

USING ULTRALIGHT C

VERSION 100212



TABLE OF CONTENTS

1	Overview.....	3
1.1	Required Reading	3
1.2	Document Scope	3
1.3	Memory Structure.....	3
1.4	Lock Bytes	3
1.5	Authentication	4
2	Using Write Lock Features for Ultralight C	5
2.1	Details	5
2.2	Write Lock Bytes using Write Tag Command	5
2.2.1	Lock Bytes at Address 0x02	5
2.2.2	Example 1 - Lock Block 0x05 and 0x0A From Writing	6
2.2.3	Example 2 - Locking Lock bits for Blocks 4 - 9.....	6
2.2.4	Lock Bytes at Address 0x28	7
2.2.5	Example 3 - Lock Block 0x24 to 0x27 and AUTH1 From Writing	7
2.3	Get Lock Status Command	8
2.3.1	Example - Get lock status of block 0x05.....	8
3	Using Authentication Features for Ultralight C	9
3.1	Details	9
3.2	Write Authentication Key with Write Tag Command	9
3.2.1	Example - Write Authentication Password 0x000102030405060708090A0B0C0D0E0F.....	10
3.3	Write Authentication Configuration with Write Tag Command	10
3.3.1	Example - Write AUTH0 to 0x10 to restrict tag access beyond address 0x10	11
3.4	Authenticate Tag Command	11
3.4.1	Example - Authenticate tag with key 0x000102030405060708090A0B0C0D0E0F.....	12
4	References	13
5	Revision History:	14

1 Overview

1.1 Required Reading

This document assumes you have read and are familiar with the [SkyeTek Protocol V3 Reference Guide](#) and the [SkyeTek Protocol V3 Basic Examples](#) application note.

1.2 Document Scope

This application note describes how to use special features of Ultralight C (ULC) tags with the M2 and M4 modules. ULC tags have 2 special features that can be accessed using SkyeTek Protocol v3 without requiring familiarity with detailed ULC tag operations.

1.3 Memory Structure

There are 192 total bytes of memory, of which 144 bytes are user memory. The memory organization of ULC tags is illustrated below:

Page address		Byte number			
Decimal	Hex	0	1	2	3
0	00h	serial number			
1	01h	serial number			
2	02h	serial number	internal	lock bytes	lock bytes
3	03h	OTP	OTP	OTP	OTP
4 to 39	04h to 27h	user memory	user memory	user memory	user memory
40	28h	lock bytes	lock bytes	-	-
41	29h	16-bit counter	16-bit counter	-	-
42	2Ah	authentication configuration			
43	2Bh	authentication configuration			
44 to 47	2Ch to 2Fh	authentication key			

Figure 1 ULC Memory Organization^[1]

1.4 Lock Bytes

The write locking feature is an extension of the write locking feature also available in the Ultralight tags. There is more memory in the ULC tag, so additional lock bytes were added.

The lock bytes are located at bytes 2/3 of block 0x02 and bytes 0/1 of block 0x28. These bytes control which blocks of memory are locked from being written. Bits in the lock bytes are one time settable and cannot be returned to zeros. This means that ONCE A BLOCK IS SET TO READ-ONLY IT CAN NEVER BE RETURNED TO A WRITABLE STATE. The use

of lock bytes is described in detail in next section of this application note. The functions described in detail are:

- Write Lock Bytes
- Get Lock Status

1.5 Authentication

The authentication feature is a new security capability added to the ULC tag and not supported by standard Ultralight tags. ULC authentication gives users the ability to secure sections of user memory from write or read/write with a 128-bit key. The ULC tag uses 3DES authentication to verify that both the reader and tag have the same key before granting access to secure sections. The authentication features described in detail are:

- Write Authentication Configuration
- Write Authentication Key
- Authenticate Tag

2 Using Write Lock Features for Ultralight C

2.1 Details

Locking blocks of memory from write access is achieved through normal write tag commands to specific blocks of memory. The ULC tag contains 2 lock bytes located at block address 0x02 and 0x28. The lock byte at 0x02 controls write access to memory blocks 0x03 - 0x0F. The lock byte at 0x28 controls write access to memory blocks 0x10 - 0x2F. Once a bit is set in any of the lock bytes it cannot be unset, so changes to these bytes are IRREVERSIBLE.

Full detailed information about these lock bytes can be found in the Ultralight C datasheet[1]. A short description of the lock bytes and how to use the SkyTek Protocol v3 to set them is described in the following sections.

2.2 Write Lock Bytes using Write Tag Command

2.2.1 Lock Bytes at Address 0x02

The layout of lock bytes at address 0x02 are shown below:

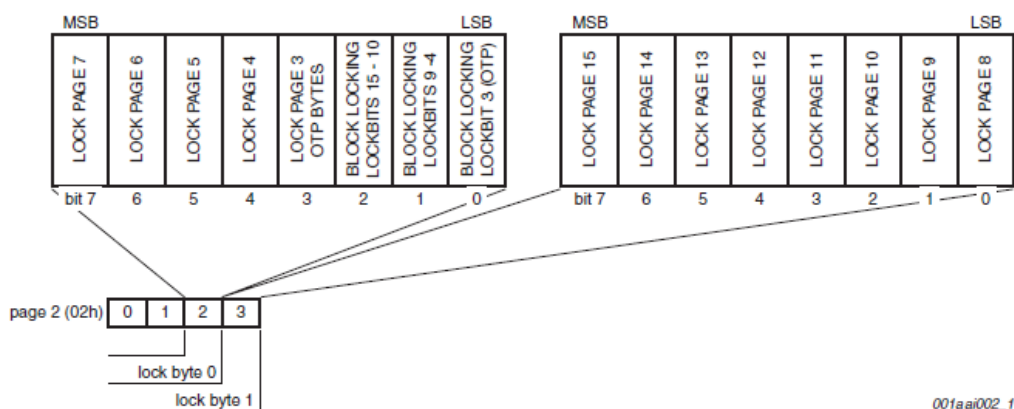


Figure 2 Lock bytes 0 and 1 ^[1]

Setting a 1 to lock bits labeled “Lock Page X” corresponds to locking the specified “Page” (equivalent to a block) for read-only access. Setting a 1 to the “Block Locking Lockbits X” makes the corresponding Lock Page bits uneditable. These bits essentially lock the read/write access for the selected pages.

2.2.2 Example 1 – Lock Block 0x05 and 0x0A From Writing

Request in ASCII:

	Flags	Cmd	Tag Type	TID Len	TID	Addr	# of Blocks	Data Length	Data	
<CR>	0840	0103	0211	07	<i>043A8589A72780</i>	0002	0001	0004	0000 2004	<CR>

- Sets bit 5 of lock byte 0 and bit 2 of lock byte 1 as illustrated in figure 2
- *Italicized* section is the Tag ID
- **Bold** section is the lock bytes

Response for success:

	Response Code	
<LF>	0103	<CR> <LF>

2.2.3 Example 2 – Locking Lock bits for Blocks 4 - 9

Request in ASCII:

	Flags	Cmd	Tag Type	TID Len	TID	Addr	# of Blocks	Data Length	Data	
<CR>	0840	0103	0211	07	<i>043A8589A72780</i>	0002	0001	0004	00000 200	<CR>

- Sets bit 2 of lock byte 0 as illustrated in figure 2
- *Italicized* section is the Tag ID
- **Bold** section is the lock bytes

Response for success:

	Response Code	
<LF>	0103	<CR> <LF>

2.2.4 Lock Bytes at Address 0x28

The layout of lock bytes at address 0x28 are shown below:

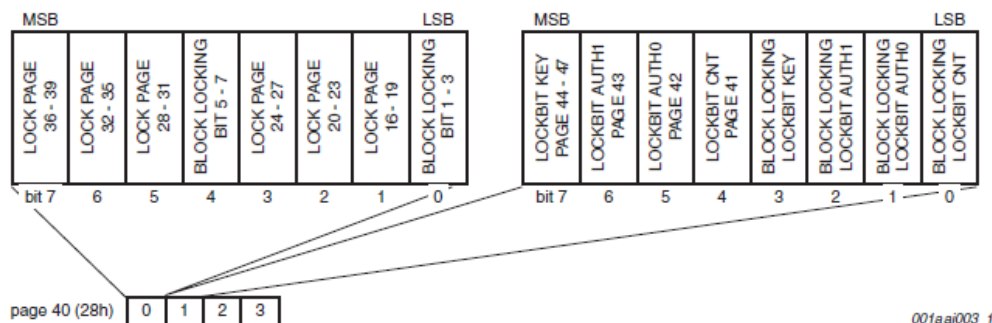


Figure 3 Lock bytes 2 and 3 ^[1]

Setting a 1 to lock bits labeled “Lock Page X-X” corresponds to locking the specified “Pages” (equivalent to blocks) for read-only access. Setting a 1 to the “Block Locking Lockbits X” makes the corresponding Lock Page bits uneditable. These bits essentially lock the read/write access for the selected pages. Note that figure 3 above also references the Key, Auth0 and Auth1. These correspond to sections of memory 0x2A-0x2F, which are used for the authentication feature described in detail in later sections of this application note.

2.2.5 Example 3 – Lock Block 0x24 to 0x27 and AUTH1 From Writing

Request in ASCII:

	Flags	Cmd	Tag Type	TID Len	TID	Addr	# of Blocks	Data Length	Data	
<CR>	0840	0103	0211	07	043A8589A72780	0028	0001	0004	80400000	<CR>

- Sets bit 7 of lock byte 2 and bit 6 of lock byte 3 as illustrated in figure 3
- *Italicized* section is the Tag ID
- **Bold** section is the lock bytes

Response for success:

	Response Code	
<LF>	0103	<CR> <LF>

2.3 Get Lock Status Command

The Get Lock Status command is a simple SkyTek Protocol v3 command that checks whether a specified block is locked and returns the status.

2.3.1 Example – Get lock status of block 0x05

Request in ASCII:

	Flags	Cmd	Tag Type	TID Len	TID	Data	
<CR>	0040	0108	0211	07	<i>043A8589A72780</i>	0005	<CR>

- Requests lock status of block 0x05
- *Italicized* section is the Tag ID
- **Bold** section is the block address in question

Response:

	Response Code	Data Length	Data	
<LF>	0108	0001	01	<CR> <LF>

- **Bold** section is the lock status
 - 0x00 for not locked
 - 0x01 for locked

3 Using Authentication Features for Ultralight C

3.1 Details

The security feature of the ULC tag allows the user to lock some or all of the tag memory from reading or writing. The amount of memory locked and whether the memory is locked only from writing or from reading and writing is set by writing specific values to the AUTH0 and AUTH1 bytes of tag memory. This is detailed in the section below titled “Write Authentication Configuration with Write Tag Command.”

Typical ULC tags come from the manufacturer with no memory restricted, allowing the user access to write their own authentication key and change the authentication configuration freely. The 128-bit 3DES authentication key can be set on a tag as described in the following section. Once access to secure memory is restricted, it can only be accessed by sending the authentication key as described in the “Authenticate Tag Command” section below.

3.2 Write Authentication Key with Write Tag Command

The authentication key is stored in address 0x2C-0x2F. Note that this area of tag memory is not readable. It can only be written to if the value of AUTH0 is 0x30, which is the typical default value from the tag manufacturer. The authentication key is stored in memory as shown below:

Byte address		0h	1h	2h	3h
Page address		Byte 0	Byte 1	Byte 2	Byte 3
2Ch	Page 44	Key1 / K0	Key1 / K1	Key1 / K2	Key1 / K3
2Dh	Page 45	Key1 / K4	Key1 / K5	Key1 / K6	Key1 / K7
2Eh	Page 46	Key2 / K0	Key2 / K1	Key2 / K2	Key2 / K3
2Fh	Page 47	Key2 / K4	Key2 / K5	Key2 / K6	Key2 / K7

Figure 4 Authentication Key Memory Configuration^[1]

The authentication key is stored in tag memory in little-endian format while it is sent in big-endian format for authentication. This can cause some confusion. Basically when you write the key to memory, it is written “backwards”. The data is sent to the reader in the order shown below:

Key1 [K0][K1][K2][K3][K4][K5][K6][K7], Key2 [K0][K1][K2][K3][K4][K5][K6][K7]

3.2.1 Example – Write Authentication Password 0x000102030405060708090A0B0C0D0E0F

Request in ASCII:

	Flags	Cmd	Tag Type	TID Len	TID	Addr	# of Blocks
<CR>	0840	0103	0211	07	043A8589A72780	002C	0004

Data Length	Data	
0010	07060504030201000F0E0D0C0B0A0908	<CR>

- Sets authentication key (Starting address 0x2C)
- *Italicized* section is the Tag ID
- **Bold** section is the key value sent in the order described in the previous section

Response for success:

	Response Code	
<LF>	0103	<CR> <LF>

3.3 Write Authentication Configuration with Write Tag Command

The configuration of the authentication feature on ULC tags is stored in the last blocks of the tag, 0x2A - 0x2F. The configuration is stored as follows:

- Address 0x2A
 - Byte 0 is AUTH0
 - AUTH0 sets the block address from which authentication is required
 - EX: If AUTH0 is 0x10, Authentication is required for access to all data from address 0x10 to the end of the tag memory 0x2F
 - Typical default value is 0x30 (no authentication required)
- Address 0x2B
 - Byte 0 is AUTH1
 - AUTH1 sets the write access for all memory defined in AUTH0
 - Value of 0x80 means only write access is restricted
 - Value of 0x00 means read and write access is restricted
 - Typical default value is 0x80 (write restrictions)
- Address 0x2C - 0x2F - 16 bytes makes up 128 bit Authentication Key

Writing authentication configuration is done with a write tag command at the correct address. This is illustrated in the following example.

3.3.1 Example – Write AUTH0 to 0x10 to restrict tag access beyond address 0x10

Request in ASCII:

	Flags	Cmd	Tag Type	TID Len	TID	Addr	# of Blocks	Data Length	Data	
<CR>	0840	0103	0211	07	<i>043A8589A72780</i>	002A	0001	0004	10000000	<CR>

- Sets AUTH0 to 0x10
- *Italicized* section is the Tag ID
- First **Bold** section is the address (0x2A for AUTH0)
- Second **Bold** section in the AUTH0 value

Response for success:

	Response Code	
<LF>	0103	<CR> <LF>

3.4 Authenticate Tag Command

The authenticate tag command is used to send the authentication key to a tag to gain access to restricted sections of tag memory. A series of commands is needed to select the tag, create a session with the tag, and then authenticate the tag. All commands during authentication and once authenticated must be sent with the RF flag on, because as soon as the tag loses power it will lose its authentication state. The RF flag ensures that the tag maintains power as long as it is within the field of the reader antenna. The example below will show how to get a ULC tag into the authenticated state with the authentication key written in the previous section. Note that the authentication key is now sent big endian as illustrated below:

Key1 [K7][K6][K5][K4][K3][K2][K1][K0], Key2 [K7][K6][K5][K4][K3][K2][K1][K0]

3.4.1 Example – Authenticate tag with key 0x000102030405060708090A0B0C0D0E0F

First Request in ASCII:

	Flags	Cmd	Tag Type	TID Len	TID	
<CR>	0048	0101	0211	07	<i>043A8589A72780</i>	<CR>

- Selects the tag with the specified tag in the field and creates a session (RF and TID flags set)
- *Italicized* section is the Tag ID

Response for success:

	Response Code	Data Length	Data	
<LF>	0101	0001	XX	<CR> <LF>

- **Bold** section is the session number used in the next command

Second Request in ASCII:

	Flags	Cmd	Tag Type	Session	Addr	Data Length
<CR>	0C08	0201	0211	XX	0000	0010

Data	
000102030405060708090A0B0C0D0E0F	<CR>

- Authenticate tag command (With RF, Data, and Session flags)
- First **bold** section is the session number noted in the previous step
- Second **bold** section is the authentication key sent big-endian

Response for success:

	Response Code	
<LF>	0201	<CR> <LF>

4 References

[1] MF0ICU2 - MIFARE Ultralight C (v.3.2) - Product short data sheet - 171432 - NXP

5 Revision History:

Revision	Author	Change
013112	Brad Alcorn	Initial Release
021712	Brad Alcorn	Fixed errors in Lock bits examples.
100212	Ryan Smith	Updated Overview Section