REYAX

TECHNOLOGY CORPORATION, LTD

06-AUG-2020 56312E30

RYRR20I

ISO15693

Command Manual

SOFTWARE PROTOCOLS

1. **The RYRR20I Protocol**

The RYRR20I protocol has a general form as shown below:

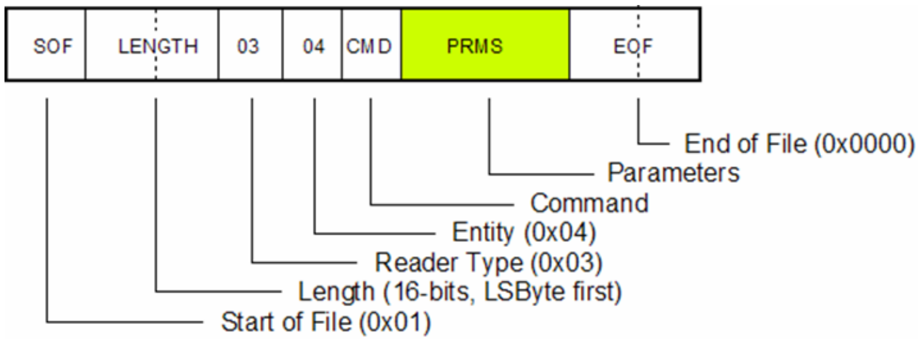


Figure 2. RYRR20I Host Protocol Format

The protocol begins with a 0x01 (Start Of File) byte, then 2-bytes for the length (Least Significant Byte first). Bytes 5 and 6 are always 0x03 and 0x04 respectively and they are followed by a command byte. Table 1 shows the commands that are described in this document.

Command	Parameters	Example
0x10 – Write Register	Address, Data, Address, data,	010C00030410002101000000
0x14 – Inventory Request	Flags, command code, data	010B000304140401000000
0x18 - ISO Request	Flags, command code, data	010B000304180020000000

Table 1. RYRR20I Command Codes

The 'PRMS' byte contains the parameters associated with the command and finally, the 2-byte EOF which is always 0x0000 completes the format.

2. **The ISO15693 Protocol**

The structure of ISO15693 commands is detailed in the standard:
ISO/IEC 15693-3. Identification cards – Contact-less integrated circuit(s) cards – Vicinity cards – Part3: Anti-collision and transmission protocol.
The general form is:

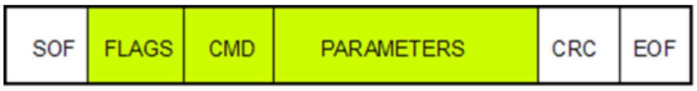


Figure 3. General Format of ISO15693 Request

When the RYRR20I is configured for ISO15693, the fields shown in green in Figure 3 replace the field marked "PRMS" in figure 2. The RYRR20I will then take that data, add the EOF and SOF and calculate the CCITT CRC-16 checksum before transmitting to a tag.

3. **Combined Protocol Example**

To see what happens, we can take a simple 'Read Single Block' command as an example:

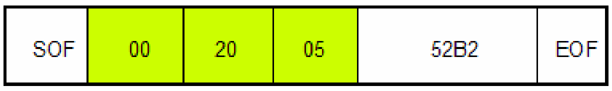


Figure 4. ISO15693 Read Block 5 Command

Figure 4 shows the complete ISO15693 Read Block 5 format.

The data in green is inserted into the RYRR20I Protocol and the total Host Command is as shown in Figure 5

01	09	00	03	04	18	00 20 05	00	00

Figure 5. RYRR20I Read Block 5 Command

CONFIGURING THE RYRR20I FOR ISO15693 OPERATION

The RYRR20I is configured for ISO15693 by using command 0x10 to write to the register. The examples below describe some of the various options and show the request/ responses from the RYRR20I:

```
>> 010C00030410002101000000
<< 010C00030410002101000000
 Register write request.
```

Information:

In this document ">>" indicates a request string and "<<" indicates a response string.

These are not part of the protocol and are added to make things clearer

The above request writes to the register to set up ISO15693 protocol with 1- out-of-4 modulation and full power

```
>> 010C00030410003101000000
<< 010C00030410003101000000
Register write request
```

The above request writes to the register to set up ISO15693 protocol with 1-out-of-4 modulation and half power

```
>> 010C00030410003101000000
<< 010C00030410003101000000
Register write request
```

The above request writes to the register to set up ISO15693 protocol with 1-out-of-256 modulation and full power

```
>> 010C00030410003101010000
<< 010C00030410003101010000
Register write request
```

The above request writes to the register to set up ISO15693 protocol and 1-out-of-256 modulation and half power
The RYRR20I is now configured to accept ISO15693 commands.

The ISO15693 Protocol

1. **ISO 15693 Flags**
The ISO15693 flags configure the tag IC. Such things as Date-rate, type of sub-carrier and option flags can all be set. The 8-bits are defined as follows:

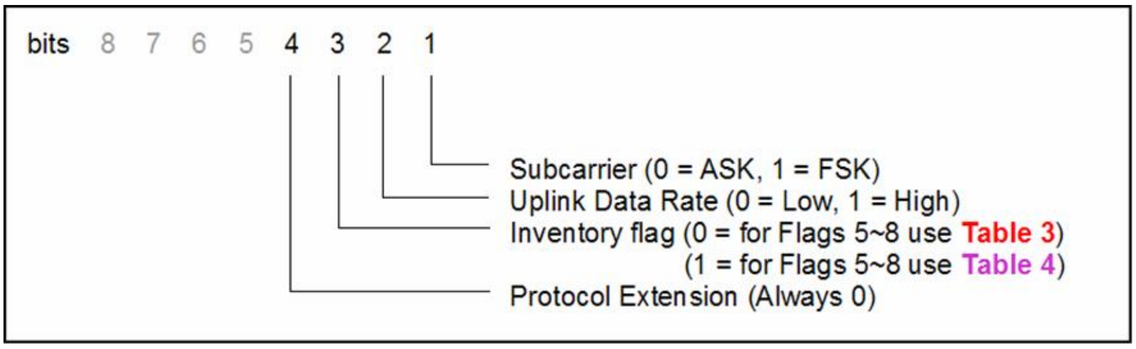

bits 8 7 6 5 4 3 2 1
Subcarrier (0 = ASK, 1 = FSK)
Uplink Data Rate (0 = Low, 1 = High)
Inventory flag (0 = for Flags 5~8 use Table 3)
(1 = for Flags 5~8 use Table 4)
Protocol Extension (Always 0)

Table 2. ISO15693 Protocol Flags

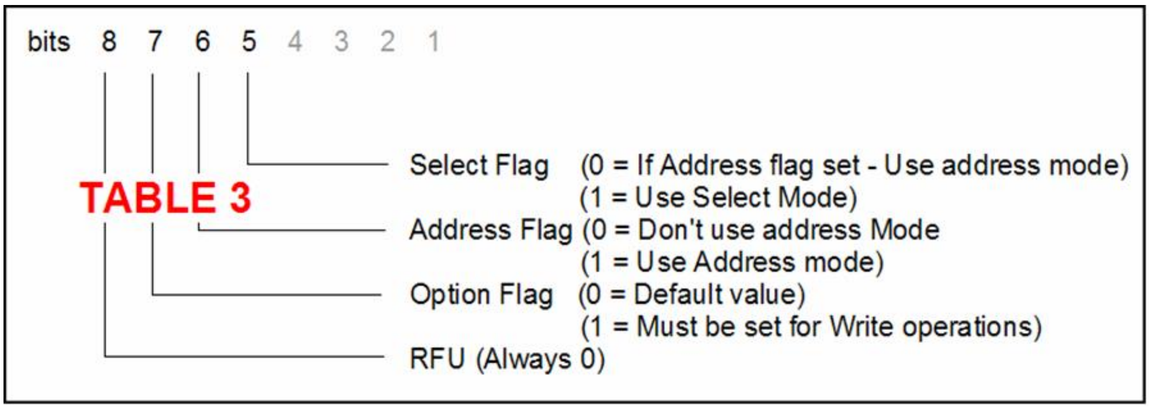

bits 8 7 6 5 4 3 2 1
TABLE 3
Select Flag (0 = If Address flag set - Use address mode)
(1 = Use Select Mode)
Address Flag (0 = Don't use address Mode
(1 = Use Address mode)
Option Flag (0 = Default value)
(1 = Must be set for Write operations)
RFU (Always 0)

Table 3. ISO15693 Inventory Flag not Set

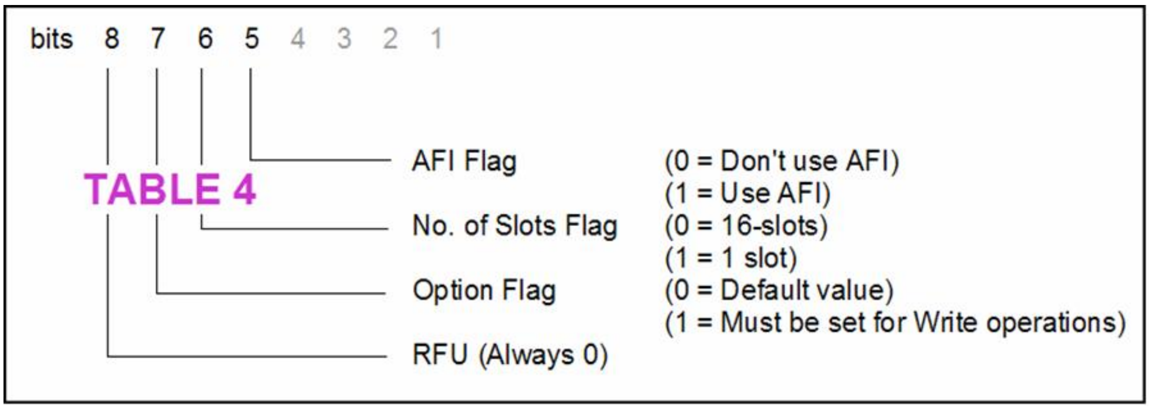

bits 8 7 6 5 4 3 2 1
TABLE 4
AFI Flag (0 = Don't use AFI)
(1 = Use AFI)
No. of Slots Flag (0 = 16-slots)
(1 = 1 slot)
Option Flag (0 = Default value)
(1 = Must be set for Write operations)
RFU (Always 0)

Table 4. ISO15693 Inventory Flag Set

As an example, for the Inventory Command to use 1-slot, the ISO15693 flags are:

```
>> 010B000304142401000000
              |  |  |
              |  |  └── 0x0100 (Parameters)
              |  └───── 0x24 (ASK, Low data rate, Inventory, No AFI, 1-slot)
              └──────── 0x14 (Inventory command)
```

And to use 16-slots:

```
>> 010B000304140401000000
              |  |  |
              |  |  └── 0x0100 (Parameters)
              |  └───── 0x04 (ASK, Low data rate, Inventory, No AFI, 16-slots)
              └──────── 0x14 (Inventory command)
```

2. ISO15693 Standard commands

Table 5 lists the standard ISO15693 commands.

Cmd	Command Type	Tag Type	Function	Option Flag
01	Mandatory	All	Inventory	No
02	Mandatory	All	Stay Quiet	No
20	Optional	Std, Pro, Plus	Read Single Block	No
21	Optional	Std, Pro, Plus	Write Single Block	Yes
22	Optional	Std, Pro, Plus	Lock Block	Yes
23	Optional	Plus only	Read Multiple Blocks	No
24	Optional	Plus only	Write Multiple Blocks	Yes
25	Optional	Plus only	Select	No
26	Optional	Plus only	Reset to Ready	No
27	Optional	Plus only	Write AFI	Yes
28	Optional	Plus only	Lock AFI	Yes
29	Optional	Plus only	Write DSFID	Yes
2A	Optional	Plus only	Lock DSFID	Yes
2B	Optional	Plus only	Get System Information	No
2C	Optional	Plus only	Get Multiple Block Security Status	No

Table 5. Standard ISO15693 Commands

In addition there are custom commands.

Cmd	Command Type	Tag Type	Function	Option Flag
A2	Custom	Plus only	Write 2 Blocks	Yes
A3	Custom	Plus only	Lock 2 Blocks	Yes
A4	Custom	Pro only	KILL	Yes
A5	Custom	Pro only	Write Single Block PWD	Yes

Table 6. Custom ISO15693 Commands

3. **Response Codes**
ISO15693 responses contain a status byte which the RYRR20I extracts and appends [inside square brackets] to the response. Table 7 shows some of the possible responses.

Response Code	Meaning
[]	No status Information
[00]	Command was successful
[0101]	Command not supported
[0102]	Command not recognized (e.g. Format error)
[0103]	Option not supported
[010F]	Unknown error
[0110]	Block not available (out of range)
[0111]	Block already locked (can' t be locked again)
[0112]	Block already locked – contents can't be changed
[0113]	Programming was unsuccessful
[0114]	Locking/Kill was unsuccessful
[01A1]	Start block must be even
[01A2]	One or both blocks already locked
[01B0]	Read Access denied

Table 7. Response codes

INDIVIDUAL ISO15693 COMMANDS

The following sections describe the requests and responses for each command.

1. **Inventory**
The request/response for an Inventory command with 1-slot is:

```
>> 010B000304142401000000
<< 010B000304142401000000

ISO 15693 Inventory request.

[9080C2E5D2C407E0,7F]
```

This shows that tag "E007C4D2E5C28090" responded.

The request/response for an Inventory command with 16-slots is:

```
>> 010B000304140401000000
<< 010B000304140401000000
ISO 15693 Inventory request.
[9080C2E5D2C407E0,7F]
[,40]
[,40]
[,40]
[,40]
[,40]
[,40] [379EC2E5D2C407E0,7F]
[,40]
[,40]
[,40]
[,40]
[,40]
[,40]
[,40]
[,40]
```

This shows that tag "E007C4D2E5C28090" responded in slot 0 and "E007C4D2E5C29E37" in slot 7

2. **Stay Quiet (0x02)**
   The Stay Quiet request/response sequences will be similar to:

```
>> 0112000304182002105 2FE01000007E00000
```

```
>> 0112000304182002 1052FE01000007E00000
<< 0112000304182002 1052FE01000007E00000
Request mode.
[]>
```

This shows a request to tag ID E007000001FE5210 to Stay Quiet. No tag responds to a Stay Quiet request, so even if the command fails the reader will still return the same response.

3. **Read Single Block (0x20)**
   The request/responses to perform an unaddressed Read of block 0x05 will be similar to:

```
>> 010B000304180020050000
<< 010B000304180020050000
Request mode.
[0078563412]
```

"[78563412]" is the block data (12345678) that was read.

Read Single Block (Addressed)

The request/responses from an addressed read (Address is "E007804651E49C57") of block 0x05 will be similar to:

```
>> 11300030418202057 9CE451468007E0050000
<< 011300030418202057 9CE451468007E0050000

Request mode.
[0078563412]
```

4. Write Single Block (0x21)

The Write Single Block command must have the Option flag set (0x**40**) and the data must be LSByte first. The ISO15693 format is shown in the figure below:

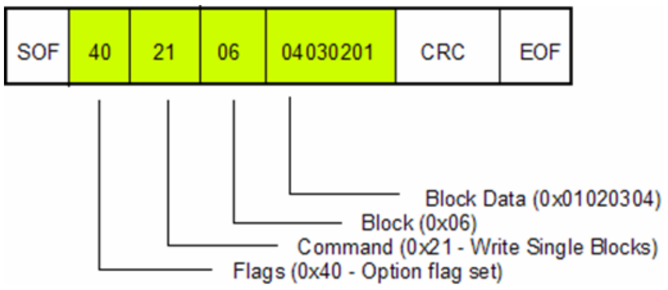

SOF
40
21
06
04030201
CRC
EOF
Block Data (0x01020304)
Block (0x06)
Command (0x21 - Write Single Blocks)
Flags (0x40 - Option flag set)

Figure 8. Write Single Block Format

This translates into the following RYRR20I request/responses

```
>> 010F00030418402106040302010000
<< 010F00030418402106040302010000

Request mode.
[00]
```

The [00] response indicates that the command succeeded. For other possible responses see Table 7.

Write Single Block (Addressed)

For an addressed Write Single Block, both Address and Option flags (0x**90**) must be set. In this example the address is E007C4D2E5C28090, the block is 0x06 and the data is 0x22334455. The RYRR20I request/responses will be similar to:

```
>> 0117000304186021909080C2E5D2C407E00655443322 0000
<< 01170003041860219080C2E5D2C407E006554433220000

Request mode.
[00]
```

5. Lock Block (0x22)

For the Lock Block command the Option flag (0x40) must be set. In this example the block to be locked is 0x06. The RYRR20I request/responses will be similar to:

```
>> 010B000304184022060000
<< 010B000304184022060000

Request mode.
[]
```

Lock Block (Addressed)

For an addressed Lock Block command, the Option and Address flags (0x90) must be set. In this example the address is E007C4D2E5C28090 and the block is 0x06. The RYRR20I request/responses will be similar to the following:

```
>> 0113000304186022909080C2E5D2C407E0060000
<< 011300030418602290 80C2E5D2C407E0060000

Request mode.
[]
```

6. **Read Multiple Blocks (0x23)**

In this example the start block is 0x01 and the number of blocks is 02 The RYRR20I request/responses will be similar to the following:

```
>> 010C00030418002301020000
<< 010C00030418002301020000

Request mode.
[00111111122222222]
```

Read Multiple Blocks (Addressed)

In this example the address is E007804651E49C57, the start block is 01 and number of blocks is 02)

```
>> 0114000304182023579CE451468007E001020000
<< 0114000304182023579CE451468007E001020000

Request mode.
[00111111122222222]
```

7. **Write Multiple Blocks (0x24)**

In this example the request is to program pages 0x0A, 0x0B and 0x0C with 0xAAAAAAAA, 0xBBBBBBBB and 0xCCCCCCCC. The GUI sends iterative Write Single Block commands

```
>> 010F0003041840210AAAAAAAAA0000
<< 010F0003041840210AAAAAAAAA0000
Request mode.
[00]
>> 010F0003041840210BBBBBBBBB0000
<< 010F0003041840210BBBBBBBBB0000
Request mode.
[00]
>> 010F0003041840210CCCCCCCCC0000
<< 010F0003041840210CCCCCCCCC0000
Request mode.
[00]
```

Write Multiple Blocks (Addressed)

This example to tag E007804651E49C57 is to write to blocks 0x07, 0x08 and 0x09 with 0xEEEEEEEE, 0xFFFFFFFF and 0x10101010 respectively. The GUI sends multiple Write Single Block commands:

```
>> 0117000304186021579CE451468007E007EEEEEEEE0000
>> 0117000304186021579CE451468007E007EEEEEEEE0000
Request
mode.
[00]
>> 0117000304186021579CE451468007E008FFFFFFFF0000
<< 0117000304186021579CE451468007E008FFFFFFFF0000
Request
mode.
[00]

>> 0117000304186021579CE451468007E009101010100000
<< 0117000304186021579CE451468007E009101010100000
Request mode.
[00]
```

8. **Select (0x25)**
   This command is only allowed as an addressed command. In this example tag E007804651E49C57 is instructed to enter Select state.

   Note that the select flag is set.

```
>> 0112000304182025 07B85812000007E00000
<< 0112000304182025 07B85812000007E00000
Request mode.
[00]
```

9. **Reset to Ready (0x26)**

   This command returns all selected tags to the normal state. The request/responses will be similar to:

```
>> 010A0003041810260000
<< 010A0003041810260000
Request mode.
[00]
```

10. **Write AFI (0x27)**
    This command writes a code to a tag that tells it which family of applications that tag belongs too. In this example the family code is 0x**30** - access control.

    The request/responses of the Write AFI command will be similar to:

```
>> 010B000304184027300000
<< 010B000304184027300000
Request mode.
[00]
```

Write AFI (Addressed)
In this example the Address is E00700001258B807 and the AFI is 0x**30.** The request/responses will be similar to:

```
>> 011300030418602707B85812000007E0300000
<< 011300030418602707B85812000007E0300000
Request mode.
[00]
```

11. **Lock AFI (0x28)**

    The AFI can be locked to prevent it being changed. The request/responses will be similar to:

```
>> 010A0003041840280000
<< 010A0003041840280000
Request mode.
[]
```

Lock AFI (Addressed)
This example locks the AFI of tag E00700001258B87B. The request/responses will be similar to:

```
>> 011200030418602807B85812000007E00000
<< 011200030418602807B85812000007E00000
Request mode.
[]
```

12. **Write DSFID (0x29)**

The DSFID is used to provide information about how the memory of a tag is configured. In this example 0x**AA** is written into the DSFID field. The request/responses to this command will be similar to:

```
>> 010B000304184029AA0000
<< 010B000304184029AA0000
Request mode.
[00]
```

Write DSFID (Addressed)

In this example the tag addressed is E00700001258B807 and the DSFID data is 0x**AA.** The request/responses will be similar to:

```
>> 0113000304186029 07B85812000007E0AA0000
<< 01130003041860290 7B85812000007E0AA0000
Request mode.
[00]
```

13. **Lock DSFID (0x2A)**

The DSFID field can also be locked to prevent the data being changed. This example shows the expected request/responses:

```
>> 010A00030418402A0000
<< 010A00030418402A0000
Request mode.
[]
```

Lock DSFID (Addressed)

This example show the locking ofthe DSFID fieldfor tag E00700001258B807. The request/responses will be similar to:

```
>> 011200030418602A07B85812000007E00000
<< 011200030418602A07B85812000007E00000
Request mode.
[]
```

14. **Get System Information (0x2B)**

This command returns tag information. The request/responses will be similar to:

```
>> 010A00030418002B0000
<< 010A00030418002B0000
Request mode.
[000F07B85812000007E0AA303F0388]
```

The System information is decoded as follows:

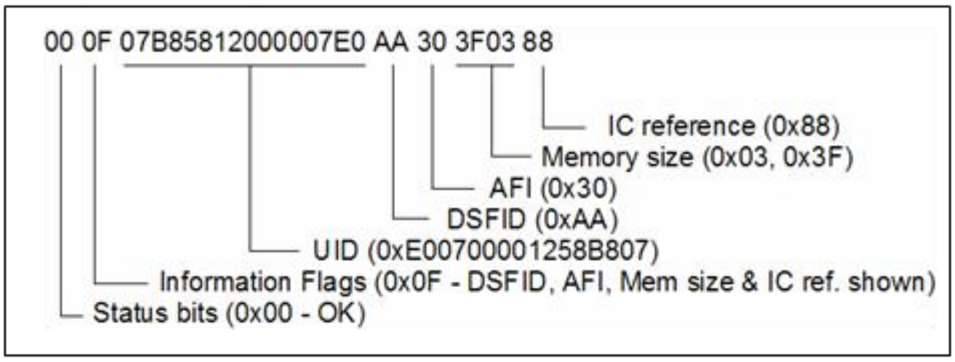


Figure 9. System Information

Get System Information (Addressed)

This example requests the system information from tag E00700001258B807. The request/responses will be similar to:

```
>> 0112000304182020B07B85812000007E00000
<< 0112000304182020B07B85812000007E00000
Request mode.
[000F07B85812000007E0AA303F0388]
```

15. **Get Multiple Block Security Status (0x2C)**

This command returns the status of multiple blocks, indicating if they are locked or open. In this example, the security status of blocks 01 to 04 is requested:

```
>> 010C00030418002C01030000
<< 010C00030418002C01030000
Request mode.
[0001010100]
```

From the response it can be seen that the status (0x00) is OK and blocks 01, 02 and 03 are locked (0x010101) and block 04 is unlocked (0x00)

Get Multiple Block Security Status (Addressed)

This example requests the security status from tag E00700001258B807 for blocks 01 to 04. The request/responses will be similar to:

```
>> 011400030418202C07B85812000007E001030000
<< 011400030418202C07B85812000007E001030000
Request mode.
[0001010100]
```

16. **Write 2 Blocks (0xA2)**

This custom command instructs the tag (not the reader) to program 2 consecutive blocks. All custom commands require the manufacturer code (TI's is 0x07) to follow the command. In this example the Start Block is 0x02 and Data is 0xAAAAAAAA and 0xBBBBBBBB respectively.

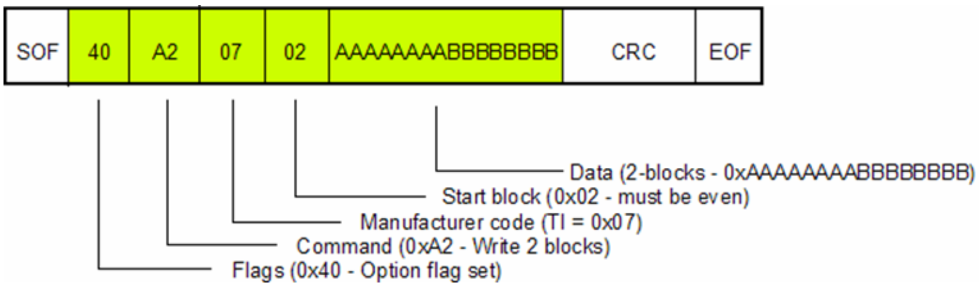

SOF
40
A2
07
02
AAAAAAAABBBBBBBB
CRC
EOF
Data (2-blocks - 0xAAAAAAAABBBBBBBB)
Start block (0x02 - must be even)
Manufacturer code (TI = 0x07)
Command (0xA2 - Write 2 blocks)
Flags (0x40 - Option flag set)

Figure 10. Write 2 Blocks Format

This translates into the following RYRR20I request/responses:

```
>> 0114000304184 0A20702AAAAAAAABBBBBBBB0000
<< 011400030418 40A20702AAAAAAAABBBBBBBB0000
Request mode.
[]
```

Write 2 Blocks (Addressed)

This example request tag E00700001258B81E to write 2 blocks starting with block 0x00. the data to be written is 0xEEEEEEEE and 0xFFFFFFFF. The request/responses will be similar to:

```
>> 011C0003041860A2071EB85812000007E000EEEEEEEEFFFFFFFF0000
<< 011C0003041860A2071EB85812000007E000EEEEEEEEFFFFFFFF0000
Request mode.
[]
```

17. **Lock 2 Blocks (0xA3)**

This is a custom command to request the tag (not the reader) to lock 2 consecutive blocks. In this example the start block is 0x02. The ISO15693 format is shown in the figure below:

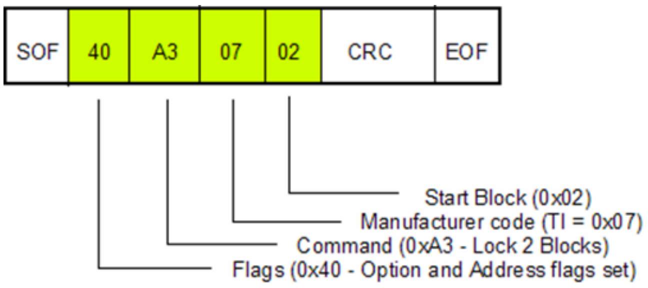

SOF
40
A3
07
02
CRC
EOF
Start Block (0x02)
Manufacturer code (TI = 0x07)
Command (0xA3 - Lock 2 Blocks)
Flags (0x40 - Option and Address flags set)

Figure 11. Lock 2 Blocks Format

This translates to the following RYRR20I command:

```
>> 010C00030418 40A307020000
<< 010C00030418 40A307020000
Request mode.
[]
```

Lock 2 Blocks (Addressed)

This example requests tag 0xE00700001258B81E to lock 2 blocks starting at block 0x00.

```
>> 0114000304186 0A3071EB85812000007E0000000
<< 0114000304186 0A3071EB85812000007E0000000
Request mode.
[]
```

18. **Kill (0xA4)**

This is a HF-I Pro custom command to permanently disable a tag and requires a password to be programmed into page 11 (0x0B) and locked. In this example the password is 0x12345678 and the UID is E007C4A509C247A8. The ISO15693 format is shown in Figure 11

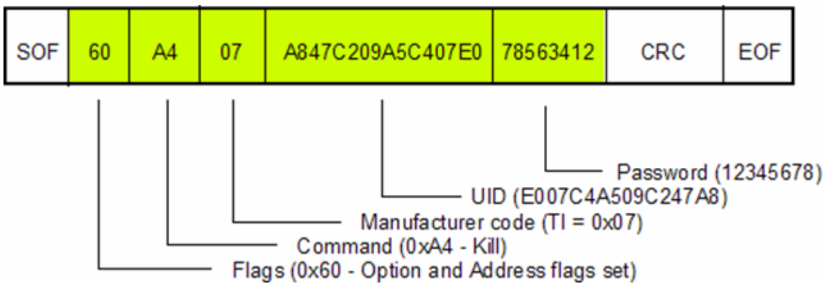


Figure 12. Kill Command Format

This translates into the following RYRR20I request/responses:

```
>> 01170003041863A407A847C209A5C407E0785634120000
<< 01170003041863A407A847C209A5C407E0785634120000
Request mode.
[]
```

19. **Write Single Block Password (0xA5)**

This is a HF-I Pro custom command to write data to an already locked location. To do this the 32-bit password locked in block 11 (0x0B) must be known, as well as the UID. All data is MSByte first. In the example the address is 0xE007C4A509C2477C, the password is 0x1234567 and the data is 0x77777777. The ISO format is shown in the figure below:

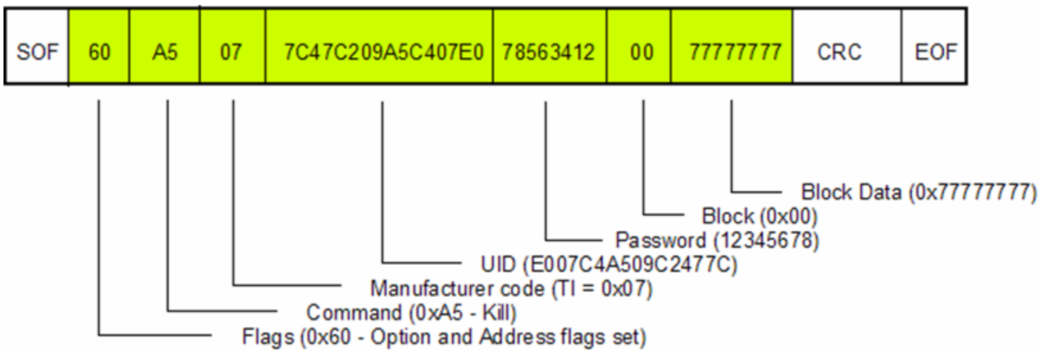


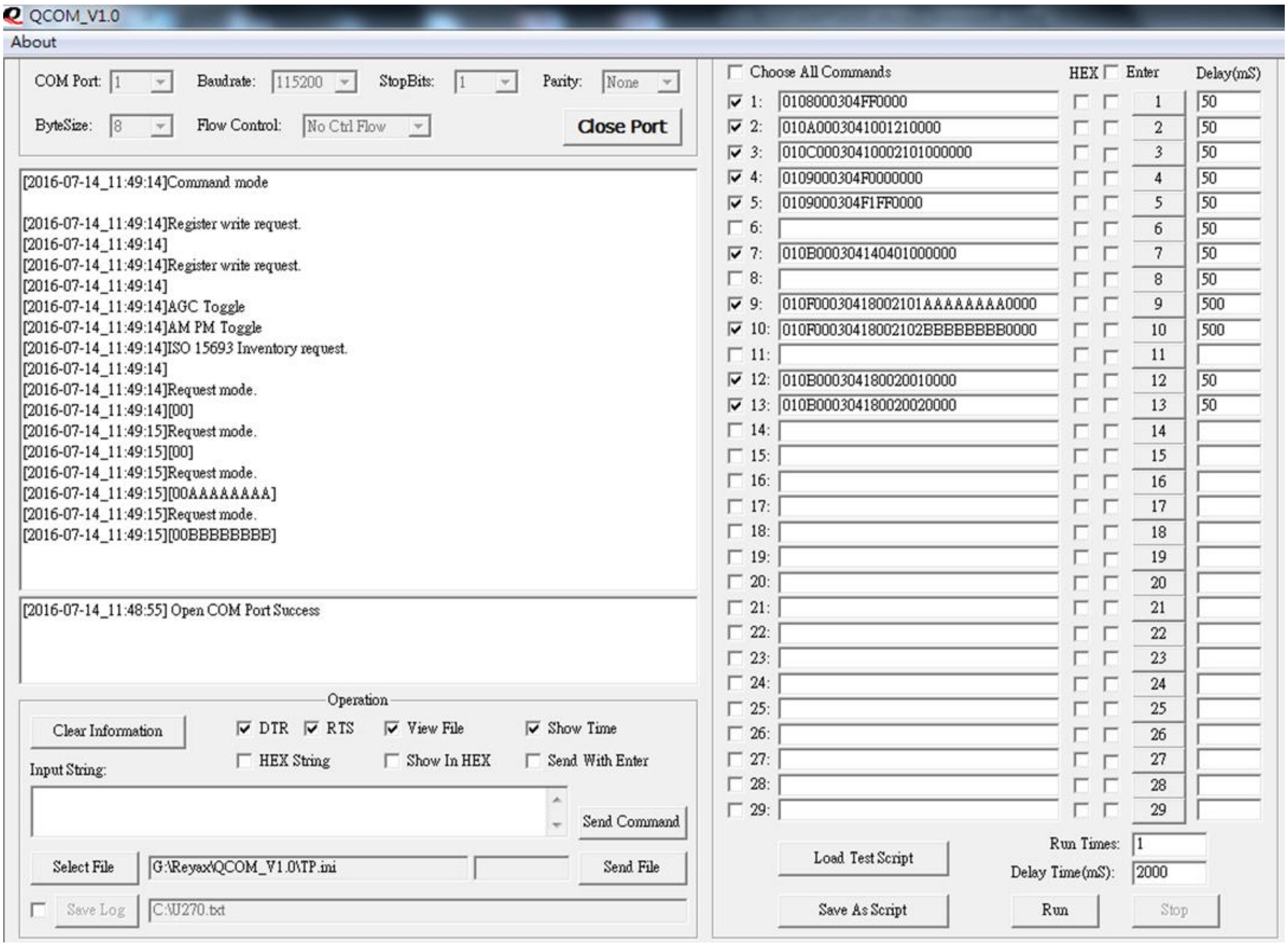
Figure 13. WriteSingleBlockPWD format

This translates into the following RYRR20I request/responses:

```
>> 011C0003041863A5077C47C209A5C407E07856341200777777770000
<< 011C0003041863A5077C47C209A5C407E07856341200777777770000
Request mode.
[]
```

20. Examples

RYRR20I Read / Write TAG memory



COM Port: 1 Baudrate: 115200 StopBits: 1 Parity: None
ByteSize: 8 Flow Control: No Ctrl Flow

```
[2016-07-14_11:49:14]Command mode

[2016-07-14_11:49:14]Register write request.
[2016-07-14_11:49:14]
[2016-07-14_11:49:14]Register write request.
[2016-07-14_11:49:14]
[2016-07-14_11:49:14]AGC Toggle
[2016-07-14_11:49:14]AM PM Toggle
[2016-07-14_11:49:14]ISO 15693 Inventory request.
[2016-07-14_11:49:14]
[2016-07-14_11:49:14]Request mode.
[2016-07-14_11:49:14][00]
[2016-07-14_11:49:15]Request mode.
[2016-07-14_11:49:15][00]
[2016-07-14_11:49:15]Request mode.
[2016-07-14_11:49:15][00AAAAAAAA]
[2016-07-14_11:49:15]Request mode.
[2016-07-14_11:49:15][00BBBBBBBB]
```

```
[2016-07-14_11:48:55] Open COM Port Success
```

Operation
[✓] DTR [✓] RTS [✓] View File [✓] Show Time
[ ] HEX String [ ] Show In HEX [ ] Send With Enter

Input String:

G:\Reyax\QCOM_V1.0\TP.ini

[ ] C:\U270.txt

[ ] Choose All Commands

	Command	HEX	Enter		Delay(mS)
[✓] 1:	0108000304FF0000	[ ]	[ ]	1	50
[✓] 2:	010A0003041001210000	[ ]	[ ]	2	50
[✓] 3:	010C00030410002101000000	[ ]	[ ]	3	50
[✓] 4:	0109000304F0000000	[ ]	[ ]	4	50
[✓] 5:	0109000304F1FF0000	[ ]	[ ]	5	50
[ ] 6:		[ ]	[ ]	6	50
[✓] 7:	010B000304140401000000	[ ]	[ ]	7	50
[ ] 8:		[ ]	[ ]	8	50
[✓] 9:	010F00030418002101AAAAAAAA0000	[ ]	[ ]	9	500
[✓] 10:	010F00030418002102BBBBBBBB0000	[ ]	[ ]	10	500
[ ] 11:		[ ]	[ ]	11	
[✓] 12:	010B000304180020010000	[ ]	[ ]	12	50
[✓] 13:	010B000304180020020000	[ ]	[ ]	13	50
[ ] 14:		[ ]	[ ]	14	
[ ] 15:		[ ]	[ ]	15	
[ ] 16:		[ ]	[ ]	16	
[ ] 17:		[ ]	[ ]	17	
[ ] 18:		[ ]	[ ]	18	
[ ] 19:		[ ]	[ ]	19	
[ ] 20:		[ ]	[ ]	20	
[ ] 21:		[ ]	[ ]	21	
[ ] 22:		[ ]	[ ]	22	
[ ] 23:		[ ]	[ ]	23	
[ ] 24:		[ ]	[ ]	24	
[ ] 25:		[ ]	[ ]	25	
[ ] 26:		[ ]	[ ]	26	
[ ] 27:		[ ]	[ ]	27	
[ ] 28:		[ ]	[ ]	28	
[ ] 29:		[ ]	[ ]	29	

Run Times: 1
Delay Time(mS): 2000

Copyright © 2020, REYAX TECHNOLOGY CO., LTD.

Show in ASCII string

```
0108000304FF0000                        'Entry command mode
Command mode
010A0003041001210000                    'Parameter setup
Register write request.
010C00030410002101000000                'Parameter setup
Register write request.
0109000304F0000000                      'Parameter setup
AGC Toggle
0109000304F1FF0000                      'Parameter setup
AM PM Toggle
010B000304140401000000                  'ISO15693 protocol
ISO 15693 Inventory request
010F00030418002101AAAAAAAA0000          'write data AAAAAAAA to 01 memory block
Request mode.
[00]
010F00030418002102BBBBBBBB0000          'write data BBBBBBBB to 02 memory block
Request mode.
[00]
010B000304180020010000                  'Read 01 memory block
Request mode.
 [00AAAAAAAA]
010B000304180020020000                  'Read 02 memory block
Request mode.
[00BBBBBBBB]
```

REYAX
TECHNOLOGY CORPORATION, LTD

Taiwan: sales@reyax.com
China: sales@reyax.com.cn
http://reyax.com

Copyright © 2020, REYAX TECHNOLOGY CO., LTD.