continue to use unless otherwise specified.

Command:

```
.wr -ap 00000000 -db usr -dl 04 -do 0005 -da 5555666677778888 -ql all -qs s1 -qt
b -sa 4 -sb epc -sd 0123456789abba9876543210 -sl 60 -so 0020 -st s1
```

Response:

```
CS: .wr -qlall -qss1 -qtb -sa4 -sbepc -sd0123456789abba9876543210 -sl60 -so0020
-sts1 -ap00000000 -dbusr -dl04 -do0005 -da5555666677778888
EP: 0123456789ABBA9876543210
EB: 004
WW: 0
OK:
```

The command's response provides the command executed after the CS: header and then for each responding transponder provides the outcome of the request. The above response shows a single transponder responding with error *EB: 004* (memory is locked or perma-locked) - an unsuccessful operation!

Task:

Attempt to write to the User memory with correct access password.

Command:

```
.wr -ap ddddcccc -db usr -dl 04 -do 0005 -da 5555666677778888 -ql all -qs s1 -qt
b -sa 4 -sb epc -sd 0123456789abba9876543210 -sl 60 -so 0020 -st s1
```

Response:

```
CS: .wr -qlall -qss1 -qtb -sa4 -sbepc -sd0123456789abba9876543210 -sl60 -so0020
-sts1 -apdddcccc -dbusr -dl04 -do0005 -da5555666677778888
EP: 0123456789ABBA9876543210
WW: 4
OK:
```

The command's response provides the command executed after the CS: header and then for each responding transponder provides the outcome of the request. The above response shows 4 words written to a single transponder - a successful operation.

The transponder now requires a password to modify the User memory bank but it is not yet secure because the password itself can still be easily read or rewritten. To prevent this we can use the lock command to protect the reserved memory bank. The lock parameters allow for independent control over the Access password and the Kill password. The following example will modify the transponder so that the Access password can only be read or modified using the access password.

Task:

Prevent the reading and writing of the Access password unless the current Access password is specified.

Command:

```
.lo -lp 30080 -ap ddddcccc -ql all -qs s1 -qt b -sa 4 -sb epc -sd
0123456789abba9876543210 -sl 60 -so 0020 -st s1
```

Response:

```
CS: .lo LCMD 000044 -lp30080 -qlall -qss1 -qtb -sa4 -sbepc
-sd0123456789abba9876543210 -sl60 -so0020 -sts1 -apddddcccc
EP: 0123456789ABBA9876543210
LS: Lock Success
OK:
```

The command's response provides the command executed after the CS: header and then for each responding transponder provides the outcome of the request. The above response shows that the lock succeeded - a successful operation.
The Access password area is now only readable or writable using the Access password. This can be verified using read and write commands as described in the previous 'READING' and 'WRITING' sections.

The Lock command can also be used to set the Permalock bits which make memory banks permanently unchangeable – often these bits, once set, cannot be reset so use with caution.

WRITING A TRANSPONDER EPC

When writing a transponder *EPC* it is not advised to select the transponder by *EPC* as the value used to identify the transponder is changing during the process. This can cause problems in the situation where the write only partially succeeds (the response to the command returns the number of words successfully written)

When updating the *EPC* and the transponder supports a serialized *TID* bank (the *TID* memory contains a unique 64 bit value) the *TID* can be used as the select data and mask.

NOTES

READING AND WRITING LARGE AMOUNTS OF DATA

The Technology Solutions reader hardware limits the write command to 32 words at a time and a read command to 253 words at a time. Depending on the regulatory region the reader configured to operate the maximum dwell time at a particular frequency may reduce these values. The reader uses a frequency hopping algorithm and one of the constraints of the algorithm is the maximum time the reader can use any particular frequency before a hop is required. For example the ETSI region requires a hop every two seconds so is unlikely to affect these values. The FCC region requires a hop every 400ms in this case 32 words is approaching the maximum number of words that can be written before a hop is required.

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 C, Loughborough Technology Centre, Epinal Way, Loughborough, Leicestershire, LE11 3GE. United Kingdom.
Telephone:	+44 (0)1509 238248
Fax:	+44 (0)1509 220020
Email:	enquiries@tsl.uk.com
Website:	www.tsl.uk.com

ISO 9001: 2008