



## SRIX4K

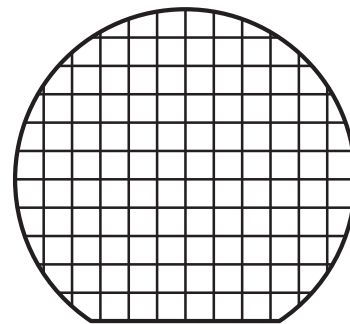
---

13.56 MHz short-range contactless memory chip  
with 4096-bit EEPROM, anticollision and anti-clone functions

---

### Features

- ISO 14443-2 Type B air interface compliant
- ISO 14443-3 Type B frame format compliant
- 13.56 MHz carrier frequency
- 847 kHz subcarrier frequency
- 106 Kbit/second data transfer
- France Telecom proprietary anti-clone function
- 8 bit Chip\_ID based anticollision system
- 2 count-down binary counters with automated antitearing protection
- 64-bit unique identifier
- 4096-bit EEPROM with write protect feature
- Read\_block and Write\_block (32 bits)
- Internal tuning capacitor
- 1million erase/write cycles
- 40-year data retention
- Self-timed programming cycle
- 5 ms typical programming time



- Unsawn wafer
- Bumped and sawn wafer

# Contents

|          |                                                                         |           |
|----------|-------------------------------------------------------------------------|-----------|
| <b>1</b> | <b>Description .....</b>                                                | <b>7</b>  |
| <b>2</b> | <b>Signal description .....</b>                                         | <b>8</b>  |
|          | 2.0.1 AC1, AC0 .....                                                    | 8         |
| <b>3</b> | <b>Data transfer .....</b>                                              | <b>9</b>  |
| 3.1      | Input data transfer from the reader to the SRIX4K (request frame) ..... | 9         |
| 3.1.1    | Character transmission format for request frame .....                   | 9         |
| 3.1.2    | Request start of frame .....                                            | 10        |
| 3.1.3    | Request end of frame .....                                              | 10        |
| 3.2      | Output data transfer from the SRIX4K to the reader (answer frame) ....  | 11        |
| 3.2.1    | Character transmission format for answer frame .....                    | 11        |
| 3.2.2    | Answer start of frame .....                                             | 11        |
| 3.2.3    | Answer end of frame .....                                               | 12        |
| 3.3      | Transmission frame .....                                                | 12        |
| 3.4      | CRC .....                                                               | 13        |
| <b>4</b> | <b>Memory mapping .....</b>                                             | <b>14</b> |
| 4.1      | Resettable OTP area .....                                               | 15        |
| 4.2      | 32-bit binary counters .....                                            | 16        |
| 4.3      | EEPROM area .....                                                       | 17        |
| 4.4      | System area .....                                                       | 18        |
| 4.4.1    | OTP_Lock_Reg .....                                                      | 19        |
| 4.4.2    | Fixed Chip_ID (Option) .....                                            | 19        |
| <b>5</b> | <b>SRIX4K operation .....</b>                                           | <b>20</b> |
| <b>6</b> | <b>SRIX4K states .....</b>                                              | <b>21</b> |
| 6.1      | Power-off state .....                                                   | 21        |
| 6.2      | Ready state .....                                                       | 21        |
| 6.3      | Inventory state .....                                                   | 21        |
| 6.4      | Selected state .....                                                    | 21        |
| 6.5      | Deselected state .....                                                  | 21        |

|                         |                                                |           |
|-------------------------|------------------------------------------------|-----------|
| 6.6                     | Deactivated state .....                        | 21        |
| <b>7</b>                | <b>Anticollision .....</b>                     | <b>23</b> |
| 7.1                     | Description of an anticollision sequence ..... | 25        |
| <b>8</b>                | <b>Anti-clone function .....</b>               | <b>27</b> |
| <b>9</b>                | <b>SRIX4K commands .....</b>                   | <b>28</b> |
| 9.1                     | Initiate() command .....                       | 29        |
| 9.2                     | Pcall16() command .....                        | 30        |
| 9.3                     | Slot_marker(SN) command .....                  | 31        |
| 9.4                     | Select(Chip_ID) command .....                  | 32        |
| 9.5                     | Completion() command .....                     | 33        |
| 9.6                     | Reset_to_inventory() command .....             | 34        |
| 9.7                     | Read_block(Addr) command .....                 | 35        |
| 9.8                     | Write_block (Addr, Data) command .....         | 36        |
| 9.9                     | Get_UID() command .....                        | 37        |
| 9.10                    | Power-on state .....                           | 38        |
| <b>10</b>               | <b>Maximum rating .....</b>                    | <b>39</b> |
| <b>11</b>               | <b>DC and ac parameters .....</b>              | <b>40</b> |
| <b>12</b>               | <b>Part numbering .....</b>                    | <b>42</b> |
| <b>Appendix A</b>       | <b>ISO 14443 Type B CRC calculation .....</b>  | <b>43</b> |
| <b>Appendix B</b>       | <b>SRIX4K command summary.....</b>             | <b>44</b> |
| <b>Revision history</b> | <b>.....</b>                                   | <b>46</b> |

# List of tables

Table 1. Signal names . . . . . 7

Table 2. Bit description . . . . . 10

Table 3. Standard anticollision sequence . . . . . 25

Table 4. Command code . . . . . 28

Table 5. Absolute maximum ratings . . . . . 39

Table 6. Operating conditions . . . . . 40

Table 7. DC characteristics . . . . . 40

Table 8. AC characteristics . . . . . 40

Table 9. Ordering information scheme . . . . . 42

Table 10. Document revision history . . . . . 46

## List of figures

|            |                                                                      |    |
|------------|----------------------------------------------------------------------|----|
| Figure 1.  | Logic diagram . . . . .                                              | 7  |
| Figure 2.  | Die floor plan. . . . .                                              | 8  |
| Figure 3.  | 10% ASK modulation of the received wave . . . . .                    | 9  |
| Figure 4.  | SRIX4K request frame character format. . . . .                       | 9  |
| Figure 5.  | Request start of frame . . . . .                                     | 10 |
| Figure 6.  | Request end of frame . . . . .                                       | 10 |
| Figure 7.  | Wave transmitted using BPSK subcarrier modulation . . . . .          | 11 |
| Figure 8.  | Answer start of frame . . . . .                                      | 11 |
| Figure 9.  | Answer end of frame. . . . .                                         | 12 |
| Figure 10. | Example of a complete transmission frame . . . . .                   | 12 |
| Figure 11. | CRC transmission rules . . . . .                                     | 13 |
| Figure 12. | SRIX4K memory mapping . . . . .                                      | 14 |
| Figure 13. | Resettable OTP area (addresses 0 to 4) . . . . .                     | 15 |
| Figure 14. | Write_block update in Standard mode (binary format) . . . . .        | 15 |
| Figure 15. | Write_block update in Reload mode (binary format). . . . .           | 16 |
| Figure 16. | Binary counter (addresses 5 to 6). . . . .                           | 16 |
| Figure 17. | Count down example (binary format) . . . . .                         | 17 |
| Figure 18. | EEPROM (addresses 7 to 127) . . . . .                                | 18 |
| Figure 19. | System area . . . . .                                                | 18 |
| Figure 20. | State transition diagram . . . . .                                   | 22 |
| Figure 21. | SRIX4K Chip_ID description. . . . .                                  | 23 |
| Figure 22. | Description of a possible anticollision sequence . . . . .           | 24 |
| Figure 23. | Example of an anticollision sequence. . . . .                        | 26 |
| Figure 24. | Initiate request format . . . . .                                    | 29 |
| Figure 25. | Initiate response format . . . . .                                   | 29 |
| Figure 26. | Initiate frame exchange between reader and SRIX4K . . . . .          | 29 |
| Figure 27. | Pcall16 request format . . . . .                                     | 30 |
| Figure 28. | Pcall16 response format. . . . .                                     | 30 |
| Figure 29. | Pcall16 frame exchange between reader and SRIX4K . . . . .           | 30 |
| Figure 30. | Slot_marker request format . . . . .                                 | 31 |
| Figure 31. | Slot_marker response format . . . . .                                | 31 |
| Figure 32. | Slot_marker frame exchange between reader and SRIX4K. . . . .        | 31 |
| Figure 33. | Select request format . . . . .                                      | 32 |
| Figure 34. | Select response format. . . . .                                      | 32 |
| Figure 35. | Select frame exchange between reader and SRIX4K . . . . .            | 32 |
| Figure 36. | Completion request format . . . . .                                  | 33 |
| Figure 37. | Completion response format. . . . .                                  | 33 |
| Figure 38. | Completion frame exchange between reader and SRIX4K . . . . .        | 33 |
| Figure 39. | Reset_to_inventory request format. . . . .                           | 34 |
| Figure 40. | Reset_to_inventory response format . . . . .                         | 34 |
| Figure 41. | Reset_to_inventory frame exchange between reader and SRIX4K. . . . . | 34 |
| Figure 42. | Read_block request format. . . . .                                   | 35 |
| Figure 43. | Read_block response format . . . . .                                 | 35 |
| Figure 44. | Read_block frame exchange between reader and SRIX4K . . . . .        | 35 |
| Figure 45. | Write_block request format. . . . .                                  | 36 |
| Figure 46. | Write_block response format . . . . .                                | 36 |
| Figure 47. | Write_block frame exchange between reader and SRIX4K . . . . .       | 37 |
| Figure 48. | Get_UID request format . . . . .                                     | 37 |

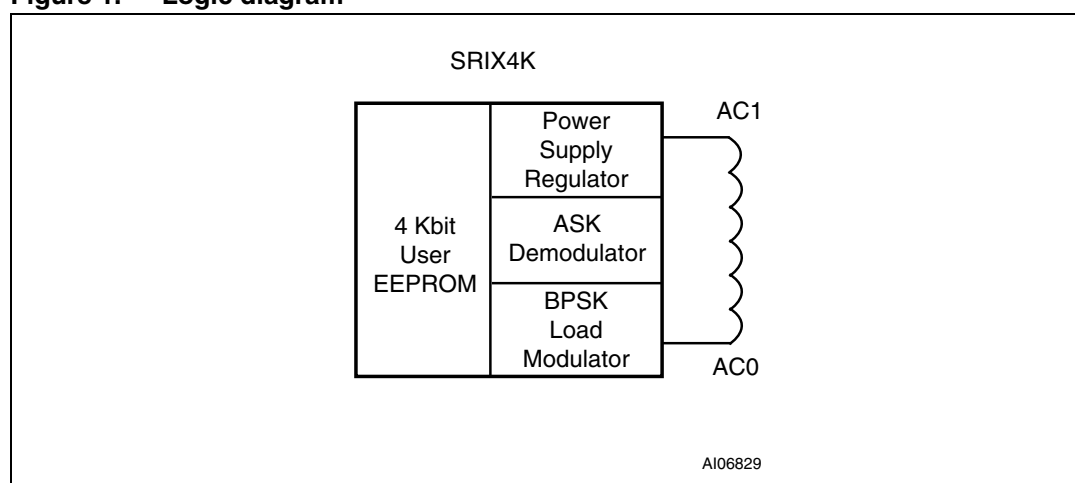
|            |                                                                       |    |
|------------|-----------------------------------------------------------------------|----|
| Figure 49. | Get_UID response format. . . . .                                      | 37 |
| Figure 50. | 64-bit unique identifier of the SRIX4K. . . . .                       | 38 |
| Figure 51. | Get_UID frame exchange between reader and SRIX4K . . . . .            | 38 |
| Figure 52. | SRIX4K synchronous timing, transmit and receive. . . . .              | 41 |
| Figure 53. | Initiate frame exchange between reader and SRIX4K . . . . .           | 44 |
| Figure 54. | Pcall16 frame exchange between reader and SRIX4K . . . . .            | 44 |
| Figure 55. | Slot_marker frame exchange between reader and SRIX4K. . . . .         | 44 |
| Figure 56. | Select frame exchange between reader and SRIX4K . . . . .             | 44 |
| Figure 57. | Completion frame exchange between reader and SRIX4K . . . . .         | 44 |
| Figure 58. | Reset_to_inventory frame exchange between reader and SRIX4K . . . . . | 45 |
| Figure 59. | Read_block frame exchange between reader and SRIX4K . . . . .         | 45 |
| Figure 60. | Write_block frame exchange between reader and SRIX4K . . . . .        | 45 |
| Figure 61. | Get_UID frame exchange between reader and SRIX4K . . . . .            | 45 |

# 1 Description

The SRIX4K is a contactless memory, powered by an externally transmitted radio wave. It contains a 4096-bit user EEPROM fabricated with STMicroelectronics CMOS technology. The memory is organized as 128 blocks of 32 bits. The SRIX4K is accessed via the 13.56 MHz carrier. Incoming data are demodulated and decoded from the received amplitude shift keying (ASK) modulation signal and outgoing data are generated by load variation using bit phase shift keying (BPSK) coding of a 847 kHz subcarrier. The received ASK wave is 10% modulated. The data transfer rate between the SRIX4K and the reader is 106 Kbit/s in both reception and emission modes.

The SRIX4K follows the ISO 14443-2 Type B recommendation for the radio-frequency power and signal interface.

**Figure 1. Logic diagram**



The SRIX4K is specifically designed for short range applications that need secure and reusable products. The SRIX4K includes an anticollision mechanism that allows it to detect and select tags present at the same time within range of the reader. The anticollision is based on a probabilistic scanning method using slot markers. The SRIX4K provides an anti-clone function which allows its authentication. Using the STMicroelectronics single chip coupler, CRX14, it is easy to design a reader with the authentication capability and to build a system with a high level of security.

**Table 1. Signal names**

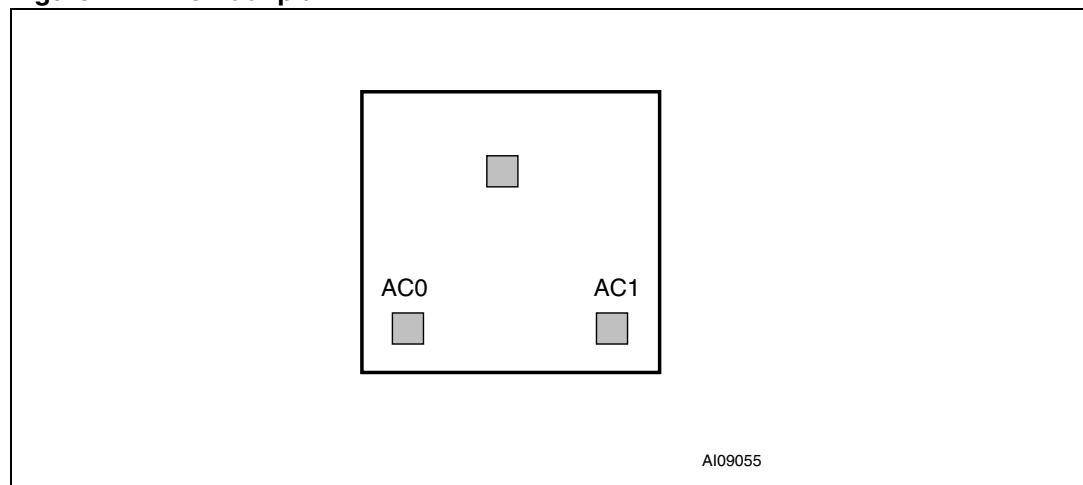
| Signal name | Description  |
|-------------|--------------|
| AC1         | Antenna coil |
| AC0         | Antenna coil |

The SRIX4K contactless EEPROM can be randomly read and written in block mode (each block containing 32 bits). The instruction set includes the following ten commands:

- Read\_block
- Write\_block
- Initiate
- Pcall16
- Slot\_marker
- Select
- Completion
- Reset\_to\_inventory
- Authenticate
- Get\_UID

The SRIX4K memory is organized in three areas, as described in [Figure 12](#). The first area is a resettable OTP (one time programmable) area in which bits can only be switched from 1 to 0. Using a special command, it is possible to erase all bits of this area to 1. The second area provides two 32-bit binary counters which can only be decremented from FFFF FFFFh to 0000 0000h, and gives a capacity of 4,294,967,296 units per counter. The last area is the EEPROM memory. It is accessible by block of 32 bits and includes an auto-erase cycle during each Write\_block command.

**Figure 2. Die floor plan**



## 2 Signal description

### 2.0.1 AC1, AC0

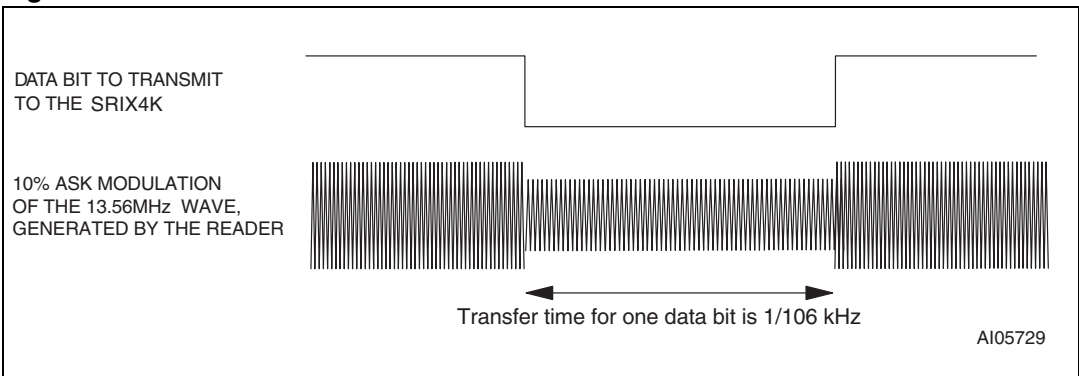
The pads for the antenna coil. AC1 and AC0 must be directly bonded to the antenna.

### 3 Data transfer

#### 3.1 Input data transfer from the reader to the SRIX4K (request frame)

The reader must generate a 13.56 MHz sinusoidal carrier frequency at its antenna, with enough energy to “remote-power” the memory. The energy received at the SRIX4K’s antenna is transformed into a supply voltage by a regulator, and into data bits by the ASK demodulator. For the SRIX4K to decode correctly the information it receives, the reader must 10% amplitude-modulate the 13.56 MHz wave before sending it to the SRIX4K. This is represented in [Figure 3](#). The data transfer rate is 106 Kbits/s.

**Figure 3. 10% ASK modulation of the received wave**



##### 3.1.1 Character transmission format for request frame

The SRIX4K transmits and receives data bytes as 10-bit characters, with the least significant bit ( $b_0$ ) transmitted first, as shown in [Figure 4](#). Each bit duration, an ETU (elementary time unit), is equal to  $9.44 \mu\text{s}$  ( $1/106 \text{ kHz}$ ).

These characters, framed by a start of frame (SOF) and an end of frame (EOF), are put together to form a command frame as shown in [Figure 10](#). A frame includes an SOF, commands, addresses, data, a CRC and an EOF as defined in the ISO 14443-3 Type B Standard. If an error is detected during data transfer, the SRIX4K does not execute the command, but it does not generate an error frame.

**Figure 4. SRIX4K request frame character format**

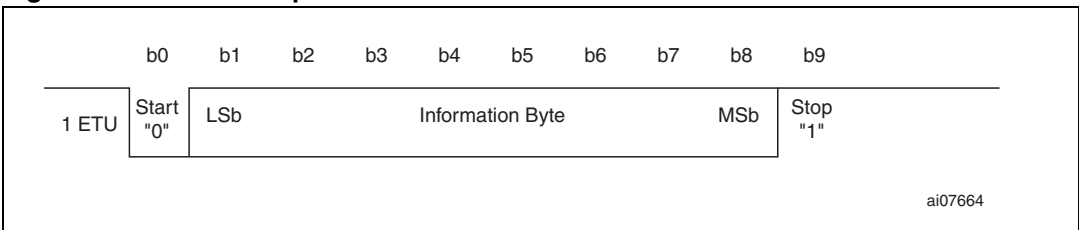


Table 2. Bit description

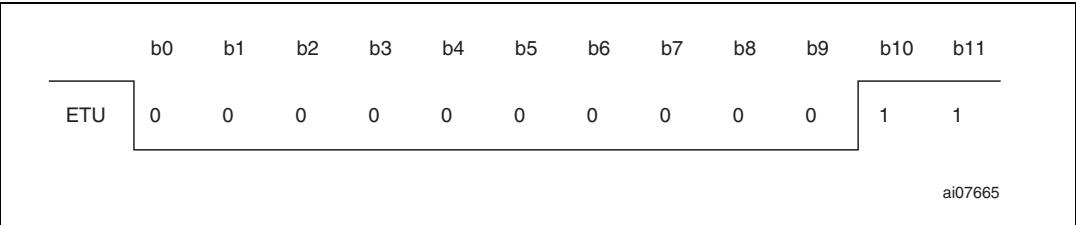
| Bit                              | Description                                      | Value                                                             |
|----------------------------------|--------------------------------------------------|-------------------------------------------------------------------|
| b <sub>0</sub>                   | Start bit used to synchronize the transmission   | b <sub>0</sub> = 0                                                |
| b <sub>1</sub> to b <sub>8</sub> | Information byte (command, address or data)      | The information byte is sent with the least significant bit first |
| b <sub>9</sub>                   | Stop bit used to indicate the end of a character | b <sub>9</sub> = 1                                                |

3.1.2 Request start of frame

The SOF described in [Figure 5](#) is composed of:

- one falling edge,
- followed by 10 ETUs at logic-0,
- followed by a single rising edge,
- followed by at least 2 ETUs (and at most 3) at logic-1.

Figure 5. Request start of frame

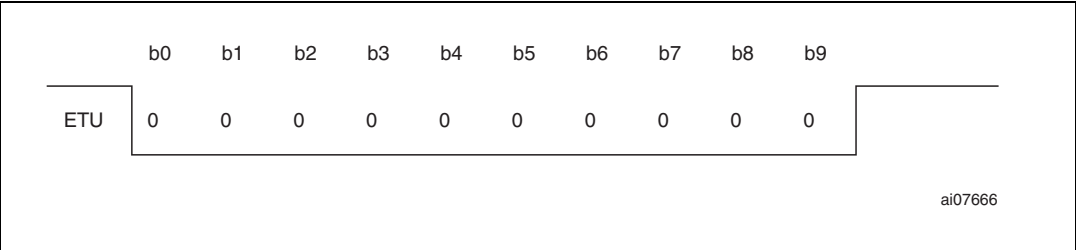


3.1.3 Request end of frame

The EOF shown in [Figure 6](#) is composed of:

- one falling edge,
- followed by 10 ETUs at logic-0,
- followed by a single rising edge.

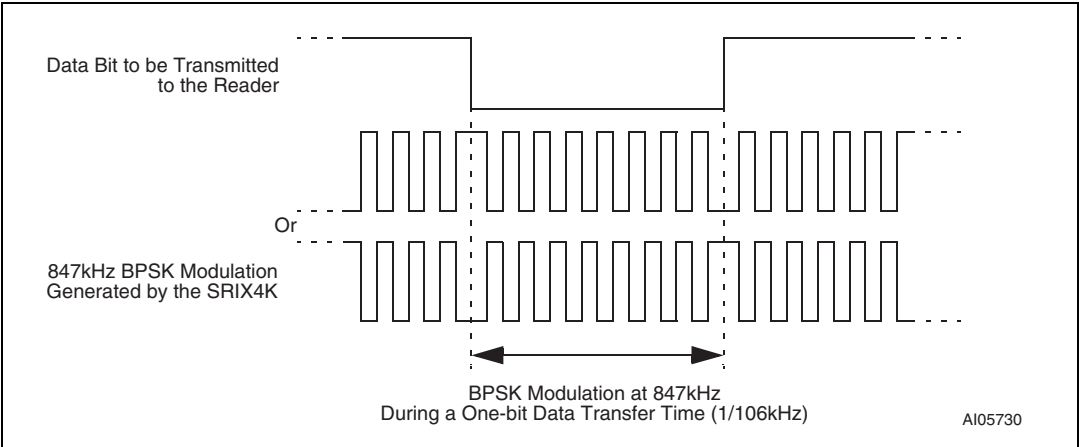
Figure 6. Request end of frame



3.2 Output data transfer from the SRIX4K to the reader (answer frame)

The data bits issued by the SRIX4K use retro-modulation. Retro-modulation is obtained by modifying the SRIX4K current consumption at the antenna (load modulation). The load modulation causes a variation at the reader antenna by inductive coupling. With appropriate detector circuitry, the reader is able to pick up information from the SRIX4K. To improve load-modulation detection, data is transmitted using a BPSK encoded, 847 kHz subcarrier frequency  $f_s$  as shown in [Figure 7](#), and as specified in the ISO 14443-2 Type B Standard.

Figure 7. Wave transmitted using BPSK subcarrier modulation



3.2.1 Character transmission format for answer frame

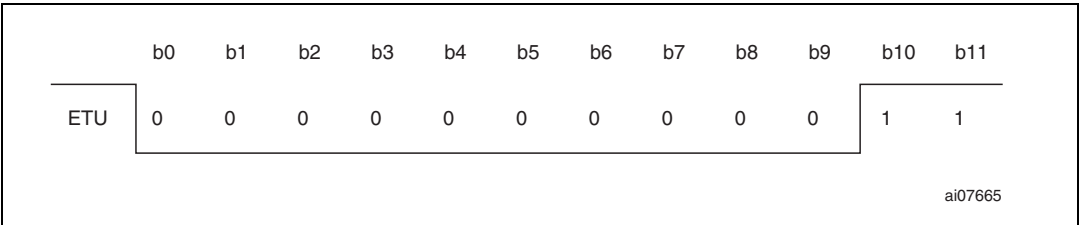
The character format is the same as for input data transfer ([Figure 4](#)). The transmitted frames are made up of an SOF, data, a CRC and an EOF ([Figure 10](#)). As with an input data transfer, if an error occurs, the reader does not issue an error code to the SRIX4K, but it should be able to detect it and manage the situation. The data transfer rate is 106 Kbits/second.

3.2.2 Answer start of frame

The SOF described in [Figure 8](#) is composed of:

- followed by 10 ETUs at logic-0
- followed by 2 ETUs at logic-1

Figure 8. Answer start of frame

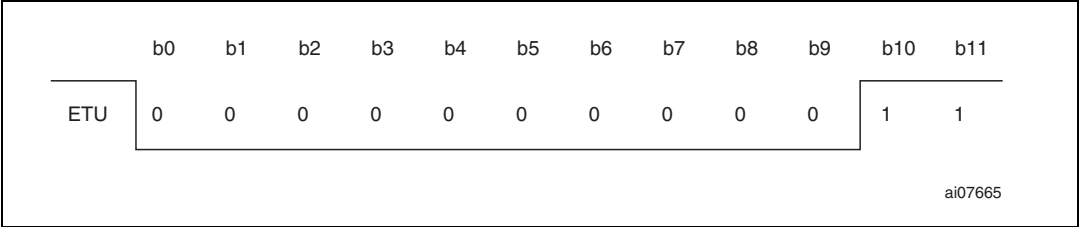


3.2.3 Answer end of frame

The EOF shown in [Figure 9](#) is composed of:

- followed by 10 ETUs at logic-0,
- followed by 2 ETUs at logic-1.

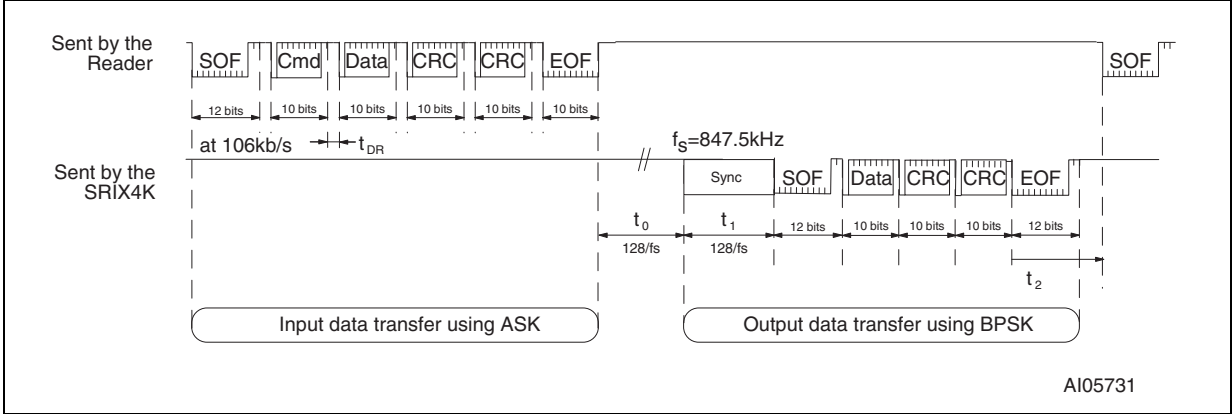
Figure 9. Answer end of frame



3.3 Transmission frame

Between the request data transfer and the Answer data transfer, all ASK and BPSK modulations are suspended for a minimum time of  $t_0 = 128/f_S$ . This delay allows the reader to switch from Transmission to Reception mode. It is repeated after each frame. After  $t_0$ , the 13.56 MHz carrier frequency is modulated by the SRIX4K at 847 kHz for a period of  $t_1 = 128/f_S$  to allow the reader to synchronize. After  $t_1$ , the first phase transition generated by the SRIX4K forms the start bit ('0') of the Answer SOF. After the falling edge of the Answer EOF, the reader waits a minimum time,  $t_2$ , before sending a new request frame to the SRIX4K.

Figure 10. Example of a complete transmission frame



3.4 CRC

The 16-bit CRC used by the SRIX4K is generated in compliance with the ISO 14443 Type B recommendation. For further information, please see [Appendix A](#). The initial register contents are all 1s: FFFFh.

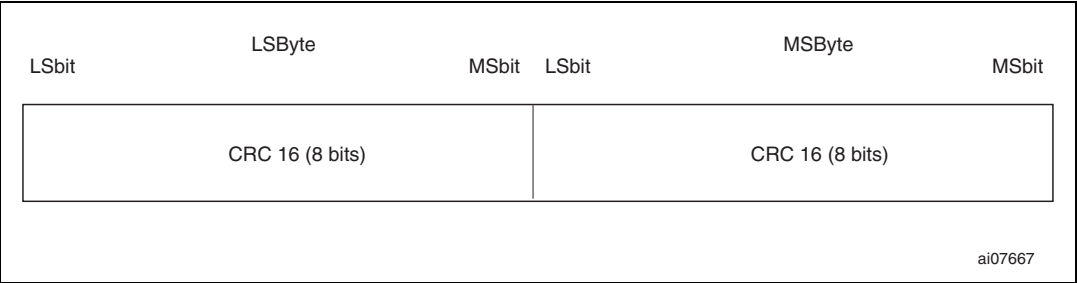
The two-byte CRC is present in every request and in every answer frame, before the EOF. The CRC is calculated on all the bytes between SOF (not included) and the CRC field.

Upon reception of a request from a reader, the SRIX4K verifies that the CRC value is valid. If it is invalid, the SRIX4K discards the frame and does not answer the reader.

Upon reception of an Answer from the SRIX4K, the reader should verify the validity of the CRC. In case of error, the actions to be taken are the reader designer’s responsibility.

The CRC is transmitted with the least significant byte first and each byte is transmitted with the least significant bit first.

Figure 11. CRC transmission rules



## 4 Memory mapping

The SRIX4K is organized as 128 blocks of 32 bits as shown in [Figure 12](#). All blocks are accessible by the Read\_block command. Depending on the write access, they can be updated by the Write\_block command. A Write\_block updates all the 32 bits of the block.

**Figure 12. SRIX4K memory mapping**

| Block Addr | Msb<br>b <sub>31</sub> | 32-bit block                    |                                 |                               |  | Lsb<br>b <sub>0</sub> | Description         |
|------------|------------------------|---------------------------------|---------------------------------|-------------------------------|--|-----------------------|---------------------|
| 0          |                        | b <sub>24</sub> b <sub>23</sub> | b <sub>16</sub> b <sub>15</sub> | b <sub>8</sub> b <sub>7</sub> |  |                       | Resettable OTP bits |
| 1          |                        |                                 |                                 |                               |  |                       |                     |
| 2          |                        |                                 |                                 |                               |  |                       |                     |
| 3          |                        |                                 |                                 |                               |  |                       |                     |
| 4          |                        |                                 |                                 |                               |  |                       |                     |
| 5          |                        |                                 |                                 |                               |  |                       | Count down counter  |
| 6          |                        |                                 |                                 |                               |  |                       |                     |
| 7          |                        |                                 |                                 |                               |  |                       | Lockable EEPROM     |
| 8          |                        |                                 |                                 |                               |  |                       |                     |
| 9          |                        |                                 |                                 |                               |  |                       |                     |
| 10         |                        |                                 |                                 |                               |  |                       |                     |
| 11         |                        |                                 |                                 |                               |  |                       |                     |
| 12         |                        |                                 |                                 |                               |  |                       |                     |
| 13         |                        |                                 |                                 |                               |  |                       |                     |
| 14         |                        |                                 |                                 |                               |  |                       |                     |
| 15         |                        |                                 |                                 |                               |  |                       |                     |
| 16         |                        |                                 |                                 |                               |  |                       | EEPROM              |
| ...        |                        |                                 |                                 |                               |  |                       |                     |
| 127        |                        |                                 |                                 |                               |  |                       |                     |

|     |              |             |                        |                 |
|-----|--------------|-------------|------------------------|-----------------|
| 255 | OTP_Lock_Reg | ST Reserved | Fixed Chip_ID (Option) | System OTP bits |
|-----|--------------|-------------|------------------------|-----------------|

|      |                  |     |
|------|------------------|-----|
| UID0 | 64 bits UID area | ROM |
| UID1 |                  |     |

### 4.1      Resettable OTP area

In this area contains five individual 32-bit Boolean words (see [Figure 13](#) for a map of the area). A Write\_block command will not erase the previous contents of the block as the write cycle is not preceded by an auto-erase cycle. This feature can be used to reset selected bits from 1 to 0. All bits previously at 0 remain unchanged. When the 32 bits of a block are all at 0, the block is empty, and cannot be updated any more. See [Figure 14](#) and [Figure 15](#) for examples of the result of the Write\_block command in the resettable OTP area.

**Figure 13.   Resettable OTP area (addresses 0 to 4)**

| Block address | MSb<br>b31          | b24 b23 | 32-bit block<br>b16 b15 | b8 b7 | LSb<br>b0 | Description           |
|---------------|---------------------|---------|-------------------------|-------|-----------|-----------------------|
| 0             | 32-bit Boolean area |         |                         |       |           | Resettable<br>OTP bit |
| 1             | 32-bit Boolean area |         |                         |       |           |                       |
| 2             | 32-bit Boolean area |         |                         |       |           |                       |
| 3             | 32-bit Boolean area |         |                         |       |           |                       |
| 4             | 32-bit Boolean area |         |                         |       |           |                       |

ai07657b

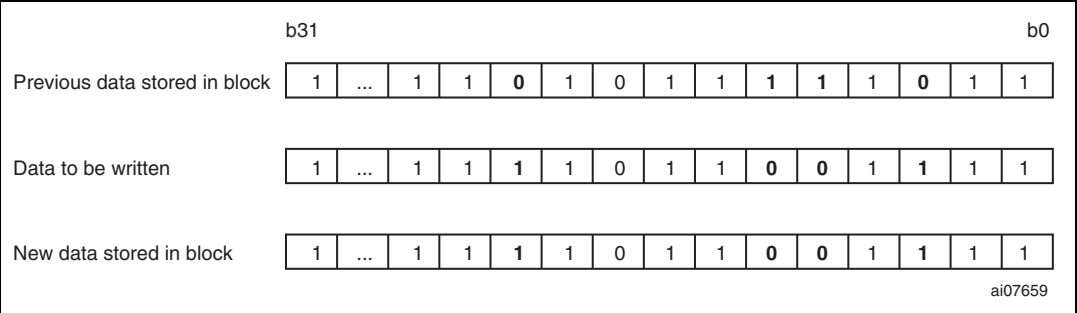
**Figure 14.   Write\_block update in Standard mode (binary format)**

|                               |     |     |   |   |   |   |   |   |   |   |   |   |   |   |   |    |
|-------------------------------|-----|-----|---|---|---|---|---|---|---|---|---|---|---|---|---|----|
|                               | b31 |     |   |   |   |   |   |   |   |   |   |   |   |   |   | b0 |
| Previous data stored in block | 1   | ... | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 1 |    |
| Data to be written            | 1   | ... | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 1 | 1 | 1 |    |
| New data stored in block      | 1   | ... | 1 | 0 | 0 | 1 | 0 | 1 | 1 | 0 | 0 | 1 | 0 | 1 | 1 |    |

ai07658

The five 32-bit blocks making up the resettable OTP area can be erased in one go by adding an auto-erase cycle to the Write\_block command. An auto-erase cycle is added each time the SRIX4K detects a Reload command. The Reload command is implemented through a specific update of the 32-bit binary counter located at block address 6 (see [Section 4.2: 32-bit binary counters](#) for details).

Figure 15. Write\_block update in Reload mode (binary format)



## 4.2 32-bit binary counters

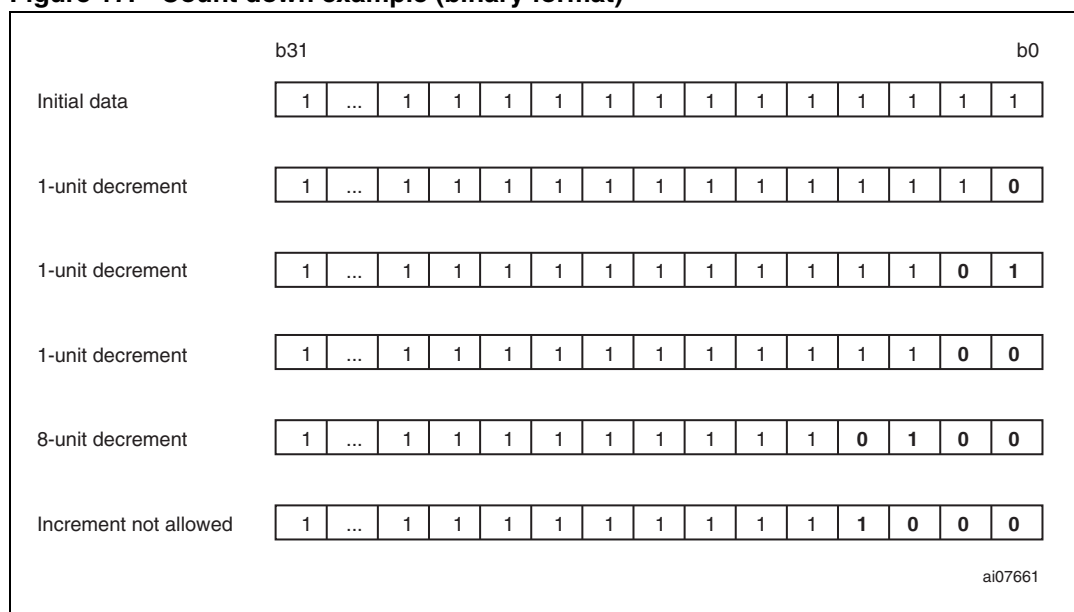
The two 32-bit binary counters located at block addresses 5 and 6, respectively, are used to count down from  $2^{32}$  (4096 million) to 0. The SRIX4K uses dedicated logic that only allows the update of a counter if the new value is lower than the previous one. This feature allows the application to count down by steps of 1 or more. The initial value in Counter 5 is FFFF FFFEh and is FFFF FFFFh in Counter 6. When the value displayed is 0000 0000h, the counter is empty and cannot be reloaded. The counter is updated by issuing the Write\_block command to block address 5 or 6, depending on which counter is to be updated. The Write\_block command writes the new 32-bit value to the counter block address. [Figure 17](#) shows examples of how the counters operate.

The counter programming cycles are protected by automated antitearing logic. This function allows the counter value to be protected in case of power down within the programming cycle. In case of power down, the counter value is not updated and the previous value continues to be stored.

Figure 16. Binary counter (addresses 5 to 6)

| Block Address | MSb<br>b31            | b24 b23 | 32-bit block<br>b16 b15 | b8 b7 | LSb<br>b0 | Description        |
|---------------|-----------------------|---------|-------------------------|-------|-----------|--------------------|
| 5             | 32-bit binary counter |         |                         |       |           | Count down Counter |
| 6             | 32-bit binary counter |         |                         |       |           |                    |

ai07660b

**Figure 17. Count down example (binary format)**

The counter with block address 6 controls the Reload command used to reset the resettable OTP area (addresses 0 to 4). Bits  $b_{31}$  to  $b_{21}$  act as an 11-bit Reload counter; whenever one of these 11 bits is updated, the SRIX4K detects the change and adds an Erase cycle to the Write\_block command for locations 0 to 4 (see [Section 4.1: Resettable OTP area](#)). The Erase cycle remains active until a Power-off or a Select command is issued. The SRIX4K's resettable OTP area can be reloaded up to 2,047 times ( $2^{11}-1$ ).

### 4.3 EEPROM area

The 121 blocks between addresses 7 and 127 are EEPROM blocks of 32 bits each (484 bytes in total). (See [Figure 18](#) for a map of the area.) These blocks can be accessed using the Read\_block and Write\_block commands. The Write\_block command for the EEPROM area always includes an auto-erase cycle prior to the write cycle.

Blocks 7 to 15 can be write-protected. Write access is controlled by the 8 bits of the OTP\_Lock\_Reg located at block address 255 (see [Section 4.4.1: OTP\\_Lock\\_Reg](#) for details). Once protected, these blocks (7 to 15) cannot be unprotected.

**Figure 18. EEPROM (addresses 7 to 127)**

| Block address | MSb<br>b31 | b24 b23 | 32-bit block<br>b16 b15 | b8 b7 | LSb<br>b0 | Description        |
|---------------|------------|---------|-------------------------|-------|-----------|--------------------|
| 7             |            |         | User area               |       |           | Lockable<br>EEPROM |
| 8             |            |         | User area               |       |           |                    |
| 9             |            |         | User area               |       |           |                    |
| 10            |            |         | User area               |       |           |                    |
| 11            |            |         | User area               |       |           |                    |
| 12            |            |         | User area               |       |           |                    |
| 13            |            |         | User area               |       |           |                    |
| 14            |            |         | User area               |       |           |                    |
| 15            |            |         | User area               |       |           |                    |
| 16            |            |         | User area               |       |           | EEPROM             |
| ...           |            |         | User area               |       |           |                    |
| 127           |            |         | User area               |       |           |                    |

Ai07662c

## 4.4 System area

This area is used to modify the settings of the SRIX4K. It contains 3 registers: OTP\_Lock\_Reg, Fixed Chip\_ID and ST Reserved. See [Figure 19](#) for a map of this area.

A Write\_block command in this area will not erase the previous contents. Selected bits can thus be set from 1 to 0. All bits previously at 0 remain unchanged. Once all the 32 bits of a block are at 0, the block is empty and cannot be updated any more.

**Figure 19. System area**

| Block address | MSb<br>b31   | b24 b23 | 32-bit block<br>b16 b15 | b8 b7                     | LSb<br>b0 | Description |
|---------------|--------------|---------|-------------------------|---------------------------|-----------|-------------|
| 255           | OTP_Lock_Reg |         | ST reserved             | Fixed Chip_ID<br>(Option) |           | OTP         |

ai07663b

#### 4.4.1 OTP\_Lock\_Reg

The 8 bits,  $b_{31}$  to  $b_{24}$ , of the System area (block address 255) are used as OTP\_Lock\_Reg bits in the SRIX4K. They control the write access to the 9 EEPROM blocks with addresses 7 to 15 as follows:

- When  $b_{24}$  is at 0, blocks 7 and 8 are write-protected
- When  $b_{25}$  is at 0, block 9 is write-protected
- When  $b_{26}$  is at 0, block 10 is write-protected
- When  $b_{27}$  is at 0, block 11 is write-protected
- When  $b_{28}$  is at 0, block 12 is write-protected
- When  $b_{29}$  is at 0, block 13 is write-protected
- When  $b_{30}$  is at 0, block 14 is write-protected
- When  $b_{31}$  is at 0, block 15 is write-protected.

The OTP\_Lock\_Reg bits cannot be erased. Once write-protected, EEPROM blocks behave like ROM blocks and cannot be unprotected.

#### 4.4.2 Fixed Chip\_ID (Option)

The SRIX4K is provided with an anticollision feature based on a random 8-bit Chip\_ID. Prior to selecting an SRIX4K, an anticollision sequence has to be run to search for the Chip\_ID of the SRIX4K. This is a very flexible feature, however the searching loop requires time to run.

For some applications, much time could be saved by knowing the value of the SRIX4K Chip\_ID beforehand, so that the SRIX4K can be identified and selected directly without having to run an anticollision sequence. This is why the SRIX4K was designed with an optional mask setting used to program a fixed 8-bit Chip\_ID to bits  $b_7$  to  $b_0$  of the system area. When the fixed Chip\_ID option is used, the random Chip\_ID function is disabled.

## 5 SRIX4K operation

All commands, data and CRC are transmitted to the SRIX4K as 10-bit characters using ASK modulation. The start bit of the 10 bits,  $b_0$ , is sent first. The command frame received by the SRIX4K at the antenna is demodulated by the 10% ASK demodulator, and decoded by the internal logic. Prior to any operation, the SRIX4K must have been selected by a Select command. Each frame transmitted to the SRIX4K must start with a start of frame, followed by one or more data characters, two CRC bytes and the final end of frame. When an invalid frame is decoded by the SRIX4K (wrong command or CRC error), the memory does not return any error code.

When a valid frame is received, the SRIX4K may have to return data to the reader. In this case, data is returned using BPSK encoding, in the form of 10-bit characters framed by an SOF and an EOF. The transfer is ended by the SRIX4K sending the 2 CRC bytes and the EOF.

## 6 SRIX4K states

The SRIX4K can be switched into different states. Depending on the current state of the SRIX4K, its logic will only answer to specific commands. These states are mainly used during the anticollision sequence, to identify and to access the SRIX4K in a very short time. The SRIX4K provides 6 different states, as described in the following paragraphs and in [Figure 20](#).

### 6.1 Power-off state

The SRIX4K is in Power-off state when the electromagnetic field around the tag is not strong enough. In this state, the SRIX4K does not respond to any command.

### 6.2 Ready state

When the electromagnetic field is strong enough, the SRIX4K enters the Ready state. After power-up, the Chip\_ID is initialized with a random value. The whole logic is reset and remains in this state until an Initiate() command is issued. Any other command will be ignored by the SRIX4K.

### 6.3 Inventory state

The SRIX4K switches from the Ready to the Inventory state after an Initiate() command has been issued. In Inventory state, the SRIX4K will respond to any anticollision commands: Initiate(), Pcall16() and Slot\_marker(), and then remain in the Inventory state. It will switch to the Selected state after a Select(Chip\_ID) command is issued, if the Chip\_ID in the command matches its own. If not, it will remain in Inventory state.

### 6.4 Selected state

In Selected state, the SRIX4K is active and responds to all Read\_block(), Write\_block(), Authenticate() and Get\_UID() commands. When an SRIX4K has entered the Selected state, it no longer responds to anticollision commands. So that the reader can access another tag, the SRIX4K can be switched to the Deselected state by sending a Select(Chip\_ID2) with a Chip\_ID that does not match its own, or it can be placed in Deactivated state by issuing a Completion() command. Only one SRIX4K can be in Selected state at a time.

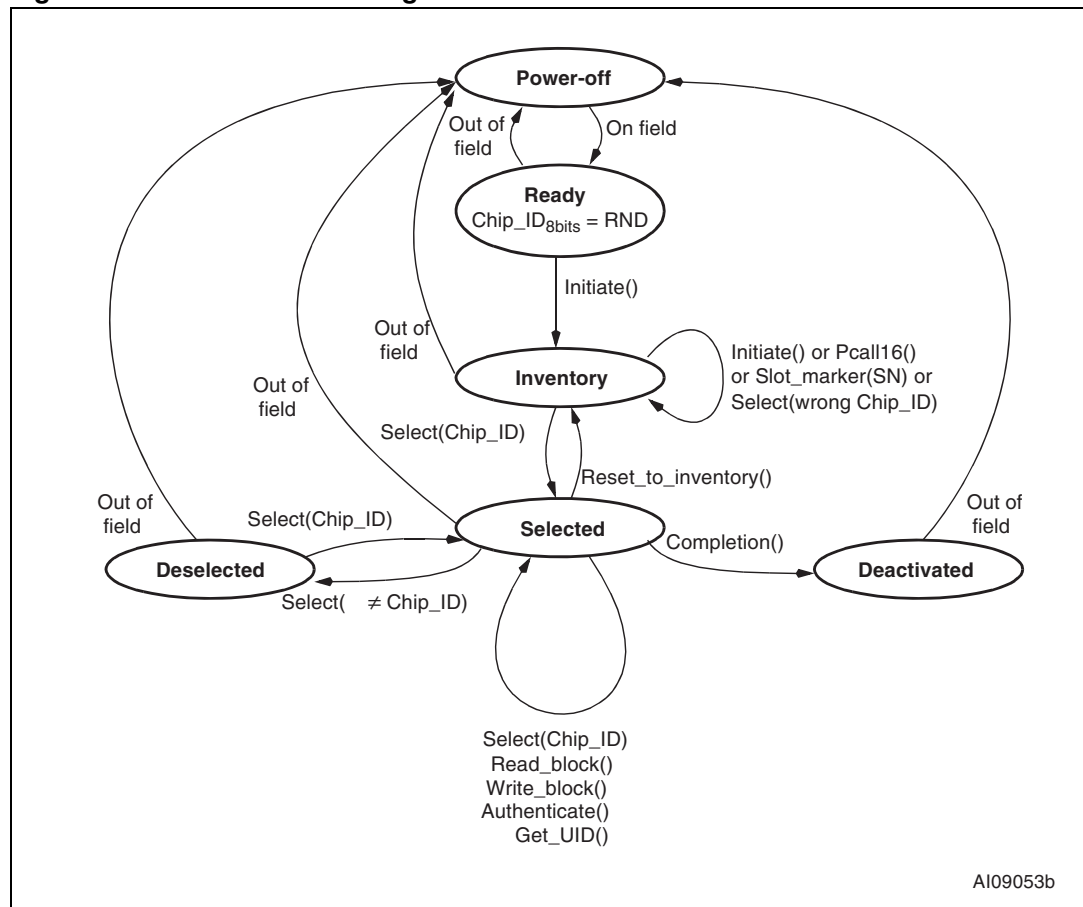
### 6.5 Deselected state

Once the SRIX4K is in Deselected state, only a Select(Chip\_ID) command with a Chip\_ID matching its own can switch it back to Selected state. All other commands are ignored.

### 6.6 Deactivated state

When in this state, the SRIX4K can only be turned off. All commands are ignored.

Figure 20. State transition diagram



## 7 Anticollision

The SRIX4K provides an anticollision mechanism that searches for the Chip\_ID of each device that is present in the reader field range. When known, the Chip\_ID is used to select an SRIX4K individually, and access its memory. The anticollision sequence is managed by the reader through a set of commands described in [Section 5: SRIX4K operation](#):

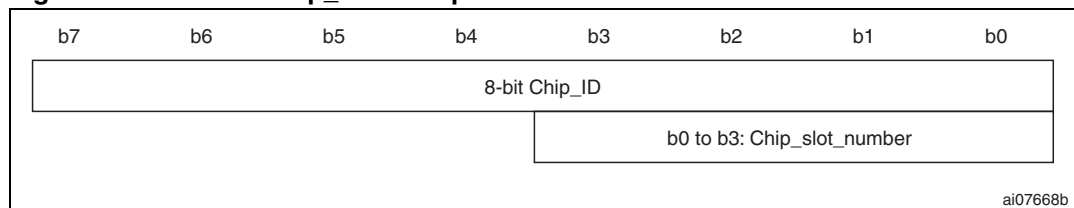
- Initiate()
- Pcall16()
- Slot\_marker().

The reader is the master of the communication with one or more SRIX4K device(s). It initiates the tag communication activity by issuing an Initiate(), Pcall16() or Slot\_marker() command to prompt the SRIX4K to answer. During the anticollision sequence, it might happen that two or more SRIX4K devices respond simultaneously, so causing a collision. The command set allows the reader to handle the sequence, to separate SRIX4K transmissions into different time slots. Once the anticollision sequence has completed, SRIX4K communication is fully under the control of the reader, allowing only one SRIX4K to transmit at a time.

The Anticollision scheme is based on the definition of time slots during which the SRIX4K devices are invited to answer with minimum identification data: the Chip\_ID. The number of slots is fixed at 16 for the Pcall16() command. For the Initiate() command, there is no slot and the SRIX4K answers after the command is issued. SRIX4K devices are allowed to answer only once during the anticollision sequence. Consequently, even if there are several SRIX4K devices present in the reader field, there will probably be a slot in which only one SRIX4K answers, allowing the reader to capture its Chip\_ID. Using the Chip\_ID, the reader can then establish a communication channel with the identified SRIX4K. The purpose of the anticollision sequence is to allow the reader to select one SRIX4K at a time.

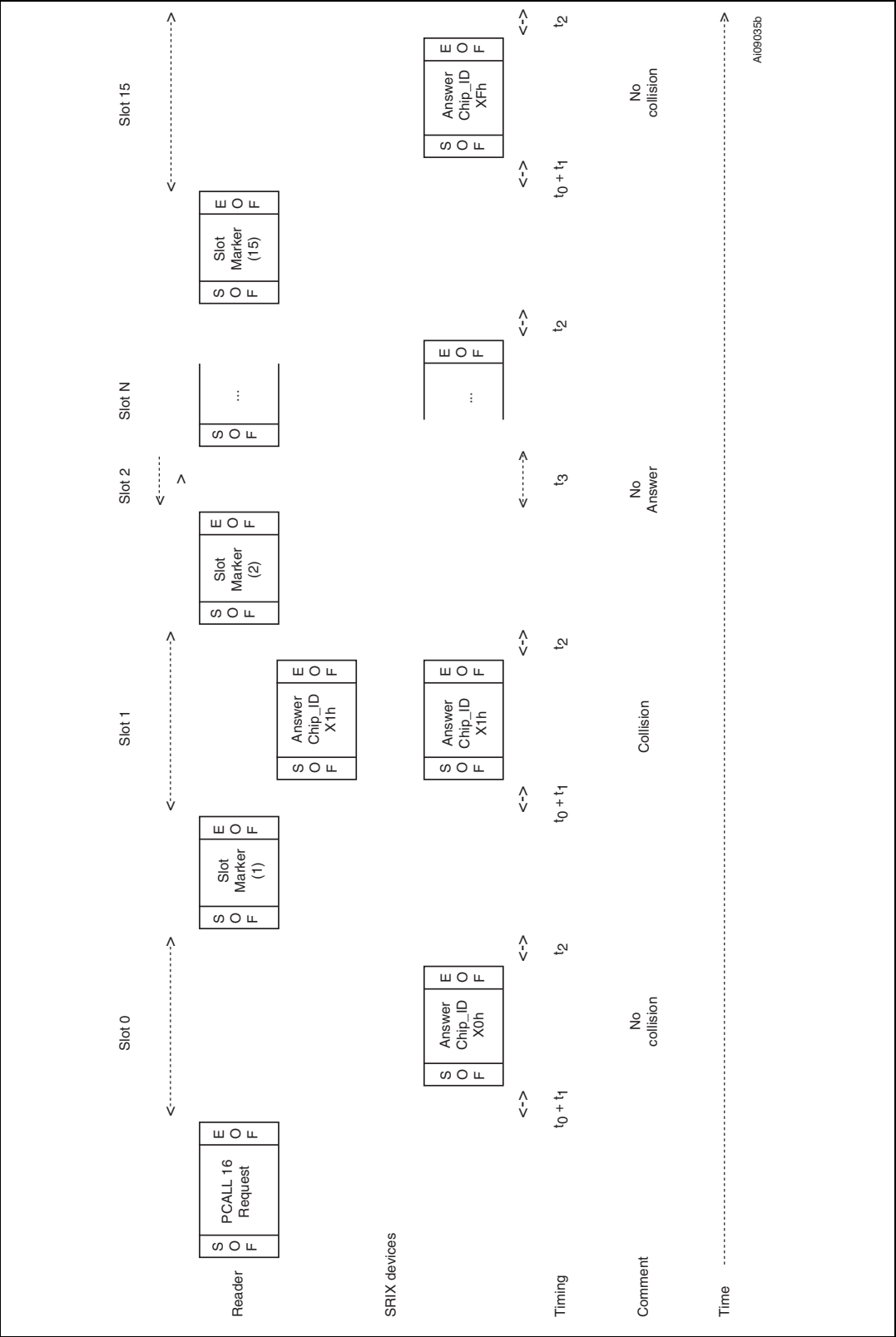
The SRIX4K is given an 8-bit Chip\_ID value used by the reader to select only one among up to 256 tags present within its field range. The Chip\_ID is initialized with a random value during the Ready state, or after an Initiate() command in the Inventory state. The four least significant bits ( $b_0$  to  $b_3$ ) of the Chip\_ID are also known as the Chip\_slot\_number. This 4-bit value is used by the Pcall16() and Slot\_marker() commands during the anticollision sequence in the Inventory state.

**Figure 21. SRIX4K Chip\_ID description**



Each time the SRIX4K receives a Pcall16() command, the Chip\_slot\_number is given a new 4-bit random value. If the new value is 0000<sub>b</sub>, the SRIX4K returns its whole 8-bit Chip\_ID in its answer to the Pcall16() command. The Pcall16() command is also used to define the slot number 0 of the anticollision sequence. When the SRIX4K receives the Slot\_marker(SN) command, it compares its Chip\_slot\_number with the Slot\_number parameter (SN). If they match, the SRIX4K returns its Chip\_ID as a response to the command. If they do not, the SRIX4K does not answer. The Slot\_marker(SN) command is used to define all the anticollision slot numbers from 1 to 15.

Figure 22. Description of a possible anticollision sequence



1. The value X in the Answer Chip\_ID means a random hexadecimal character from 0 to F.

## 7.1 Description of an anticollision sequence

The anticollision sequence is initiated by the Initiate() command which triggers all the SRIX4K devices that are present in the reader field range, and that are in Inventory state. Only SRIX4K devices in Inventory state will respond to the Pcall16() and Slot\_marker(SN) anticollision commands.

A new SRIX4K introduced in the field range during the anticollision sequence will not be taken into account as it will not respond to the Pcall16() or Slot\_marker(SN) command (Ready state). To be considered during the anticollision sequence, it must have received the Initiate() command and entered the Inventory state.

[Table 3](#) shows the elements of a standard anticollision sequence. (See [Figure 23](#) for an example.)

**Table 3. Standard anticollision sequence**

|         |         |                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|---------|---------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Step 1  | Init:   | Send Initiate().<br>– If no answer is detected, go to step1.<br>– If only 1 answer is detected, select and access the SRIX4K. After accessing the SRIX4K, deselect the tag and go to step1.<br>– If a collision (many answers) is detected, go to step2.                                                                                                                                                                                              |
| Step 2  | Slot 0  | Send Pcall16().<br>– If no answer or collision is detected, go to step3.<br>– If 1 answer is detected, store the Chip_ID, Send Select() and go to step3.                                                                                                                                                                                                                                                                                              |
| Step 3  | Slot 1  | Send Slot_marker(1).<br>– If no answer or collision is detected, go to step4.<br>– If 1 answer is detected, store the Chip_ID, Send Select() and go to step4.                                                                                                                                                                                                                                                                                         |
| Step 4  | Slot 2  | Send Slot_marker(2).<br>– If no answer or collision is detected, go to step5.<br>– If 1 answer is detected, store the Chip_ID, Send Select() and go to step5.                                                                                                                                                                                                                                                                                         |
| Step N  | Slop N  | Send Slot_marker(3 up to 14) ...<br>– If no answer or collision is detected, go to stepN+1.<br>– If 1 answer is detected, store the Chip_ID, Send Select() and go to stepN+1.                                                                                                                                                                                                                                                                         |
| Step 17 | Slot 15 | Send Slot_marker(15).<br>– If no answer or collision is detected, go to step18.<br>– If 1 answer is detected, store the Chip_ID, Send Select() and go to step18.                                                                                                                                                                                                                                                                                      |
| Step 18 |         | All the slots have been generated and the Chip_ID values should be stored into the reader memory. Issue the Select(Chip_ID) command and access each identified SRIX4K one by one. After accessing each SRIX4K, switch them into Deselected or Deactivated state, depending on the application needs.<br>– If collisions were detected between Step2 and Step17, go to Step2.<br>– If no collision was detected between Step2 and Step17, go to Step1. |

After each Slot\_marker() command, there may be several, one or no answers from the SRIX4K devices. The reader must handle all the cases and store all the Chip\_IDs, correctly decoded. At the end of the anticollision sequence, after Slot\_marker(15), the reader can start working with one SRIX4K by issuing a Select() command containing the desired Chip\_ID. If a collision is detected during the anticollision sequence, the reader has to generate a new sequence in order to identify all unidentified SRIX4K devices in the field. The anticollision sequence can stop when all SRIX4K devices have been identified.

**Figure 23. Example of an anticollision sequence**

| Command        | Tag 1<br>Chip_ID | Tag 2<br>Chip_ID | Tag 3<br>Chip_ID | Tag 4<br>Chip_ID | Tag 5<br>Chip_ID | Tag 6<br>Chip_ID | Tag 7<br>Chip_ID | Tag 8<br>Chip_ID | Comments                                                                  |
|----------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|------------------|---------------------------------------------------------------------------|
| READY State    | 28h              | 75h              | 40h              | 01h              | 02h              | FEh              | A9h              | 7Ch              | Each tag gets a random Chip_ID                                            |
| INITIATE ()    | 40h              | 13h              | 3Fh              | 4Ah              | 50h              | 48h              | 52h              | 7Ch              | Each tag get a new random Chip_ID.<br>All tags answer: collisions         |
| PCALL16()      | 45h              | 12h              | 30h              | 43h              | 55h              | 43h              | 53h              | 73h              | All CHIP_SLOT_NUMBERS get<br>a new random value<br>Slot0: only one answer |
| SELECT(30h)    |                  |                  | 30h              |                  |                  |                  |                  |                  | Tag3 is identified                                                        |
| SLOT_MARKER(1) |                  |                  |                  |                  |                  |                  |                  |                  | Slot1: no answer                                                          |
| SLOT_MARKER(2) |                  | 12h              |                  |                  |                  |                  |                  |                  | Slot2: only one answer                                                    |
| SELECT(12h)    |                  | 12h              |                  |                  |                  |                  |                  |                  | Tag2 is identified                                                        |
| SLOT_MARKER(3) |                  |                  |                  | 43h              |                  | 43h              | 53h              | 73h              | Slot3: collisions                                                         |
| SLOT_MARKER(4) |                  |                  |                  |                  |                  |                  |                  |                  | Slot4: no answer                                                          |
| SLOT_MARKER(5) | 45h              |                  |                  |                  | 55h              |                  |                  |                  | Slot5: collisions                                                         |
| SLOT_MARKER(6) |                  |                  |                  |                  |                  |                  |                  |                  | Slot6: no answer                                                          |
| SLOT_MARKER(N) |                  |                  |                  |                  |                  |                  |                  |                  | SlotN: no answer                                                          |
| SLOT_MARKER(F) |                  |                  |                  |                  |                  |                  |                  |                  | SlotF: no answer                                                          |
| PCALL16()      | 40h              |                  |                  | 41h              | 53h              | 42h              | 50h              | 74h              | All CHIP_SLOT_NUMBERS get<br>a new random value<br>Slot0: collisions      |
| SLOT_MARKER(1) | 40h              |                  |                  | 41h              |                  |                  | 50h              |                  | Slot1: only one answer                                                    |
| SELECT(41h)    |                  |                  |                  | 41h              |                  |                  |                  |                  | Tag4 is identified                                                        |
| SLOT_MARKER(2) |                  |                  |                  |                  |                  | 42h              |                  |                  | Slot2: only one answer                                                    |
| SELECT(42h)    |                  |                  |                  |                  |                  | 42h              |                  |                  | Tag6 is identified                                                        |
| SLOT_MARKER(3) |                  |                  |                  |                  | 53h              |                  |                  |                  | Slot3: only one answer                                                    |
| SELECT(53h)    |                  |                  |                  |                  | 53h              |                  |                  |                  | Tag5 is identified                                                        |
| SLOT_MARKER(4) |                  |                  |                  |                  |                  |                  |                  | 74h              | Slot4: only one answer                                                    |
| SELECT(74h)    |                  |                  |                  |                  |                  |                  |                  | 74h              | Tag8 is identified                                                        |
| SLOT_MARKER(N) |                  |                  |                  |                  |                  |                  |                  |                  | SlotN: no answer                                                          |
| PCALL16()      | 41h              |                  |                  |                  |                  |                  | 50h              |                  | All CHIP_SLOT_NUMBERS get<br>a new random value<br>Slot0: only one answer |
| SELECT(50h)    |                  |                  |                  |                  |                  |                  | 50h              |                  | Tag7 is identified                                                        |
| SLOT_MARKER(1) | 41h              |                  |                  |                  |                  |                  |                  |                  | Slot1: only one answer but already<br>found for tag4                      |
| SLOT_MARKER(N) |                  |                  |                  |                  |                  |                  |                  |                  | SlotN: no answer                                                          |
| PCALL16()      | 43h              |                  |                  |                  |                  |                  |                  |                  | All CHIP_SLOT_NUMBERS get<br>a new random value<br>Slot0: only one answer |
| SLOT_MARKER(3) | 43h              |                  |                  |                  |                  |                  |                  |                  | Slot3: only one answer                                                    |
| SELECT(43h)    | 43h              |                  |                  |                  |                  |                  |                  |                  | Tag1 is identified                                                        |
|                |                  |                  |                  |                  |                  |                  |                  |                  | All tags are identified                                                   |

ai07669

## 8 Anti-clone function

The SRIX4K provides an anti-clone function that allows the application to authentication the device. This function uses reserved data that is stored in the SRIX4K memory at its time of manufacture.

The Authentication system is based on a proprietary challenge/response mechanism which allows the application software to authenticate any member of the secure memory tag SRXxxx family from STMicroelectronics (of which the SRIX4K is the prime example). A reader system, based on the ST CRX14 chip coupler, can check each SRIX4K tag for authenticity, and protect the application system against silicon copies or emulators.

A complete description of the Authentication system is available under non disclosure agreement (NDA) with STMicroelectronics. For more details about this SRIX4K function, please contact your nearest STMicroelectronics sales office.

## 9 SRIX4K commands

See the paragraphs below for a detailed description of the commands available on the SRIX4K. The commands and their hexadecimal codes are summarized in [Table 4](#). A brief is given in [Appendix B](#).

**Table 4. Command code**

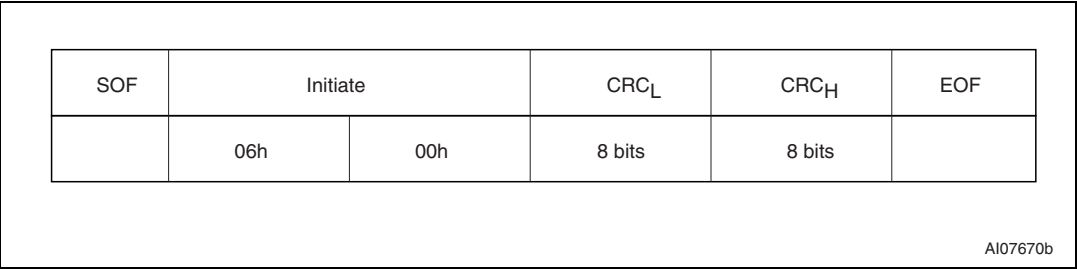
| Hexadecimal Code | Command                 |
|------------------|-------------------------|
| 06h-00h          | Initiate()              |
| 06h-04h          | Pcall16()               |
| x6h              | Slot_marker (SN)        |
| 08h              | Read_block(Addr)        |
| 09h              | Write_block(Addr, Data) |
| 0Ah              | Authenticate(RND)       |
| 0Bh              | Get_UID()               |
| 0Ch              | Reset_to_inventory      |
| 0Eh              | Select(Chip_ID)         |
| 0Fh              | Completion()            |

# 9.1 Initiate() command

Command code = 06h - 00h

Initiate() is used to initiate the anticollision sequence of the SRIX4K. On receiving the Initiate() command, all SRIX4K devices in Ready state switch to Inventory state, set a new 8-bit Chip\_ID random value, and return their Chip\_ID value. This command is useful when only one SRIX4K in Ready state is present in the reader field range. It speeds up the Chip\_ID search process. The Chip\_slot\_number is not used during Initiate() command access.

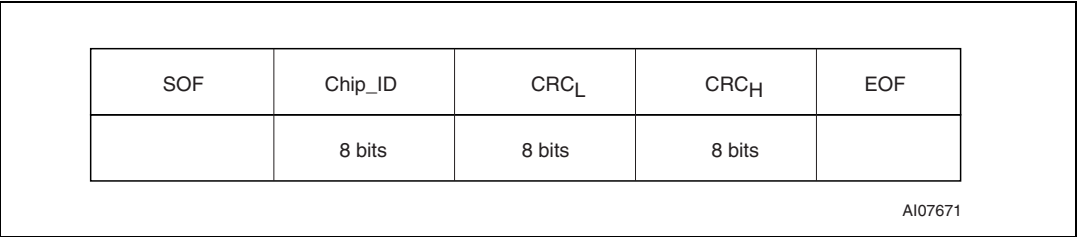
Figure 24. Initiate request format



Request parameter:

- No parameter

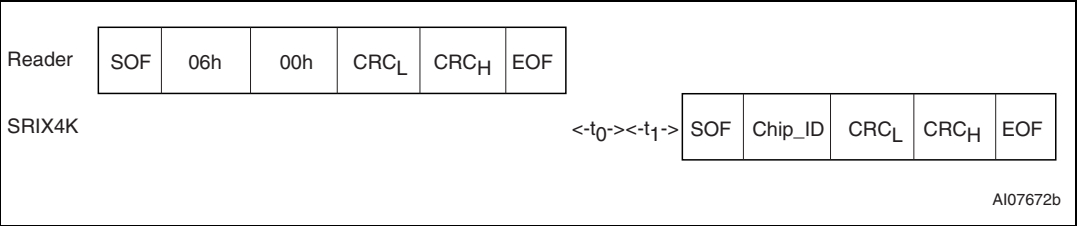
Figure 25. Initiate response format



Response parameter:

- Chip\_ID of the SRIX4K

Figure 26. Initiate frame exchange between reader and SRIX4K



## 9.2 Pcall16() command

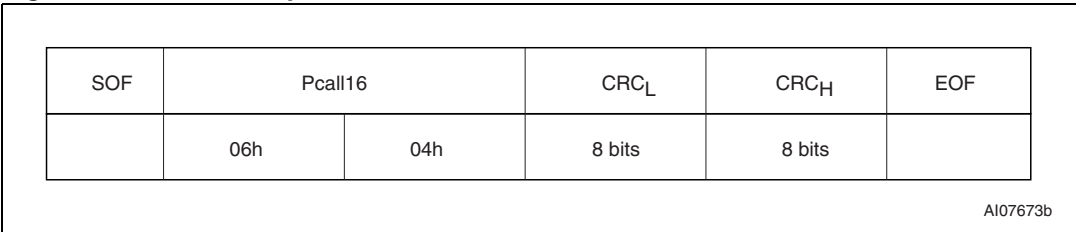
Command code = 06h - 04h

The SRIX4K must be in Inventory state to interpret the Pcall16() command.

On receiving the Pcall16() command, the SRIX4K first generates a new random Chip\_slot\_number value (in the 4 least significant bits of the Chip\_ID). Chip\_slot\_number can take on a value between 0 and 15 (1111<sub>b</sub>). The value is retained until a new Pcall16() or Initiate() command is issued, or until the SRIX4K is powered off. The new Chip\_slot\_number value is then compared with the value 0000<sub>b</sub>. If they match, the SRIX4K returns its Chip\_ID value. If not, the SRIX4K does not send any response.

The Pcall16() command, used together with the Slot\_marker() command, allows the reader to search for all the Chip\_IDs when there are more than one SRIX4K device in Inventory state present in the reader field range.

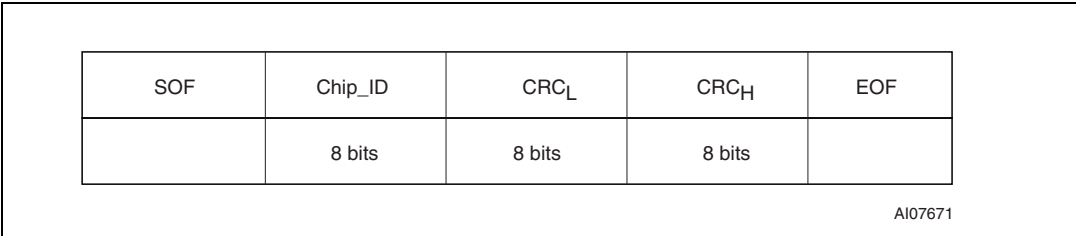
**Figure 27. Pcall16 request format**



Request parameter:

- No parameter

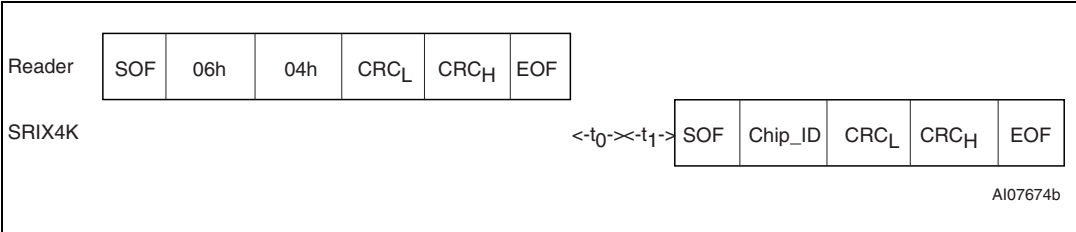
**Figure 28. Pcall16 response format**



Response parameter:

- Chip\_ID of the SRIX4K

**Figure 29. Pcall16 frame exchange between reader and SRIX4K**



### 9.3 Slot\_marker(SN) command

Command code = x6h

The SRIX4K must be in Inventory state to interpret the Slot\_marker(SN) command.

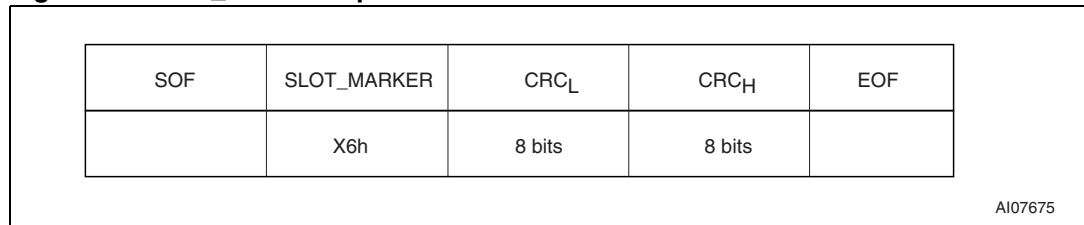
The Slot\_marker byte code is divided into two parts:

- $b_3$  to  $b_0$ : 4-bit command code with fixed value 6.
- $b_7$  to  $b_4$ : 4 bits known as the Slot\_number (SN). They assume a value between 1 and 15. The value 0 is reserved by the Pcall16() command.

On receiving the Slot\_marker() command, the SRIX4K compares its Chip\_slot\_number value with the Slot\_number value given in the command code. If they match, the SRIX4K returns its Chip\_ID value. If not, the SRIX4K does not send any response.

The Slot\_marker() command, used together with the Pcall16() command, allows the reader to search for all the Chip\_IDs when there are more than one SRIX4K device in Inventory state present in the reader field range.

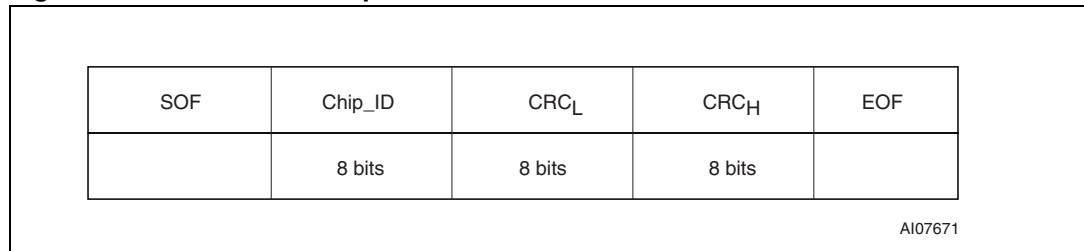
**Figure 30. Slot\_marker request format**



Request parameter:

- x: Slot number

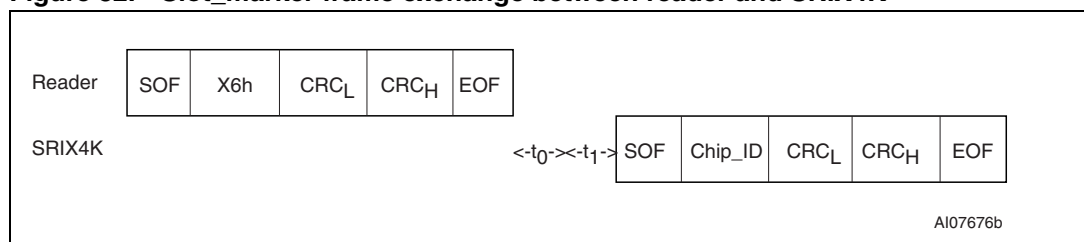
**Figure 31. Slot\_marker response format**



Response parameters:

- Chip\_ID of the SRIX4K

**Figure 32. Slot\_marker frame exchange between reader and SRIX4K**

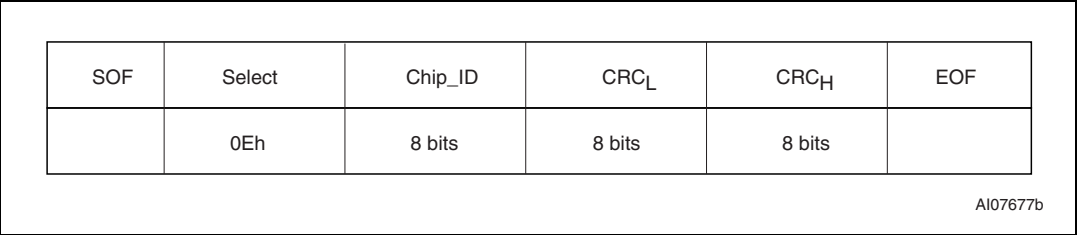


9.4 Select(Chip\_ID) command

Command code = 0Eh

The Select() command allows the SRIX4K to enter the Selected state. Until this command is issued, the SRIX4K will not accept any other command, except for Initiate(), Pcall16() and Slot\_marker(). The Select() command returns the 8 bits of the Chip\_ID value. An SRIX4K in Selected state, that receives a Select() command with a Chip\_ID that does not match its own is automatically switched to Deselected state.

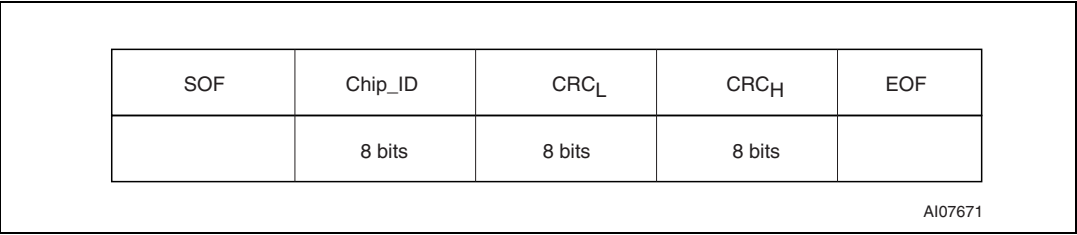
Figure 33. Select request format



Request parameter:

- 8-bit Chip\_ID stored during the anticollision sequence

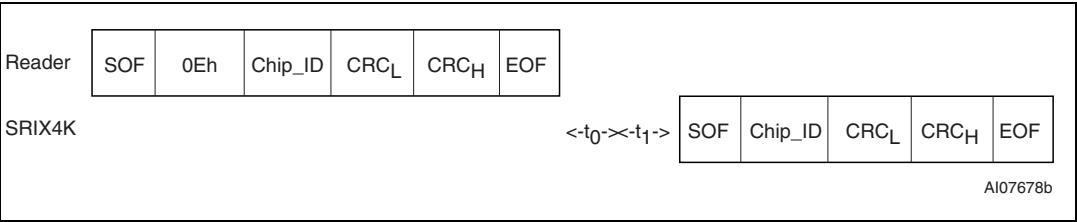
Figure 34. Select response format



Response parameters:

- Chip\_ID of the selected tag. Must be equal to the transmitted Chip\_ID

Figure 35. Select frame exchange between reader and SRIX4K



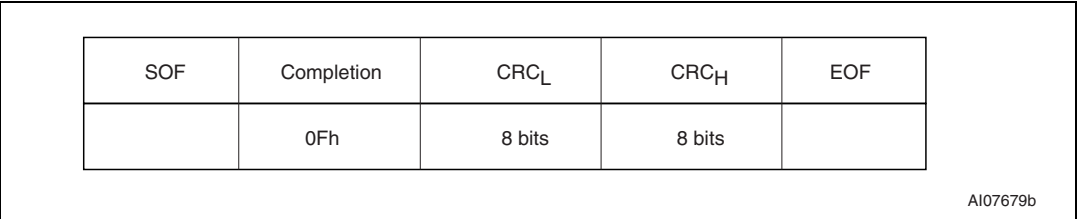
9.5 Completion() command

Command code = 0Fh

On receiving the Completion() command, an SRIX4K in Selected state switches to Deactivated state and stops decoding any new commands. The SRIX4K is then locked in this state until a complete reset (tag out of the field range). A new SRIX4K can thus be accessed through a Select() command without having to remove the previous one from the field. The Completion() command does not generate a response.

All SRIX4K devices not in Selected state ignore the Completion() command.

Figure 36. Completion request format



Request parameters:

- No parameter

Figure 37. Completion response format

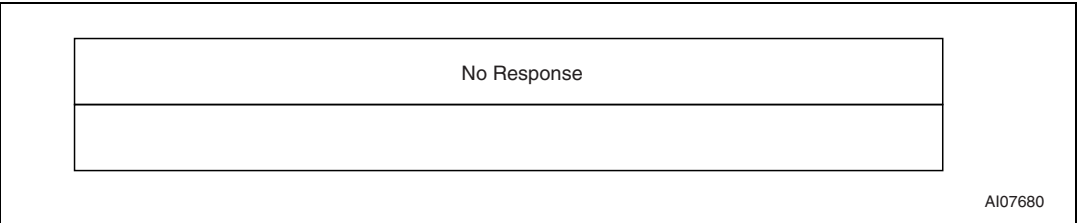


Figure 38. Completion frame exchange between reader and SRIX4K



9.6      **Reset\_to\_inventory() command**

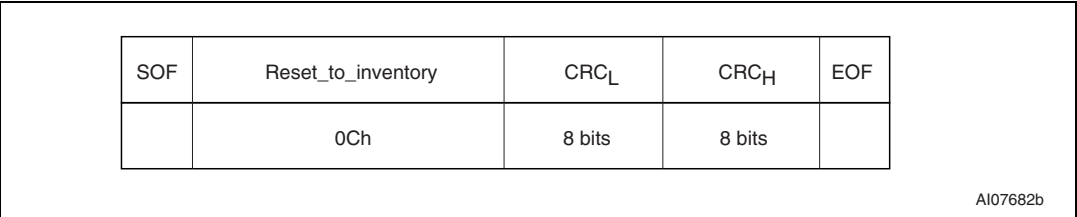
Command code = 0Ch

On receiving the Reset\_to\_inventory() command, all SRIX4K devices in Selected state revert to Inventory state. The concerned SRIX4K devices are thus resubmitted to the anticollision sequence. This command is useful when two SRIX4K devices with the same 8-bit Chip\_ID happen to be in Selected state at the same time. Forcing them to go through the anticollision sequence again allows the reader to generates new Pcall16() commands and so, to set new random Chip\_IDs.

The Reset\_to\_inventory() command does not generate a response.

All SRIX4K devices that are not in Selected state ignore the Reset\_to\_inventory() command.

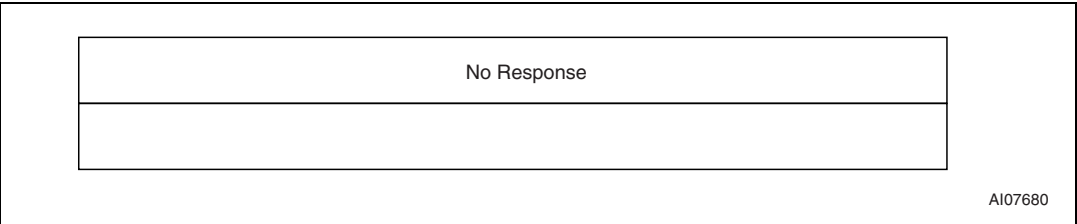
**Figure 39.   Reset\_to\_inventory request format**



Request parameter:

- No parameter

**Figure 40.   Reset\_to\_inventory response format**



**Figure 41.   Reset\_to\_inventory frame exchange between reader and SRIX4K**



## 9.7 Read\_block(Addr) command

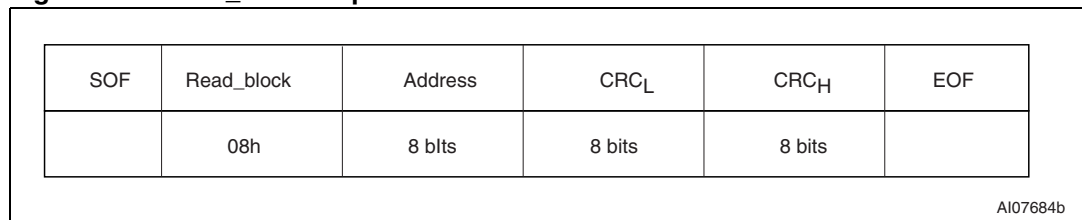
Command code = 08h

On receiving the Read\_block command, the SRIX4K reads the desired block and returns the 4 data bytes contained in the block. Data bytes are transmitted with the Least Significant byte first and each byte is transmitted with the least significant bit first.

The address byte gives access to the 128 blocks of the SRIX4K (addresses 0 to 127). Read\_block commands issued with a block address above 127 will not be interpreted and the SRIX4K will not return any response, except for the System area located at address 255.

The SRIX4K must have received a Select() command and be switched to Selected state before any Read\_block() command can be accepted. All Read\_block() commands sent to the SRIX4K before a Select() command is issued are ignored.

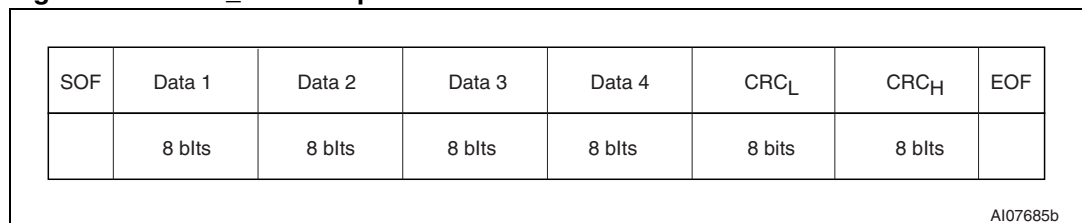
**Figure 42. Read\_block request format**



Request parameter:

- Address: block addresses from 0 to 127, or 255

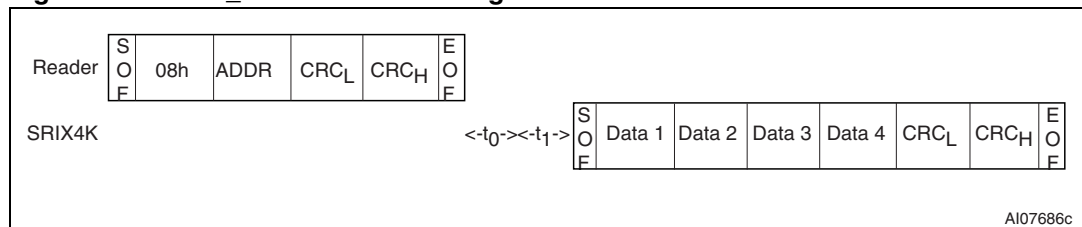
**Figure 43. Read\_block response format**



Response parameters:

- Data 1: Less significant data byte
- Data 2: Data byte
- Data 3: Data byte
- Data 4: Most significant data byte

**Figure 44. Read\_block frame exchange between reader and SRIX4K**



# 9.8 Write\_block (Addr, Data) command

Command code = 09h

On receiving the Write\_block command, the SRIX4K writes the 4 bytes contained in the command to the addressed block, provided that the block is available and not write-protected. Data bytes are transmitted with the least significant byte first, and each byte is transmitted with the least significant bit first.

The address byte gives access to the 128 blocks of the SRIX4K (addresses 0 to 127). Write\_block commands issued with a block address above 127 will not be interpreted and the SRIX4K will not return any response, except for the System area located at address 255.

The result of the Write\_block command is submitted to the addressed block. See the following Figures for a complete description of the Write\_block command:

- [Figure 13: Resettable OTP area \(addresses 0 to 4\).](#)
- [Figure 16: Binary counter \(addresses 5 to 6\).](#)
- [Figure 18: EEPROM \(addresses 7 to 127\).](#)

The Write\_block command does not give rise to a response from the SRIX4K. The reader must check after the programming time,  $t_W$ , that the data was correctly programmed. The SRIX4K must have received a Select() command and be switched to Selected state before any Write\_block command can be accepted. All Write\_block commands sent to the SRIX4K before a Select() command is issued, are ignored.

**Figure 45. Write\_block request format**

| SOF | Write_block | Address | Data 1 | Data 2 | Data 3 | Data 4 | CRC <sub>L</sub> | CRC <sub>H</sub> | EOF |
|-----|-------------|---------|--------|--------|--------|--------|------------------|------------------|-----|
|     | 09h         | 8 blts  | 8 blts | 8 blts | 8 blts | 8 blts | 8 blts           | 8 blts           |     |

AI07687b

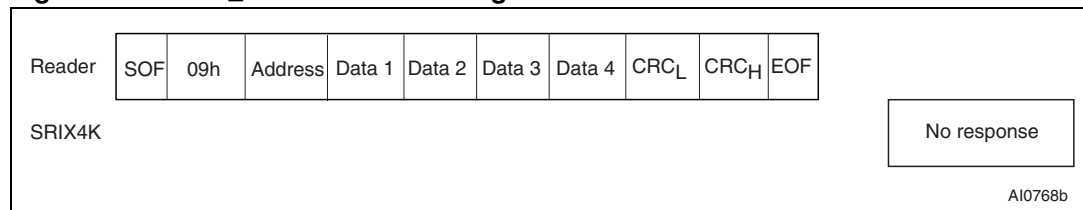
Request parameters:

- Address: block addresses from 0 to 127, or 255
- Data 1: Less significant data byte
- Data 2: Data byte
- Data 3: Data byte
- Data 4: Most significant data byte.

**Figure 46. Write\_block response format**

|             |
|-------------|
| No Response |
|-------------|

AI07680

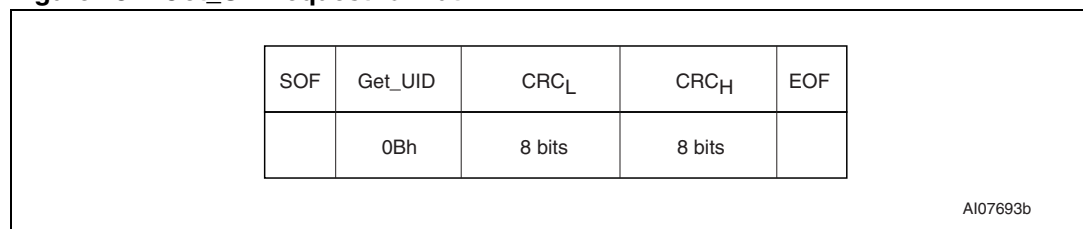
**Figure 47. Write\_block frame exchange between reader and SRIX4K**

## 9.9 Get\_UID() command

Command code = 0Bh

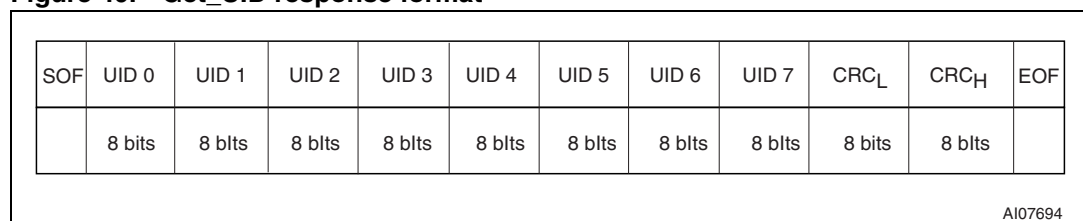
On receiving the Get\_UID command, the SRIX4K returns its 8 UID bytes. UID bytes are transmitted with the least significant byte first, and each byte is transmitted with the least significant bit first.

The SRIX4K must have received a Select() command and be switched to Selected state before any Get\_UID() command can be accepted. All Get\_UID() commands sent to the SRIX4K before a Select() command is issued, are ignored.

**Figure 48. Get\_UID request format**

Request parameter:

- No parameter

**Figure 49. Get\_UID response format**

Response parameters:

- UID 0: Less significant UID byte
- UID 1 to UID 6: UID bytes
- UID 7: Most significant UID byte.

Unique Identifier (UID)

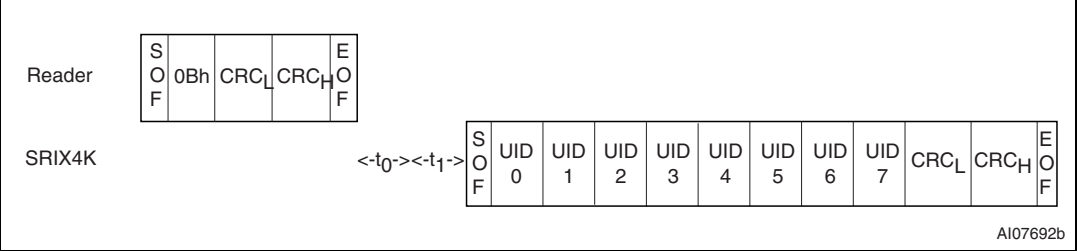
Members of the SRIX4K family are uniquely identified by a 64-bit Unique Identifier (UID). This is used for addressing each SRIX4K device uniquely after the anticollision loop. The UID complies with ISO/IEC 15963 and ISO/IEC 7816-6. It is a read-only code, and comprises (as summarized in *Figure 50*):

- an 8-bit prefix, with the most significant bits set to D0h
- an 8-bit IC manufacturer code (ISO/IEC 7816-6/AM1) set to 02h (for STMicroelectronics)
- a 6-bit IC code set to 00 0011b = 3d for SRIX4K
- a 42-bit unique serial number

Figure 50. 64-bit unique identifier of the SRIX4K



Figure 51. Get\_UID frame exchange between reader and SRIX4K



9.10 Power-on state

After power-on, the SRIX4K is in the following state:

- It is in the low-power state.
- It is in Ready state.
- It shows highest impedance with respect to the reader antenna field.
- It will not respond to any command except Initiate().

## 10 Maximum rating

Stressing the device above the rating listed in the absolute maximum ratings table may cause permanent damage to the device. These are stress ratings only and operation of the device at these or any other conditions above those indicated in the operating sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability. Refer also to the STMicroelectronics SURE Program and other relevant quality documents.

**Table 5. Absolute maximum ratings**

| Symbol                 | Parameter                                      |                                       | Min.  | Max. | Unit   |
|------------------------|------------------------------------------------|---------------------------------------|-------|------|--------|
| $T_{STG}$<br>$t_{STG}$ | Storage conditions                             | Wafer<br>(kept in its antistatic bag) | 15    | 25   | °C     |
|                        |                                                |                                       |       | 23   | months |
| $I_{CC}$               | Supply current on AC0 / AC1                    |                                       | −20   | 20   | mA     |
| $V_{MAX}$              | Input voltage on AC0 / AC1                     |                                       | −7    | 7    | V      |
| $V_{ESD}$              | Electrostatic discharge voltage <sup>(1)</sup> | Machine model                         | −100  | 100  | V      |
|                        |                                                | Human body model                      | −1000 | 1000 | V      |

1. Mil. Std. 883 - Method 3015

# 11 DC and ac parameters

**Table 6. Operating conditions**

| Symbol | Parameter                     | Min. | Max. | Unit |
|--------|-------------------------------|------|------|------|
| $T_A$  | Ambient operating temperature | -20  | 85   | °C   |

**Table 7. DC characteristics**

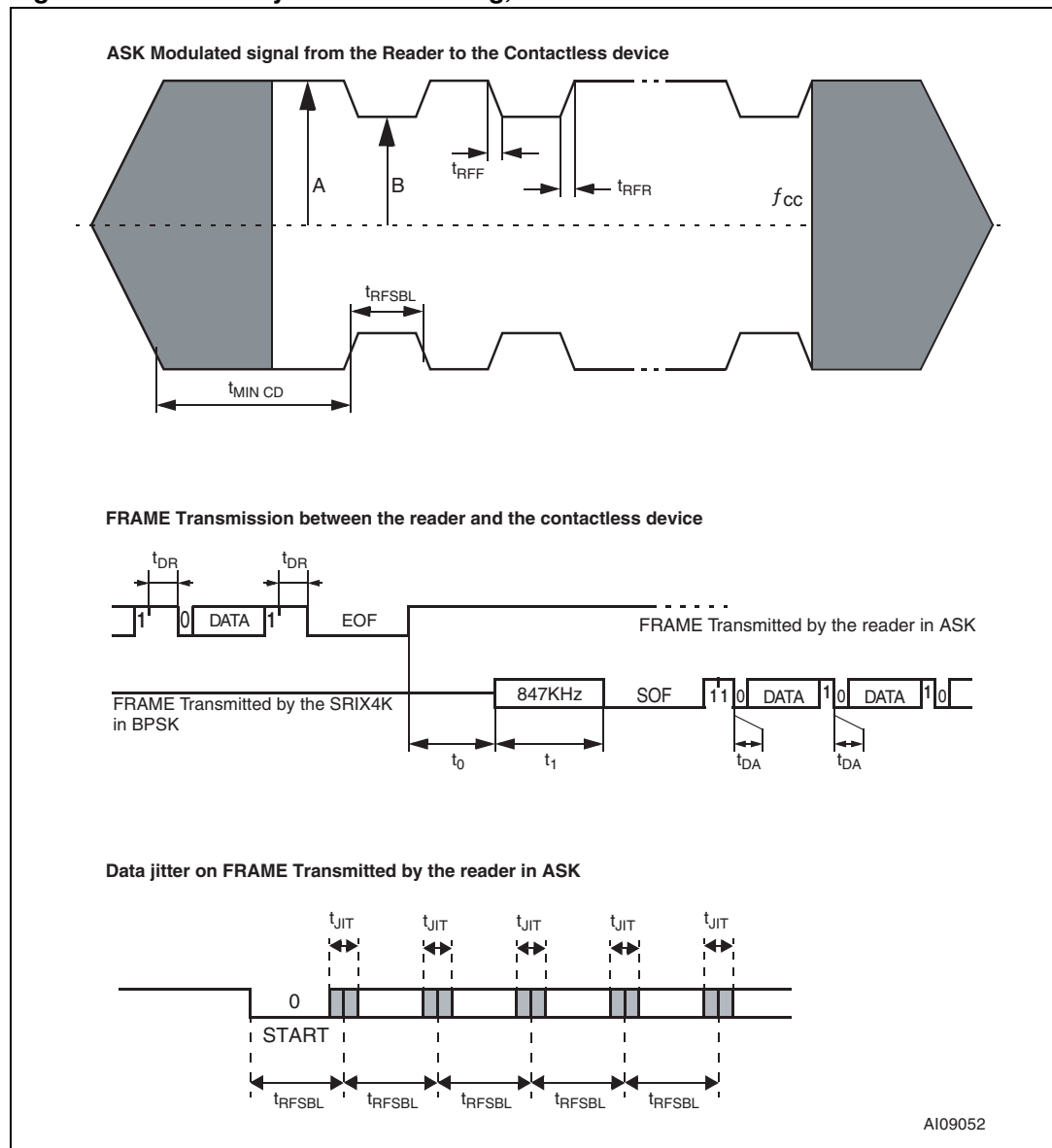
| Symbol    | Parameter                        | Condition               | Min | Typ | Max | Unit          |
|-----------|----------------------------------|-------------------------|-----|-----|-----|---------------|
| $V_{CC}$  | Regulated voltage                |                         | 2.5 |     | 3.5 | V             |
| $I_{CC}$  | Supply current (active in read)  | $V_{CC} = 3.0\text{ V}$ |     |     | 100 | $\mu\text{A}$ |
| $I_{CC}$  | Supply current (active in write) | $V_{CC} = 3.0\text{ V}$ |     |     | 250 | $\mu\text{A}$ |
| $V_{RET}$ | Retromodulation induced voltage  | ISO 10373-6             | 20  |     |     | mV            |
| $C_{TUN}$ | Internal tuning capacitor        | 13.56 MHz               |     | 64  |     | pF            |

**Table 8. AC characteristics**

| Symbol             | Parameter                                          | Condition                      | Min    | Max    | Unit          |
|--------------------|----------------------------------------------------|--------------------------------|--------|--------|---------------|
| $f_{CC}$           | External RF signal frequency                       |                                | 13.553 | 13.567 | MHz           |
| $MI_{CARRIER}$     | Carrier modulation index                           | $MI=(A-B)/(A+B)$               | 8      | 14     | %             |
| $t_{RFR}, t_{RFF}$ | 10% rise and fall times                            |                                | 0.8    | 2.5    | $\mu\text{s}$ |
| $t_{RFSBL}$        | Minimum pulse width for start bit                  | $ETU = 128/f_{CC}$             | 9.44   |        | $\mu\text{s}$ |
| $t_{JIT}$          | ASK modulation data jitter                         | Coupler to SRIX4K              | -2     | +2     | $\mu\text{s}$ |
| $t_{MIN\ CD}$      | Minimum time from carrier generation to first data |                                | 5      |        | ms            |
| $f_S$              | Subcarrier frequency                               | $f_{CC}/16$                    | 847.5  |        | kHz           |
| $t_0$              | Antenna reversal delay                             | $128/f_S$                      | 151    |        | $\mu\text{s}$ |
| $t_1$              | Synchronization delay                              | $128/f_S$                      | 151    |        | $\mu\text{s}$ |
| $t_2$              | Answer to new request delay                        | 14 ETU                         | 132    |        | $\mu\text{s}$ |
| $t_{DR}$           | Time between request characters                    | Coupler to SRIX4K              | 0      | 57     | $\mu\text{s}$ |
| $t_{DA}$           | Time between answer characters                     | SRIX4K to coupler              | 0      |        | $\mu\text{s}$ |
| $t_W$              | Programming time for write                         | With no auto-erase cycle (OTP) |        | 3      | ms            |
|                    |                                                    | With auto-erase cycle (EEPROM) |        | 5      | ms            |
|                    |                                                    | Binary counter decrement       |        | 7      | ms            |

1. All timing measurements were performed on a reference antenna with the following characteristics:  
 External size: 75 mm x 48 mm  
 Number of turns: 3  
 Width of conductor: 1 mm  
 Space between 2 conductors: 0.4 mm  
 Value of the coil: 1.4  $\mu\text{H}$   
 Tuning Frequency: 14.4 MHz.

Figure 52. SRIX4K synchronous timing, transmit and receive



# 12      Part numbering

Table 9.      Ordering information scheme

|                      |                                                                                                  |   |    |      |
|----------------------|--------------------------------------------------------------------------------------------------|---|----|------|
| Example:             | SRIX4K                                                                                           | – | W4 | /1GE |
| <b>Device type</b>   | SRIX4K                                                                                           |   |    |      |
| <b>Package</b>       | W4 = 180 µm ± 15 µm unsawn wafer<br>SBN18 = 180 µm ± 15 µm bumped and sawn wafer on 8-inch frame |   |    |      |
| <b>Customer code</b> | 1GE = generic product<br>xxx = customer code after personalization                               |   |    |      |

*Note:*      *Devices are shipped from the factory with the memory content bits erased to 1.*

For a list of available options (speed, package, etc.) or for further information on any aspect of this device, please contact your nearest ST sales office.

## Appendix A ISO 14443 Type B CRC calculation

```

#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#define BYTE unsigned char
#define USHORT unsigned short

unsigned short UpdateCrc(BYTE ch, USHORT *lpwCrc)
{
    ch = (ch^(BYTE)((*lpwCrc) & 0x00FF));
    ch = (ch^(ch<<4));
    *lpwCrc = (*lpwCrc >> 8) ^ ((USHORT)ch <<
8) ^ ((USHORT)ch<<3) ^ ((USHORT)ch>>4);
    return(*lpwCrc);
}

void ComputeCrc(char *Data, int Length, BYTE *TransmitFirst, BYTE
*TransmitSecond)
{
    BYTE chBlock; USHORTt wCrc;
    wCrc = 0xFFFF; // ISO 3309
    do
    {
        {
            chBlock = *Data++;
            UpdateCrc(chBlock, &wCrc);
        } while (--Length);
    }
    wCrc = ~wCrc; // ISO 3309
    *TransmitFirst = (BYTE) (wCrc & 0xFF);
    *TransmitSecond = (BYTE) ((wCrc >> 8) & 0xFF);
    return;
}

int main(void)
{
    BYTE BuffCRC_B[10] = {0x0A, 0x12, 0x34, 0x56}, First, Second, i;
    printf("Crc-16 G(x) = x^16 + x^12 + x^5 + 1");
    printf("CRC_B of [ ");
    for(i=0; i<4; i++)
        printf("%02X ",BuffCRC_B[i]);
    ComputeCrc(BuffCRC_B, 4, &First, &Second);
    printf("] Transmitted: %02X then %02X.", First, Second);
    return(0);
}

```

Appendix B      SRIX4K command summary

Figure 53.    Initiate frame exchange between reader and SRIX4K

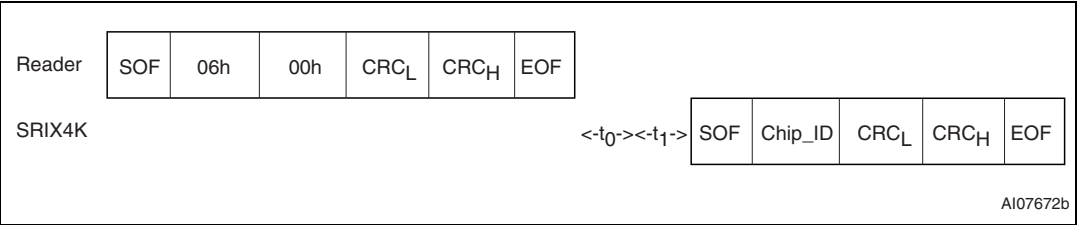


Figure 54.    Pcall16 frame exchange between reader and SRIX4K

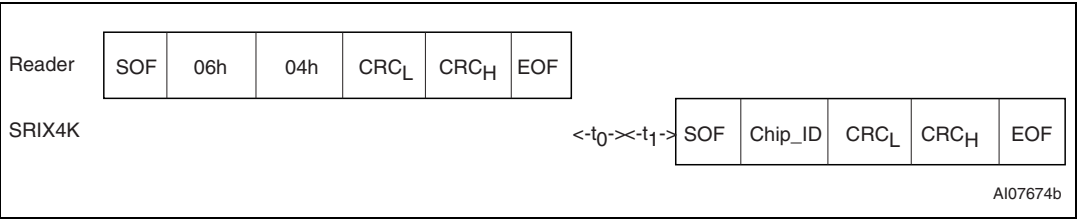


Figure 55.    Slot\_marker frame exchange between reader and SRIX4K

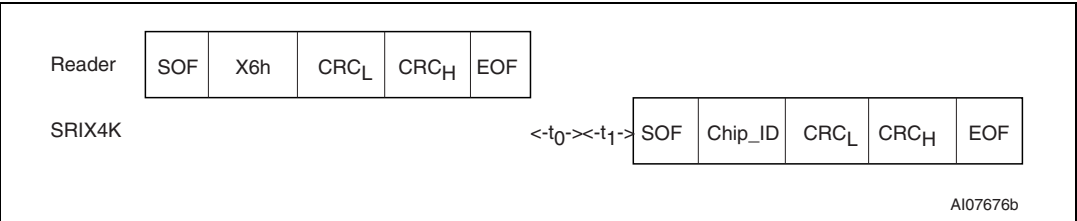


Figure 56.    Select frame exchange between reader and SRIX4K

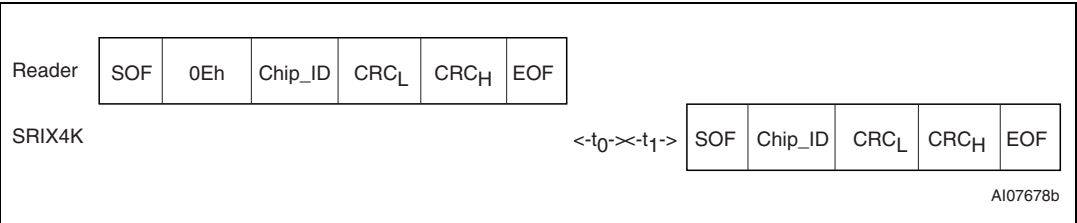


Figure 57.    Completion frame exchange between reader and SRIX4K

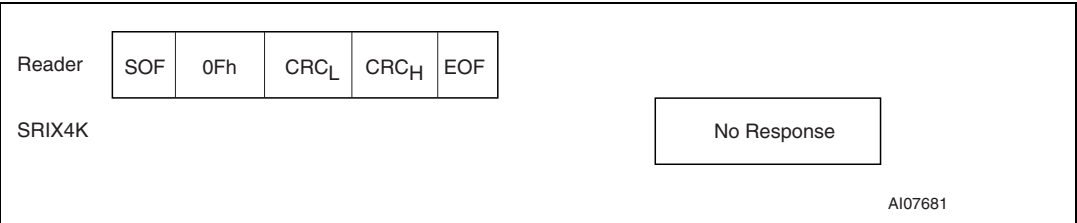


Figure 58. Reset\_to\_inventory frame exchange between reader and SRIX4K

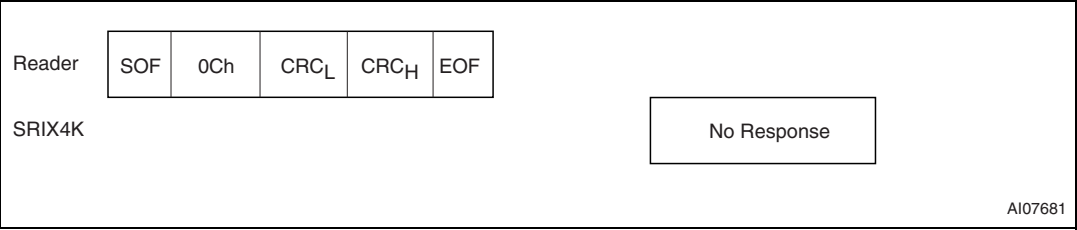


Figure 59. Read\_block frame exchange between reader and SRIX4K

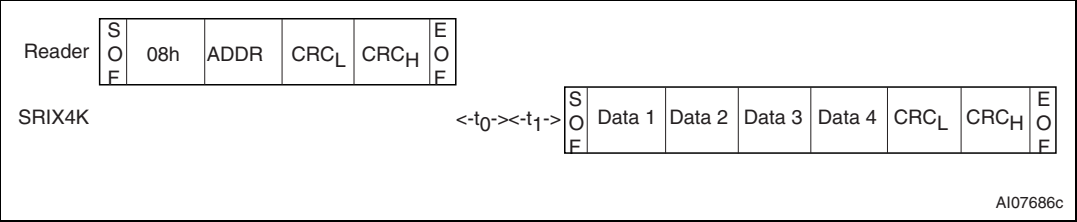


Figure 60. Write\_block frame exchange between reader and SRIX4K

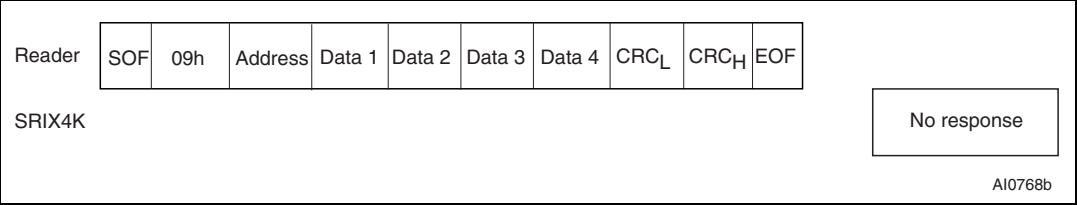
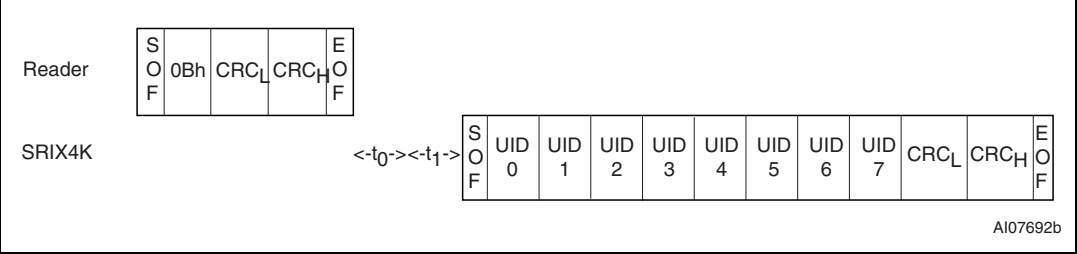


Figure 61. Get\_UID frame exchange between reader and SRIX4K



## Revision history

**Table 10. Document revision history**

| Date        | Version | Changes                                                                                                                                                                                                                                                                                                                                                                                              |
|-------------|---------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| 28-Nov-2002 | 1.0     | Document written                                                                                                                                                                                                                                                                                                                                                                                     |
| 17-Jul-2003 | 1.1     | Data briefing extracted                                                                                                                                                                                                                                                                                                                                                                              |
| 12-Mar-2004 | 2.0     | First public release of full datasheet                                                                                                                                                                                                                                                                                                                                                               |
| 26-Apr-2004 | 3.0     | Correction to memory map                                                                                                                                                                                                                                                                                                                                                                             |
| 29-Nov-2004 | 4.0     | <i>Package mechanical</i> section revised.                                                                                                                                                                                                                                                                                                                                                           |
| 13-Dec-2004 | 5.0     | V <sub>RET</sub> and C <sub>TUN</sub> parameters added to <i>Table 7: DC characteristics</i> .                                                                                                                                                                                                                                                                                                       |
| 17-Aug-2005 | 6.0     | Updated initial counter values in <i>Section 4.2: 32-bit binary counters on page 16</i> .                                                                                                                                                                                                                                                                                                            |
| 10-Apr-2007 | 7       | Document reformatted. Small text changes.<br>All antennas are ECOPACK® compliant.<br><i>Unique Identifier (UID) on page 38</i> added. C <sub>TUN</sub> min and max values removed, typical value added in <i>Table 7: DC characteristics</i> .<br>Space removed between t0 and t1 in “frame exchange between reader and SRIX4K” Figures (see <i>Appendix B: SRIX4K command summary on page 44</i> ). |
| 28-Aug-2008 | 8       | SRIX4K products no longer delivered in A3, A4 and A5 antennas.<br><i>Table 5: Absolute maximum ratings</i> and <i>Table 9: Ordering information scheme</i> clarified. Small text changes.                                                                                                                                                                                                            |

**Please Read Carefully:**

Information in this document is provided solely in connection with ST products. STMicroelectronics NV and its subsidiaries ("ST") reserve the right to make changes, corrections, modifications or improvements, to this document, and the products and services described herein at any time, without notice.

All ST products are sold pursuant to ST's terms and conditions of sale.

Purchasers are solely responsible for the choice, selection and use of the ST products and services described herein, and ST assumes no liability whatsoever relating to the choice, selection or use of the ST products and services described herein.

No license, express or implied, by estoppel or otherwise, to any intellectual property rights is granted under this document. If any part of this document refers to any third party products or services it shall not be deemed a license grant by ST for the use of such third party products or services, or any intellectual property contained therein or considered as a warranty covering the use in any manner whatsoever of such third party products or services or any intellectual property contained therein.

**UNLESS OTHERWISE SET FORTH IN ST'S TERMS AND CONDITIONS OF SALE ST DISCLAIMS ANY EXPRESS OR IMPLIED WARRANTY WITH RESPECT TO THE USE AND/OR SALE OF ST PRODUCTS INCLUDING WITHOUT LIMITATION IMPLIED WARRANTIES OF MERCHANTABILITY, FITNESS FOR A PARTICULAR PURPOSE (AND THEIR EQUIVALENTS UNDER THE LAWS OF ANY JURISDICTION), OR INFRINGEMENT OF ANY PATENT, COPYRIGHT OR OTHER INTELLECTUAL PROPERTY RIGHT.**

**UNLESS EXPRESSLY APPROVED IN WRITING BY AN AUTHORIZED ST REPRESENTATIVE, ST PRODUCTS ARE NOT RECOMMENDED, AUTHORIZED OR WARRANTED FOR USE IN MILITARY, AIR CRAFT, SPACE, LIFE SAVING, OR LIFE SUSTAINING APPLICATIONS, NOR IN PRODUCTS OR SYSTEMS WHERE FAILURE OR MALFUNCTION MAY RESULT IN PERSONAL INJURY, DEATH, OR SEVERE PROPERTY OR ENVIRONMENTAL DAMAGE. ST PRODUCTS WHICH ARE NOT SPECIFIED AS "AUTOMOTIVE GRADE" MAY ONLY BE USED IN AUTOMOTIVE APPLICATIONS AT USER'S OWN RISK.**

Resale of ST products with provisions different from the statements and/or technical features set forth in this document shall immediately void any warranty granted by ST for the ST product or service described herein and shall not create or extend in any manner whatsoever, any liability of ST.

ST and the ST logo are trademarks or registered trademarks of ST in various countries.

Information in this document supersedes and replaces all information previously supplied.

The ST logo is a registered trademark of STMicroelectronics. All other names are the property of their respective owners.

© 2008 STMicroelectronics - All rights reserved

STMicroelectronics group of companies

Australia - Belgium - Brazil - Canada - China - Czech Republic - Finland - France - Germany - Hong Kong - India - Israel - Italy - Japan - Malaysia - Malta - Morocco - Singapore - Spain - Sweden - Switzerland - United Kingdom - United States of America

[www.st.com](http://www.st.com)

## Данный компонент на территории Российской Федерации

**Вы можете приобрести в компании MosChip.**

Для оперативного оформления запроса Вам необходимо перейти по данной ссылке:

<http://moschip.ru/get-element>

Вы можете разместить у нас заказ для любого Вашего проекта, будь то серийное производство или разработка единичного прибора.

В нашем ассортименте представлены ведущие мировые производители активных и пассивных электронных компонентов.

Нашей специализацией является поставка электронной компонентной базы двойного назначения, продукции таких производителей как XILINX, Intel (ex.ALTERA), Vicor, Microchip, Texas Instruments, Analog Devices, Mini-Circuits, Amphenol, Glenair.

Сотрудничество с глобальными дистрибьюторами электронных компонентов, предоставляет возможность заказывать и получать с международных складов практически любой перечень компонентов в оптимальные для Вас сроки.

На всех этапах разработки и производства наши партнеры могут получить квалифицированную поддержку опытных инженеров.

Система менеджмента качества компании отвечает требованиям в соответствии с ГОСТ Р ИСО 9001, ГОСТ РВ 0015-002 и ЭС РД 009

### Офис по работе с юридическими лицами:

105318, г.Москва, ул.Щербаковская д.3, офис 1107, 1118, ДЦ «Щербаковский»

Телефон: +7 495 668-12-70 (многоканальный)

Факс: +7 495 668-12-70 (доб.304)

E-mail: [info@moschip.ru](mailto:info@moschip.ru)

Skype отдела продаж:

moschip.ru

moschip.ru\_4

moschip.ru\_6

moschip.ru\_9