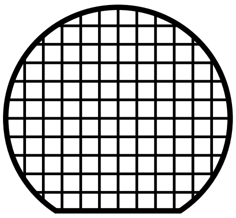
ST25TB04K

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

Datasheet - production data



- Unsawn wafer
- Bumped and sawn wafer

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
- 8 bit Chip_ID based anticollision system
- 2 count-down binary counters with automated anti-tearing protection
- 64-bit Unique Identifier
- 4096-bit EEPROM with write protect feature
- Read_block and Write_block (32 bits)
- Internal tuning capacitor: 68 pF
- 1 million erase/write cycles
- 40-year data retention
- Self-timed programming cycle
- 5 ms typical programming time

Contents

List of tables

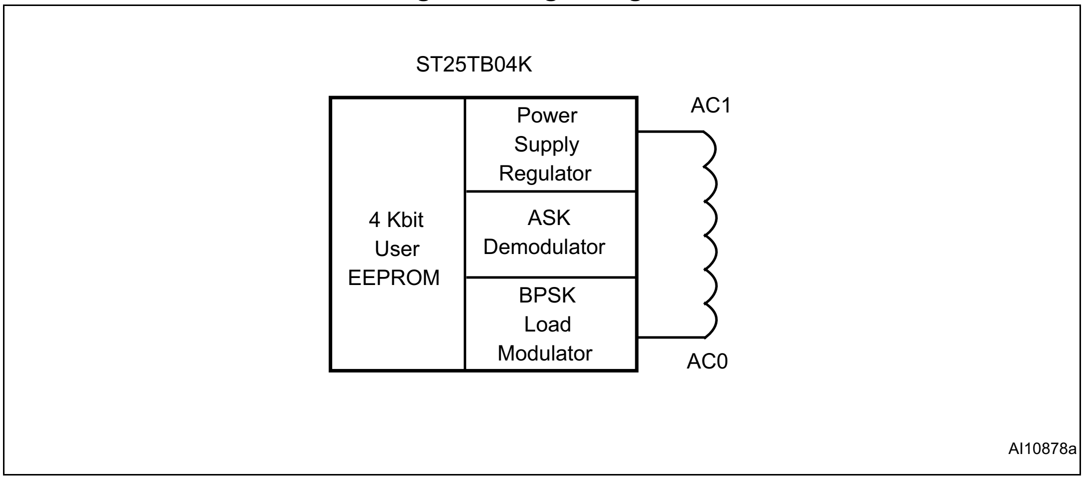
List of figures

1 Description

The ST25TB04K is a contactless memory, powered by an externally transmitted radio wave. It contains a 4096-bit user EEPROM. The memory is organized as 128 blocks of 32 bits. The ST25TB04K 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 sub-carrier. The received ASK wave is 10% modulated. The data transfer rate between the ST25TB04K and the reader is 106 kbit/s in both reception and emission modes.

The ST25TB04K follows the ISO 14443 - 2 Type B recommendation for the radio-frequency power and signal interface.

Figure 1. Logic diagram



The ST25TB04K is specifically designed for short range applications that need re-usable products. The ST25TB04K 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.

Table 1. Signal names

Signal names	Description
AC1	Antenna coil
AC0	Antenna coil

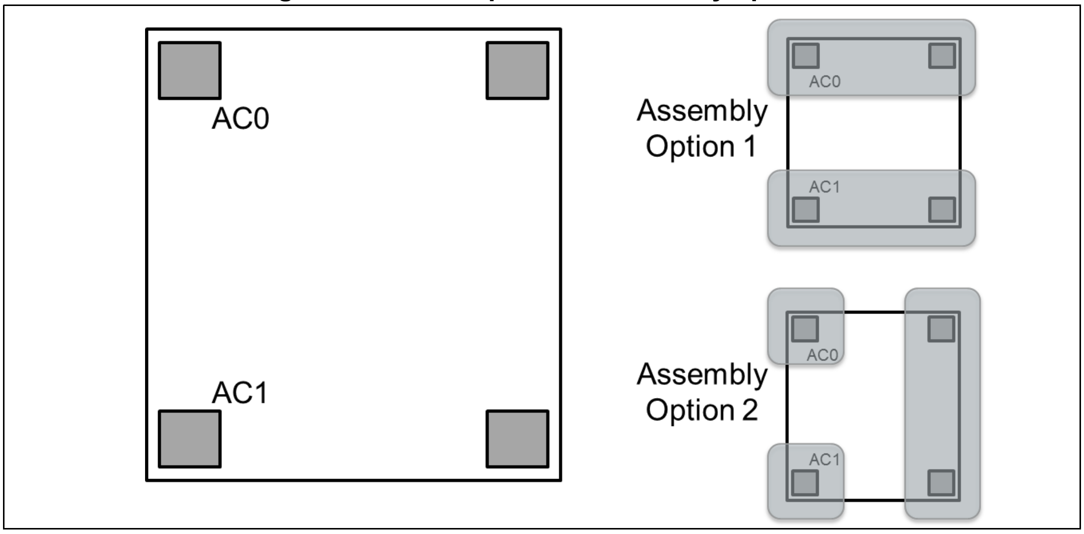
The ST25TB04K contactless EEPROM can be randomly read and written in block mode (each block containing 32 bits). The instruction set includes the following nine commands:

- Read_block
- Write_block
- Initiate
- Pcall16
- Slot_marker
- Select
- Completion
- Reset_to_inventory
- Get_UID

The ST25TB04K memory is organized in three areas, as described in *Table 3*. 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. 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.

Die floor plan and physical options related to the die assembly are described in *Figure 2*.

Figure 2. Die floor plan and assembly options



For the option 1 of the die assembly, the CTUN (referenced in *Table 13*) can increase from 0.5pF to 1pF. The option 2 of the die assembly is showing a tripod which can be used for physical stability, having no impact on CTUN parameter.

2 Signal description

2.1 AC1, AC0

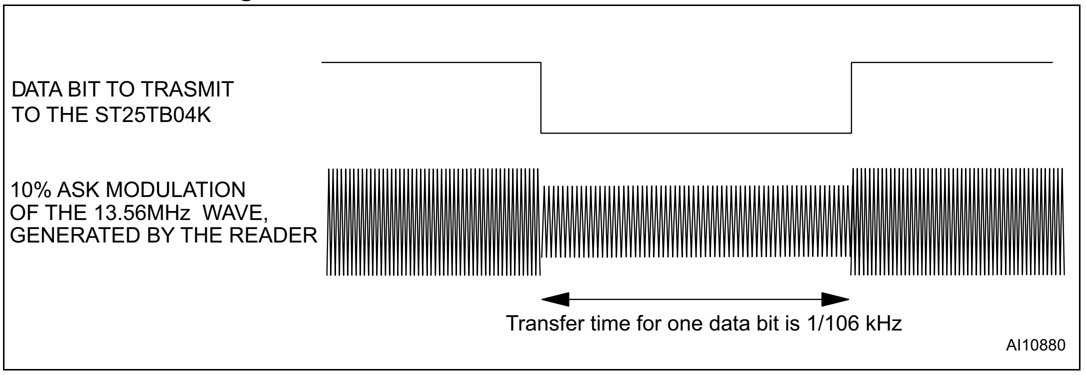
The pads for the Antenna Coil. AC1 and AC0 must be directly bonded to the antenna.

3 Data transfer

3.1 Input data transfer from reader to ST25TB04K (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 ST25TB04K’s antenna is transformed into a supply voltage by a regulator, and into data bits by the ASK demodulator. For the ST25TB04K to decode correctly the information it receives, the reader must 10% amplitude-modulate the 13.56 MHz wave before sending it to the ST25TB04K. 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 ST25TB04K 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 µs (1/106 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 ST25TB04K does not execute the command, but it does not generate an error frame.

Figure 4. ST25TB04K request frame character format

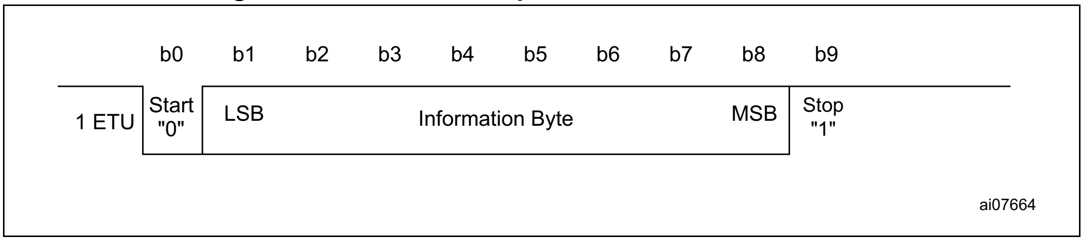


Table 2. Bit description

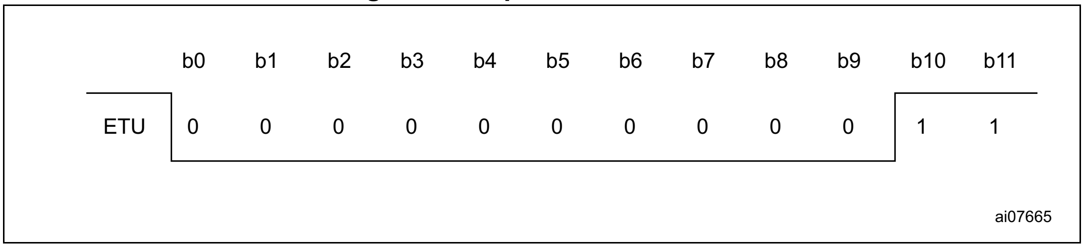
Bit	Description	Value
b_0	Start bit used to synchronize the transmission	$b_0 = 0$
b_1 to b_8	Information byte (command, address or data)	The information byte is sent with the least significant bit first
b_9	Stop bit used to indicate the end of a character	$b_9 = 1$

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

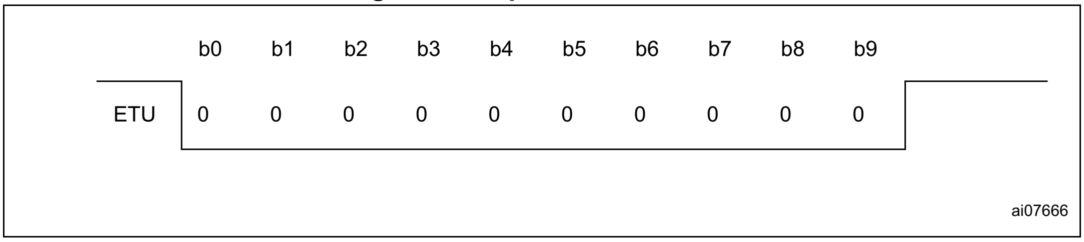


3.1.3 Request end of frame

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.

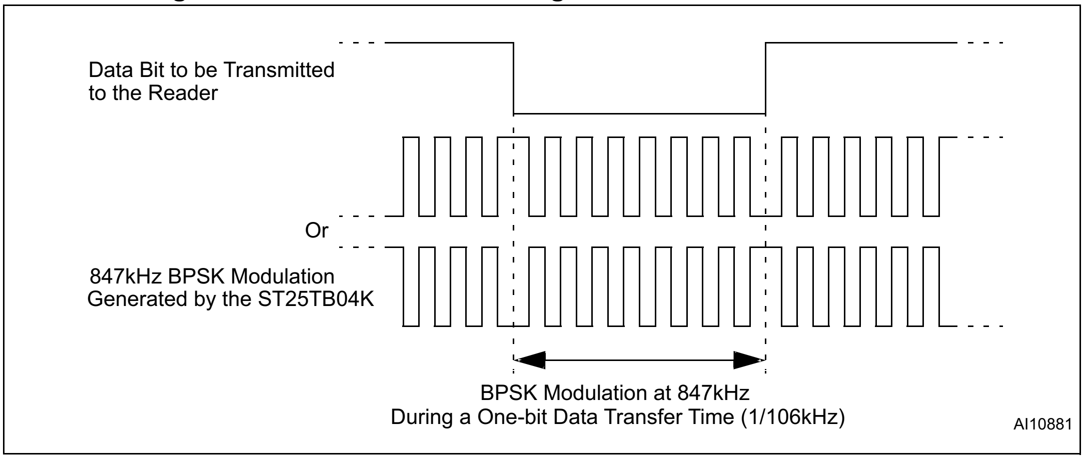
Figure 6. Request end of frame



3.2 Output data transfer from ST25TB04K to reader (answer frame)

The data bits issued by the ST25TB04K use back-scattering. Back-scattering is obtained by modifying the ST25TB04K 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 ST25TB04K. 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

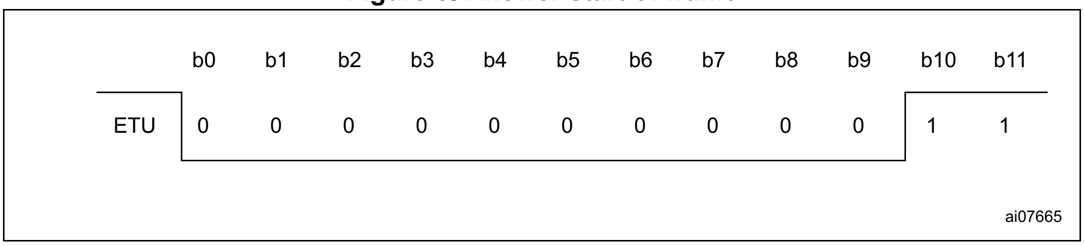
The character format is the same as for input data transfer (*Figure 4*). The transmitted frames are made up of an SOF, data, a CRC and an EOF (*Figure 10*). As with an input data transfer, if an error occurs, the reader does not issue an error code to the ST25TB04K, but it should be able to detect it 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

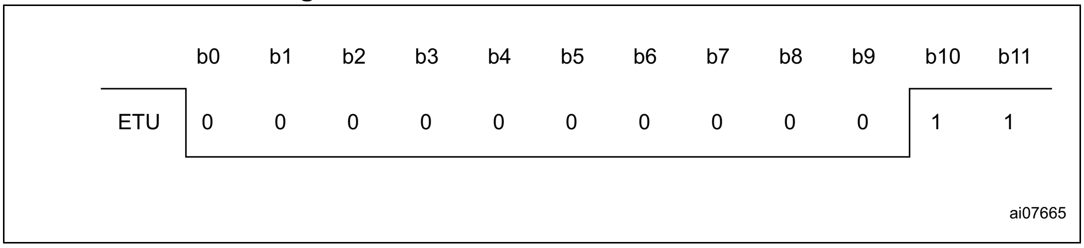


3.2.3 Answer end of frame

The EOF shown in *Figure 9* is composed of:

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

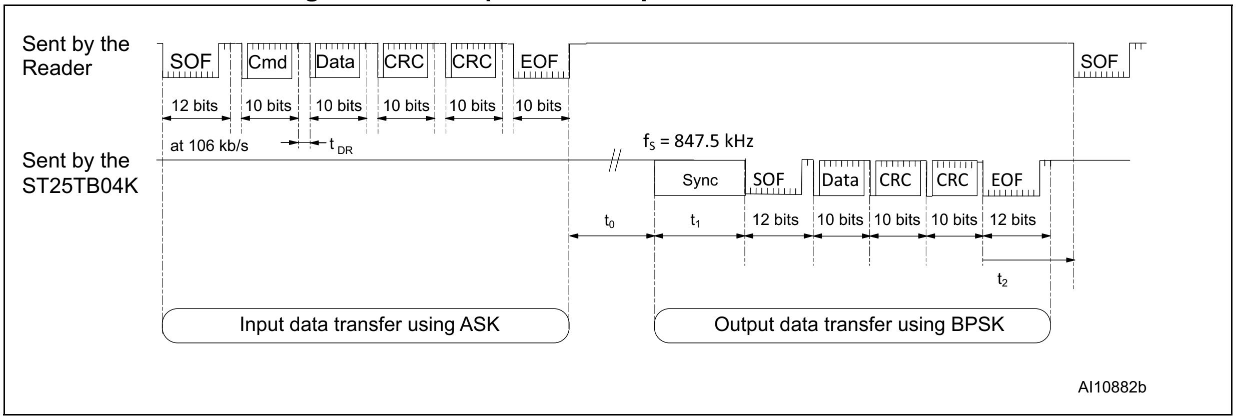
Figure 9. Answer end of frame



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 ST25TB04K 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 ST25TB04K 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 ST25TB04K.

Figure 10. Example of a complete transmission frame



3.4 CRC

The 16-bit CRC used by the ST25TB04K is generated in compliance with the ISO14443 Type B recommendation. For further information, please see *Appendix A*. The initial register contents are all 1s: FFFFh.

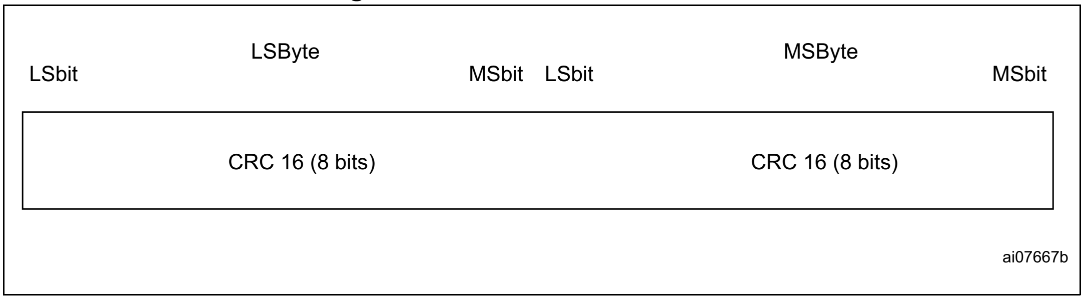
The two-byte CRC is present in every request and in every answer frame, before the EOF. The CRC is calculated on all the bytes between SOF (not included) and the CRC field.

Upon reception of a request from a reader, the ST25TB04K verifies that the CRC value is valid. If it is invalid, the ST25TB04K discards the frame and does not answer the reader.

Upon reception of an answer from the ST25TB04K, 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 ST25TB04K is organized as 128 blocks of 32 bits as shown in *Table 3*. 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.

Table 3. ST25TB04K memory mapping

<table>
<tr><th rowspan="2">Block Address</th><th>MSB</th><th colspan="4">32-bit block</th><th>LSB</th><th rowspan="2">Description</th></tr>
<tr><th>b31</th><th>b24 b23</th><th>b16</th><th>b15</th><th>b8 b7</th><th>b0</th></tr>
<tr><td>0</td><td colspan="6">32-bit Boolean area</td><td rowspan="5">Resettable OTP bit</td></tr>
<tr><td>1</td><td colspan="6">32-bit Boolean area</td></tr>
<tr><td>2</td><td colspan="6">32-bit Boolean area</td></tr>
<tr><td>3</td><td colspan="6">32-bit Boolean area</td></tr>
<tr><td>4</td><td colspan="6">32-bit Boolean area</td></tr>
<tr><td>5</td><td colspan="6">32 bits binary counter</td><td rowspan="2">Count down counter</td></tr>
<tr><td>6</td><td colspan="6">32 bits binary counter</td></tr>
<tr><td>7</td><td colspan="6">User area</td><td rowspan="9">Lockable EEPROM</td></tr>
<tr><td>8</td><td colspan="6">User area</td></tr>
<tr><td>9</td><td colspan="6">User area</td></tr>
<tr><td>10</td><td colspan="6">User area</td></tr>
<tr><td>11</td><td colspan="6">User area</td></tr>
<tr><td>12</td><td colspan="6">User area</td></tr>
<tr><td>13</td><td colspan="6">User area</td></tr>
<tr><td>14</td><td colspan="6">User area</td></tr>
<tr><td>15</td><td colspan="6">User area</td></tr>
<tr><td>16</td><td colspan="6">User area</td><td rowspan="3">EEPROM</td></tr>
<tr><td>...</td><td colspan="6">User area</td></tr>
<tr><td>127</td><td colspan="6">User area</td></tr>
<tr><td>255</td><td>OTP_Lock_Reg</td><td colspan="5">ST Reserved</td><td>System OTP bits</td></tr>
<tr><td>UID0</td><td colspan="6" rowspan="2">64 bits UID area</td><td rowspan="2">ROM</td></tr>
<tr><td>UID1</td></tr>
</table>

4.1 Resettable OTP area

In this area contains five individual 32-bit Boolean words (see *Table 4* 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 12* and *Figure 13* for examples of the result of the Write_block command in the resettable OTP area.

Table 4. Resettable OTP area (addresses 0 to 4)

Block Address	MSB		32-bit block		LSB	Description
	b31	b24 b23	b16 b15	b8 b7	b0	
0	32-bit Boolean area					Resettable OTP bit
1	32-bit Boolean area					
2	32-bit Boolean area					
3	32-bit Boolean area					
4	32-bit Boolean area					

Figure 12. 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 one reload mode is activated. The reload mode 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 13. Write_block update in Reload 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	1	1	1	0	1	1	0	0	1	1	1	1
New data stored in block	1	...	1	1	1	1	0	1	1	0	0	1	1	1	1

ai07659

4.2 32-bit binary counters

The two 32-bit binary counters are located at block addresses 5 and 6. The ST25TB04K 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 reached value is 0000 0000h, the counter is empty and cannot be reloaded. For each counter 5 and 6, the update is done by issuing the Write_block command. The Write_block command writes the new 32-bit value to the counter block address. *Table 5* 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.

Table 5. Binary counter (addresses 5 to 6)

Block Address	MSB		32-bit block		LSB	Description
	b31	b24 b23	b16 b15	b8 b7	b0	
5	32-bit Boolean area					Count down counter
6	32-bit Boolean area					

Figure 14. Countdown example (binary format)

	b31														b0
Initial data	1	...	1	1	1	1	1	1	1	1	1	1	1	1	1
1-unit decrement	1	...	1	1	1	1	1	1	1	1	1	1	1	1	0
1-unit decrement	1	...	1	1	1	1	1	1	1	1	1	1	1	0	1
1-unit decrement	1	...	1	1	1	1	1	1	1	1	1	1	1	0	0
8-unit decrement	1	...	1	1	1	1	1	1	1	1	1	0	1	0	0
Increment not allowed	1	...	1	1	1	1	1	1	1	1	1	1	0	0	0

ai07661

The counter with block address 6 controls the reload mode 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 ST25TB04K detects the change and adds an Erase cycle to the Write_block command for locations 0 to 4 (see the "*Resettable OTP area*" paragraph).

The Erase cycle remains active until a Power-off or a Select command is issued.

The ST25TB04K'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 *Table 6* 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.

Table 6. EEPROM (addresses 7 to 127)

Block Address	MSB		32-bit block		LSB	Description
	b31	b24 b23	b16 b15	b8 b7	b0	
7	user area					Lockable 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					

4.4 System area

This area is used to modify the settings of the ST25TB04K. It contains 2 registers: OTP_Lock_Reg and ST Reserved. See *Table 7* 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.

Table 7. System area

Block Address	MSB			32-bit block			LSB	Description
	b31	b24	b23	b16 b15	b8	b7	b0	
255	OTP_Lock_Reg		ST reserved					OTP

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 ST25TB04K. 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.

After any modification of the OTP_Lock_Reg bits, it is necessary to send a Select command with a valid Chip_ID to the ST25TB04K in order to load the block write protection into the logic.

5 ST25TB04K operation

All commands, data and CRC are transmitted to the ST25TB04K 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 ST25TB04K at the antenna is demodulated by the 10% ASK demodulator, and decoded by the internal logic. Prior to any operation, the ST25TB04K must have been selected by a Select command. Each frame transmitted to the ST25TB04K 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 ST25TB04K (wrong command or CRC error), the memory does not return any error code.

When a valid frame is received, the ST25TB04K 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 ST25TB04K sending the 2 CRC bytes and the EOF.

6 ST25TB04K states

The ST25TB04K can be switched into different states. Depending on the current state of the ST25TB04K, its logic will only answer to specific commands. These states are mainly used during the anticollision sequence, to identify and to access the ST25TB04K in a very short time. The ST25TB04K provides 6 different states, as described in the following paragraphs and in *Figure 15*.

6.1 Power-off state

The ST25TB04K is in Power-off state when the electromagnetic field around the tag is not strong enough. In this state, the ST25TB04K does not respond to any command.

6.2 Ready state

When the electromagnetic field is strong enough, the ST25TB04K 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 ST25TB04K.

6.3 Inventory state

The ST25TB04K switches from the Ready to the Inventory state after an Initiate() command has been issued. In Inventory state, the ST25TB04K 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 ST25TB04K is active and responds to all Read_block(), Write_block() and Get_UID() commands. When an ST25TB04K has entered the Selected state, it no longer responds to anticollision commands. So that the reader can access another tag, the ST25TB04K can be switched to the Deselected state by sending a Select(Chip_ID) 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 ST25TB04K can be in Selected state at a time.

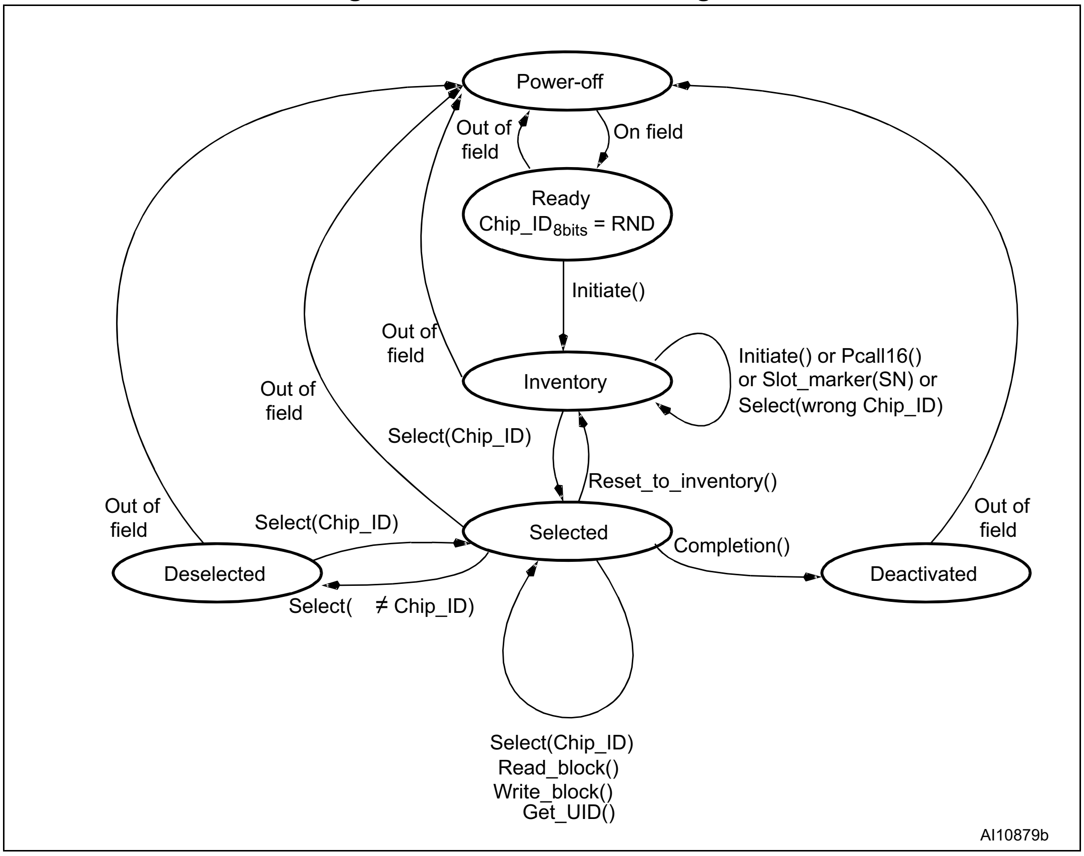
6.5 Deselected state

Once the ST25TB04K 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 ST25TB04K can only be turned off. All commands are ignored.

Figure 15. State transition diagram



7 Anticollision

The ST25TB04K 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 ST25TB04K individually, and access its memory. The anticollision sequence is managed by the reader through a set of commands described in *Section 8: ST25TB04K commands*:

- Initiate()
- Pcall16()
- Slot_marker().

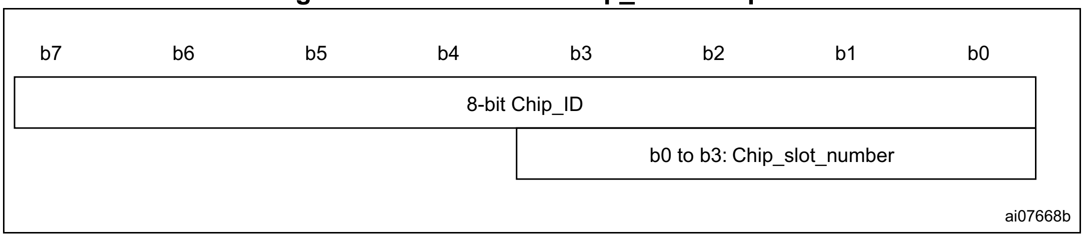
The reader is the master of the communication with one or more ST25TB04K device(s). It initiates the tag communication activity by issuing an Initiate(), Pcall16() or Slot_marker() command to prompt the ST25TB04K to answer. During the anticollision sequence, it might happen that two or more ST25TB04K devices respond simultaneously, so causing a collision. The command set allows the reader to handle the sequence, to separate ST25TB04K transmissions into different time slots. Once the anticollision sequence has completed, ST25TB04K communication is fully under the control of the reader, allowing only one ST25TB04K to transmit at a time.

The Anticollision scheme is based on the definition of time slots during which the ST25TB04K 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 ST25TB04K answers after the command is issued. ST25TB04K devices are allowed to answer only once during the anticollision sequence. Consequently, even if there are several ST25TB04K devices present in the reader field, there will probably be a slot in which only one ST25TB04K answers, allowing the reader to capture its Chip_ID. Using the Chip_ID, the reader can then establish a communication channel with the identified ST25TB04K. The purpose of the anticollision sequence is to allow the reader to select one ST25TB04K at a time.

The ST25TB04K 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.

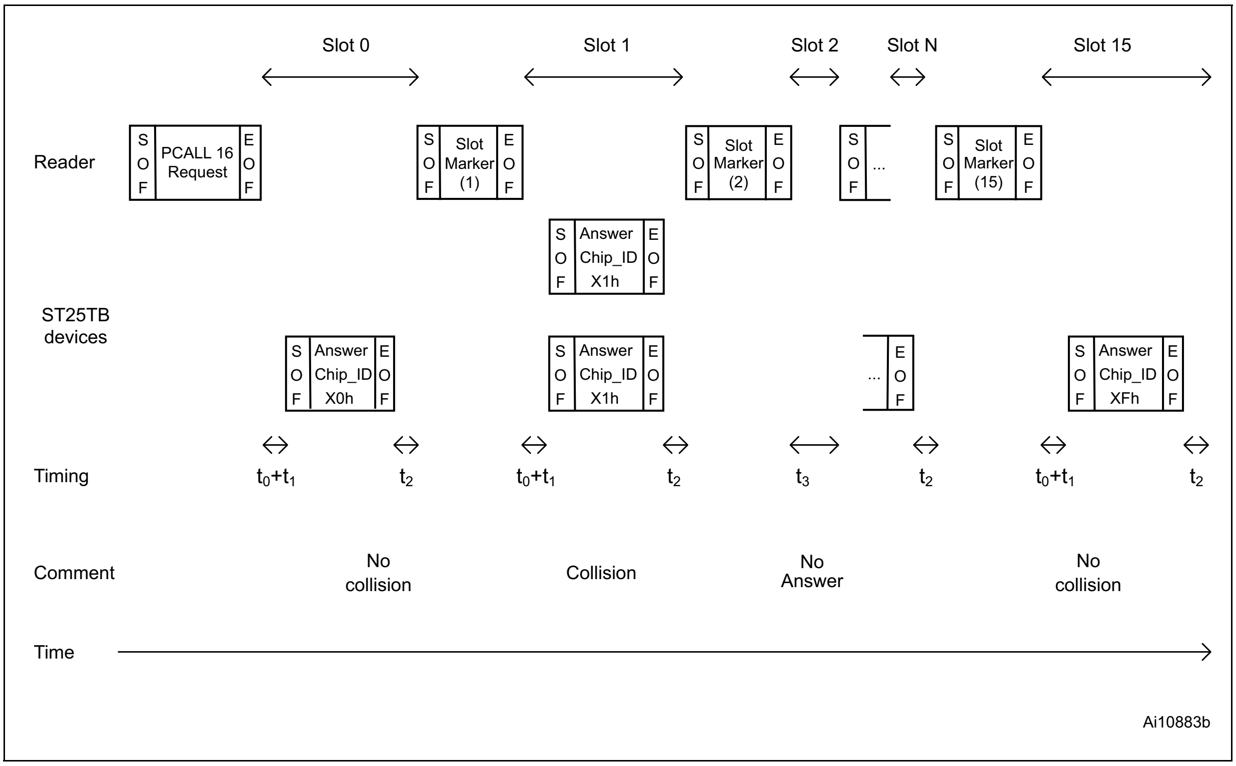
The four least significant bits (b_0 to b_3) of the Chip_ID are also known as the Chip_slot_number. This 4-bit value is used by the Pcall16() and Slot_marker() commands during the anticollision sequence in the Inventory state.

Figure 16. ST25TB04K Chip_ID description



Each time the ST25TB04K receives a Pcall16() command, the Chip_slot_number is given a new 4-bit random value. If the new value is 0000_b, the ST25TB04K 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 ST25TB04K receives the Slot_marker(SN) command, it compares its Chip_slot_number with the Slot_number parameter (SN). If they match, the ST25TB04K returns its Chip_ID as a response to the command. If they do not, the ST25TB04K does not answer. The Slot_marker(SN) command is used to define all the anticollision slot numbers from 1 to 15.

Figure 17. 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 ST25TB04K devices that are present in the reader field range, and that are in Inventory state. Only ST25TB04K devices in Inventory state will respond to the Pcall16() and Slot_marker(SN) anticollision commands.

A new ST25TB04K 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 8 shows the elements of a standard anticollision sequence. (See *Table 9* for an example.)

Table 8. Standard anticollision sequence

Step 1	Init:	Send Initiate(). – If no answer is detected, go to step1. – If only 1 answer is detected, select and access the ST25TB04K. After accessing the ST25TB04K, deselect the tag and go to step1. – If a collision (many answers) is detected, go to step2.
Step 2	Slot 0	Send Pcall16(). – If no answer or collision is detected, go to step3. – If 1 answer is detected, store the Chip_ID, Send Select() and go to step3.
Step 3	Slot 1	Send Slot_marker(1). – If no answer or collision is detected, go to step4. – If 1 answer is detected, store the Chip_ID, Send Select() and go to step4.
Step 4	Slot 2	Send Slot_marker(2). – If no answer or collision is detected, go to step5. – 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) ... – If no answer or collision is detected, go to stepN+1. – If 1 answer is detected, store the Chip_ID, Send Select() and go to stepN+1.
Step 17	Slot 15	Send Slot_marker(15). – If no answer or collision is detected, go to step18. – 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 ST25TB04K one by one. After accessing each ST25TB04K, switch them into Deselected or Deactivated state, depending on the application needs. – If collisions were detected between Step2 and Step17, go to Step2. – If no collision was detected between Step2 and Step17, go to Step1.

After each Slot_marker() command, there may be no answer, one or several answers from the ST25TB04K 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 ST25TB04K by issuing a Select() command containing the desired Chip_ID. If a collision is detected, the reader has to generate a new sequence in order to identify all unidentified ST25TB04K devices in the field. The anticollision sequence can stop when all ST25TB04K devices have been identified.

Table 9 gives an example of anticollision sequence, the gray cells highlight the fact that the related tags are not yet identified. When the tag is identified, the gray color changes to white.

Table 9. Example of an anticollision sequence

Command	Tag1 Chip_ID	Tag2 Chip_ID	Tag3 Chip_ID	Tag4 Chip_ID	Tag5 Chip_ID	Tag6 Chip_ID	Tag7 Chip_ID	Tag8 Chip_ID	Comment
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. All tags answer: collisions
PCALL16()	45h	12h	30h	43h	55h	43h	53h	73h	All CHIP_SLOT_ NUMBERs get a new random value
SELECT(30h)	-	-	30h	-	-	-	-	-	Slot0: only one answer
SLOT_MARKER(1)	-	-	30h	-	-	-	-	-	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: collision
SLOT_MARKER(4)	-	-	-	-	-	-	-	-	Slot4: no answer
SLOT_MARKER(5)	45h	-	-	-	55h	-	-	-	Slot5: collision
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 a new random value
	40h	-	-	-	-	-	50h	-	Slot0: collision
SLOT_MARKER(1)	-	-	-	41h	-	-	-	-	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 a new random value
	-	-	-	-	-	-	50h	-	Slot0: only one answer
SELECT(50h)	-	-	-	-	-	-	50h	-	Tag7 is identified
SLOT_MARKER(1)	41h	-	-	-	-	-	-	-	Slot1: only one answer but already found for tag4
SLOT_MARKER(N)	-	-	-	-	-	-	-	-	SlotN: only one answer
PCALL16()	43h	-	-	-	-	-	-	-	All CHIP_SLOT_ NUMBERs get a new random value
	-	-	-	-	-	-	-	-	Slot0: only one answer
SLOT_MARKER(3)	43h	-	-	-	-	-	-	-	Slot3: only one answer
SELECT(43h)	43h	-	-	-	-	-	-	-	Tag1 is identified
-	-	-	-	-	-	-	-	-	All tags are identified

8 ST25TB04K commands

See the paragraphs below for a detailed description of the commands available on the ST25TB04K. The commands and their hexadecimal codes are summarized in *Table 10*. A brief is given in *Appendix B*.

Table 10. Command code

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

8.1 Initiate() command

Command code = 06h - 00h

Initiate() is used to initiate the anticollision sequence of the ST25TB04K. On receiving the Initiate() command, all ST25TB04K 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 ST25TB04K 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 18. Initiate request format

SOF	Initiate		CRC_L	CRC_H	EOF
	06h	00h	8 bits	8 bits	

AI07670b

Request parameter:

- No parameter

Figure 19. Initiate response format

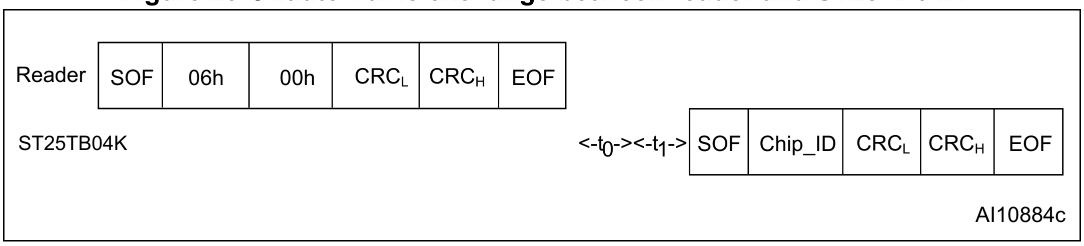
SOF	Chip_ID	CRC_L	CRC_H	EOF
	8 bits	8 bits	8 bits	

AI07671

Response parameter:

- Chip_ID of the ST25TB04K

Figure 20. Initiate frame exchange between reader and ST25TB04K



8.2 Pcall16() command

Command code = 06h - 04h

The ST25TB04K must be in Inventory state to interpret the Pcall16() command.

On receiving the Pcall16() command, the ST25TB04K 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 an 15 (1111_b). The value is retained until a new Pcall16() or Initiate() command is issued, or until the ST25TB04K is powered off. The new Chip_slot_number value is then compared with the value 0000_b. If they match, the ST25TB04K returns its Chip_ID value. If not, the ST25TB04K 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 ST25TB04K device in Inventory state present in the reader field range.

Figure 21. Pcall16 request format

SOF	PCALL16		CRC_L	CRC_H	EOF
	06h	04h	8 bits	8 bits	

AI07673b

Request parameter:

- No parameter

Figure 22. Pcall16 response format

SOF	Chip_ID	CRC_L	CRC_H	EOF
	8 bits	8 bits	8 bits	

AI07671

Response parameter:

- Chip_ID of the ST25TB04K

Figure 23. Pcall16 frame exchange between reader and ST25TB04K

Reader	SOF	06h	04h	CRC_L	CRC_H	EOF							
ST25TB04K							$<-t_0-><-t_1->$	SOF	Chip_ID	CRC_L	CRC_H	EOF	

AI10885c

8.3 Slot_marker(SN) command

Command code = x6h

The ST25TB04K 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 ST25TB04K compares its Chip_slot_number value with the Slot_number value given in the command code. If they match, the ST25TB04K returns its Chip_ID value. If not, the ST25TB04K 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 ST25TB04K device in Inventory state present in the reader field range.

Figure 24. Slot_marker request format

SOF	Slot_marker	CRC_L	CRC_H	EOF
	X6h	8 bits	8 bits	

AI07675b

Request parameter:

- x: Slot number

Figure 25. Slot_marker response format

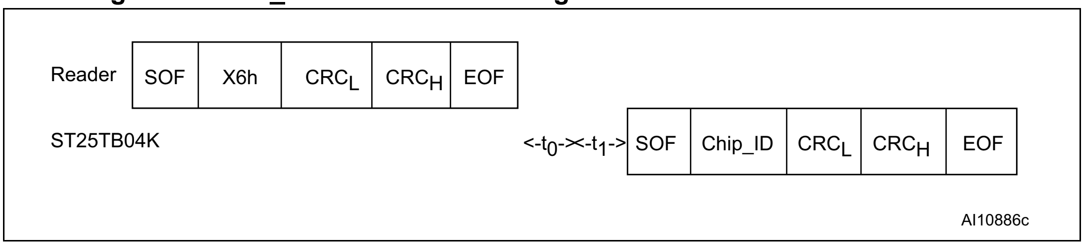
SOF	Chip_ID	CRC_L	CRC_H	EOF
	8 bits	8 bits	8 bits	

AI07671

Response parameters:

- Chip_ID of the ST25TB04K

Figure 26. Slot_marker frame exchange between reader and ST25TB04K



8.4 Select(Chip_ID) command

Command code = 0Eh

The Select() command allows the ST25TB04K to enter the Selected state. Until this command is issued, the ST25TB04K 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 ST25TB04K 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 27. Select request format

SOF	Select	Chip_ID	CRC_L	CRC_H	EOF
	0Eh	8 bits	8 bits	8 bits	

AI07677b

Request parameter:

- 8-bit Chip_ID stored during the anticollision sequence

Figure 28. Select response format

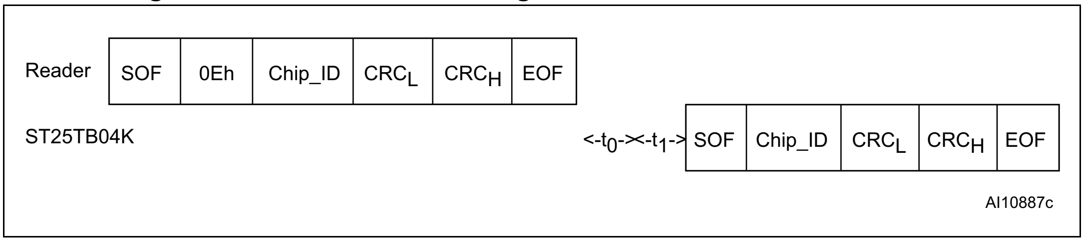
SOF	Chip_ID	CRC_L	CRC_H	EOF
	8 bits	8 bits	8 bits	

AI07677c

Response parameters:

- Chip_ID of the selected tag. Must be equal to the transmitted Chip_ID

Figure 29. Select frame exchange between reader and ST25TB04K



8.5 Completion() command

Command code = 0Fh

On receiving the Completion() command, an ST25TB04K in Selected state switches to Deactivated state and stops decoding any new commands. The ST25TB04K is then locked in this state until a complete reset (tag out of the field range). A new ST25TB04K 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 ST25TB04K devices not in Selected state ignore the Completion() command.

Figure 30. Completion request format

SOF	Completion	CRC_L	CRC_H	EOF
	0Fh	8 bits	8 bits	

AI07679b

Request parameters:

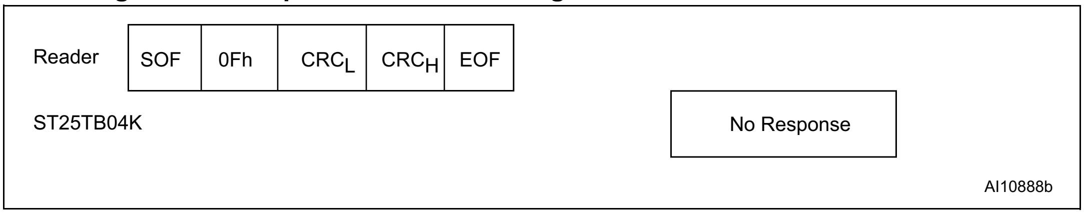
- No parameter

Figure 31. Completion response format

No Response

AI07680

Figure 32. Completion frame exchange between reader and ST25TB04K



8.6 Reset_to_inventory() command

Command code = 0Ch

On receiving the Reset_to_inventory() command, all ST25TB04K devices in Selected state revert to Inventory state. The concerned ST25TB04K devices are thus resubmitted to the anticollision sequence. This command is useful when two ST25TB04K 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 ST25TB04K devices that are not in Selected state ignore the Reset_to_inventory() command.

Figure 33. Reset_to_inventory request format

SOF	RESET_TO_INVENTORY	CRC_L	CRC_H	EOF
	0Ch	8 bits	8 bits	

AI07682

Request parameter:

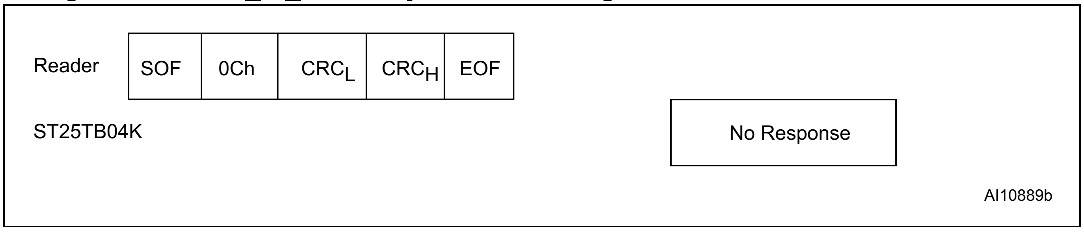
- No parameter

Figure 34. Reset_to_inventory response format

No Response

AI07680

Figure 35. Reset_to_inventory frame exchange between reader and ST25TB04K



8.7 Read_block(Addr) command

Command code = 08h

On receiving the Read_block command, the ST25TB04K 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 ST25TB04K (addresses 0 to 127). Read_block commands issued with a block address above 127 will not be interpreted and the ST25TB04K will not return any response, except for the System area located at address 255.

The ST25TB04K 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 ST25TB04K before a Select() command is issued are ignored.

Figure 36. Read_block request format

SOF	Read_block	Address	CRC_L	CRC_H	EOF
	08h	8 bits	8 bits	8 bits	

AI07684b

Request parameter:

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

Figure 37. Read_block response format

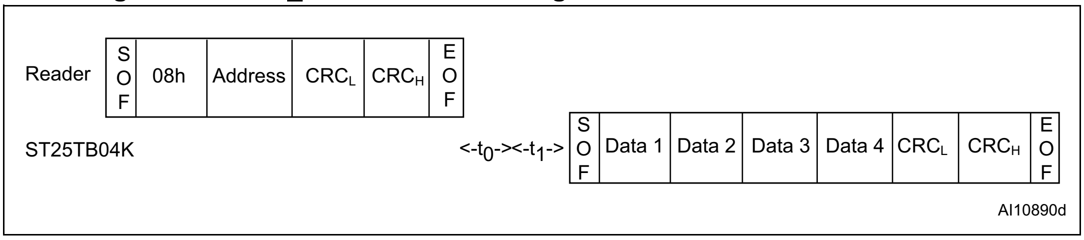
SOF	Data 1	Data 2	Data 3	Data 4	CRC_L	CRC_H	EOF
	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	

AI07685b

Response parameters:

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

Figure 38. Read_block frame exchange between reader and ST25TB04K



8.8 Write_block (Addr, Data) command

Command code = 09h

On receiving the Write_block command, the ST25TB04K 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 ST25TB04K (addresses 0 to 127). Write_block commands issued with a block address above 127 will not be interpreted and the ST25TB04K 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:

- *Table 4: Resettable OTP area (addresses 0 to 4)*.
- *Table 5: Binary counter (addresses 5 to 6)*.
- *Table 6: EEPROM (addresses 7 to 127)*.

The Write_block command does not give rise to a response from the ST25TB04K. The reader must check after the programming time, t_W, that the data was correctly programmed. The ST25TB04K 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 ST25TB04K before a Select() command is issued, are ignored.

Figure 39. Write_block request format

SOF	Write_block	Address	Data 1	Data 2	Data 3	Data 4	CRC_L	CRC_H	EOF
	09h	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	

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 40. Write_block response format

No Response

AI07680

Figure 41. Write_block frame exchange between reader and ST25TB04K

Reader	SOF	09h	Address	Data 1	Data 2	Data 3	Data 4	CRC_L	CRC_H	EOF	
ST25TB04K											No Response

AI10891g

8.9 Get_UID() command

Command code = 0Bh

On receiving the Get_UID command, the ST25TB04K 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 ST25TB04K 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 ST25TB04K before a Select() command is issued, are ignored.

Figure 42. Get_UID request format

SOF	Get_IUD	CRC_L	CRC_H	EOF
	0Bh	8 bits	8 bits	

AI07693b

Request parameter:

- No parameter

Figure 43. Get_UID response format

SOF	UID 0	UID 1	UID 2	UID 3	UID 4	UID 5	UID 6	UID 7	CRC_L	CRC_H	EOF
	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	8 bits	

AI07694

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 ST25TB04K family are uniquely identified by a 64-bit unique identifier (UID). This is used for addressing each ST25TB04K 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 44*):

- 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 8-bit product ref code set to 1Fh for ST25TB04K
- a 40-bit unique serial number

Figure 44. 64-bit unique identifier of the ST25TB04K

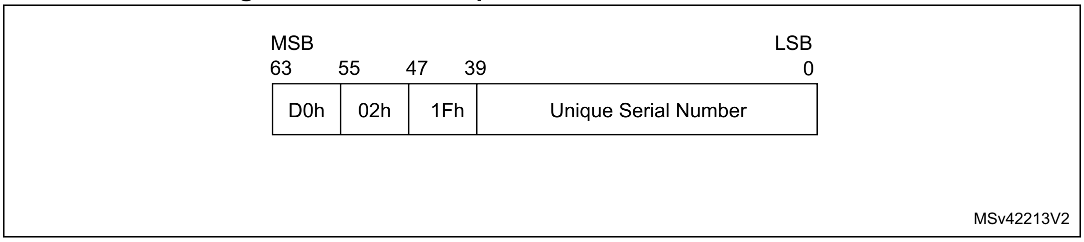
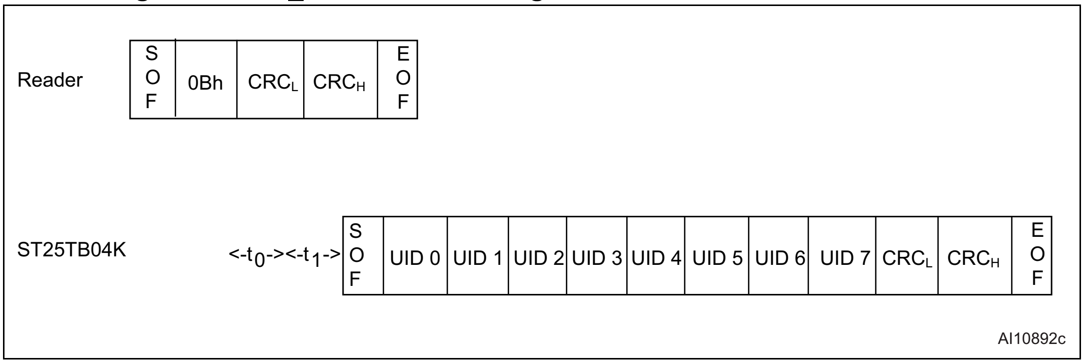


Figure 45. Get_UID frame exchange between reader and ST25TB04K



8.10 Power-on state

After power-on, the ST25TB04K 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().

9 Maximum ratings

Stressing the device above the ratings 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 ratings conditions for extended periods may affect device reliability. Refer also to the STMicroelectronics SURE Program and other relevant quality documents.

Table 11. Absolute maximum ratings

Symbol	Parameter		Min.	Max.	Unit
T_{STG}, t_{STG}	Storage conditions	Sawn wafer (kept in its original packing form)	15	25	°C
			-	9[1]	months
		Unsawn wafer (kept in its antistatic bag)	19	25	°C
			-	23	months
I_{CC}	Supply current on AC0 / AC1	-	-	40	mA
V_{MAX}[2]	RF input voltage amplitude between AC0 and AC1, GND pad left floating	-	-	10	V
V_{ESD}	Electrostatic discharge voltage	Human Body Model[3]	-	2000	V

1. Counted from ST shipment date.
2. Based on characterization, not tested in production.
3. Positive and negative pulses applied on different combinations of pin connections, according to AEC-Q100-002 (compliant with ANSI/ESDA/JEDEC JS-001-2012, C1=100 pF, R1=1500 Ω, R2=500 Ω).

10 RF electrical parameters

Table 12. Operating conditions

Symbol	Parameter	Min.	Max.	Unit
T_A	Ambient operating temperature	-40	85	°C

Table 13. Electrical characteristics

Symbol	Parameter	Condition	Min	Typ	Max	Unit
H_ISO	Operating field according to ISO	T_A = 0 °C to 50 °C	1500	-	7500	mA/m
H_extended	Operating field in extended temperature range	T_A = -40 °C to 85 °C	1500	-	7500	
V_{RET}	Back-scattering induced voltage	ISO 10373-6	20	-	-	mV
C_{TUN}	Internal tuning capacitor	13.56 MHz(1)	62	68	74	pF

1. The tuning capacitance value is measured with ST characterization equipement at chip Power On Reset. This value is to be used as reference for antenna design. Min and Max value are deduced from correlation with industrial tester limits.

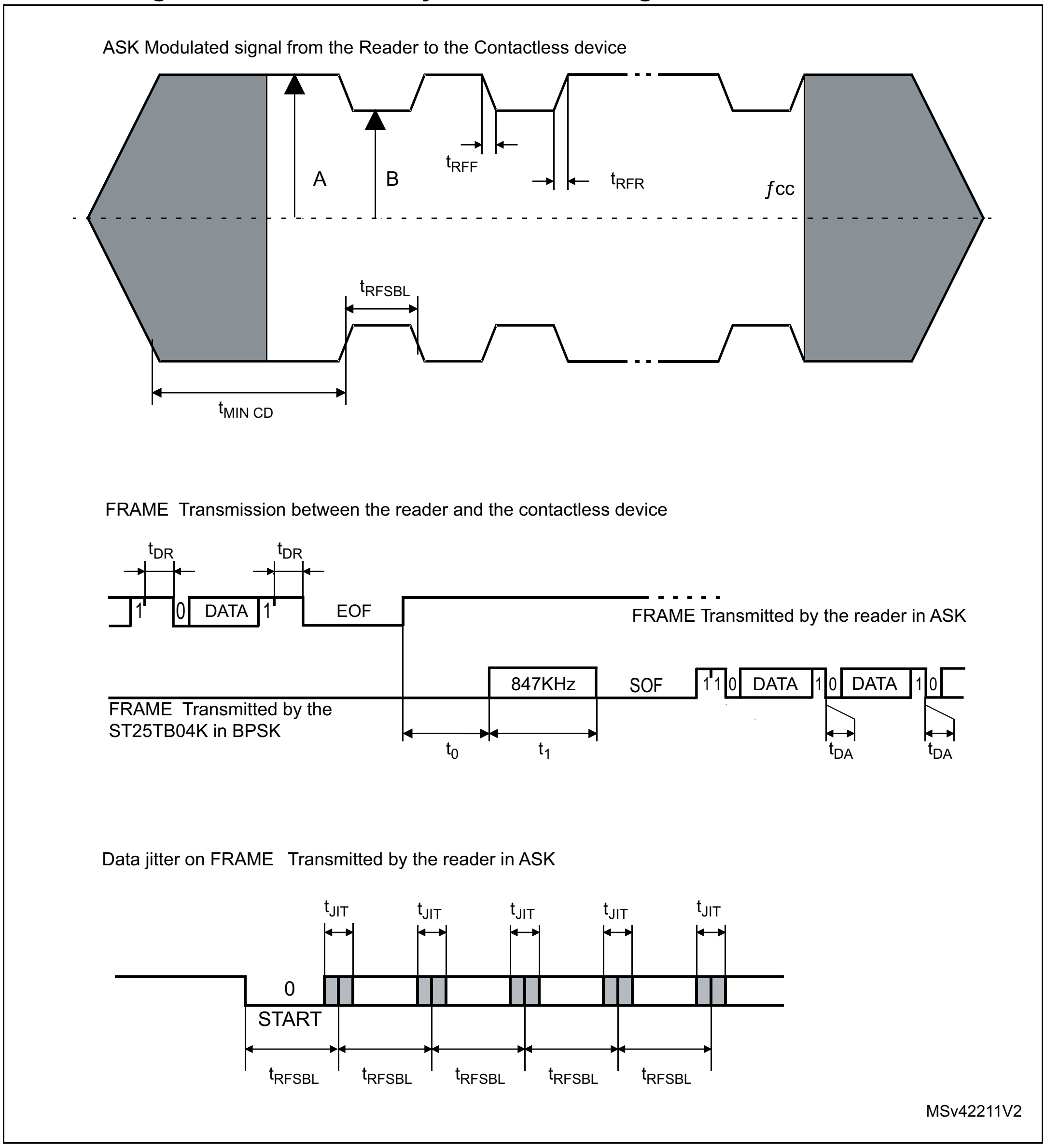
Note: For inlay implementation, the antenna design applied for SRI4K can be re-used as-is for ST25TB04K: typical 68pF value for the ST25TB04K is equivalent to what was specified in the SRI4K data-sheet as 64pF.
This change is related to a different measurement methodology between SRI4K and ST25TB04K.

Table 14. RF characteristics(1)

Symbol	Parameter	Condition	Min	Typ	Max	Unit
f_{CC}	RF carrier frequency	-	13.553	-	13.567	MHz
$MI_{CARRIER}$	Carrier modulation index	MI=(A-B)/(A+B)	8	11	14	%
t_{RFR}, t_{RFF}	10% Rise and Fall times	-	0.1	-	1.25	µs
t_{RFSBL}	Minimum pulse width for Start bit	ETU = 128/f_{CC}	-	9.44	-	µs
t_{JIT}	ASK modulation data jitter	Coupler to ST25TB04K	–2	-	+2	µs
$t_{MIN\ CD}$	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	-	-	159	-	µs
t_1	Synchronization delay	-	-	151	-	µs
t_2	Answer to new request delay	14 ETU	132	-	-	µs
t_{DR}	Time between request characters	Coupler to ST25TB04K	0	-	57	µs
t_{DA}	Time between answer characters	ST25TB04K to coupler	-	0	-	µs
t_W	Programming time for write	With no auto-erase cycle (OTP)	-	-	3	ms
		With auto-erase cycle (EEPROM)	-	-	5	ms
		Binary counter decrement with tearing condition	-	-	7	ms

1. All timing measurements were performed on a reference antenna with the following characteristics:
External size: 76 mm x 46 mm
Number of turns: 4
Width of conductor: 0.9 mm
Space between 2 conductors: 0.9 mm
Tuning Frequency: 13.58 MHz.

Figure 46. ST25TB04K synchronous timing, transmit and receive



11 Part numbering

Table 15. Ordering information scheme (bumped and sawn wafer)

Example: ST25 T B 04K - A C 6 G 6

Device type

ST25 = RF memory

Product type

T = Tags + RFID

Protocol

B = ISO14443-B

Memory density

04K (binary)

Interface

A = none

Features

C = Counter as option

Device grade

6 = -40°C to 85°C

Package/Packaging

G = Bumped 120 um

U = Unsawn 725 um

Capacitor value

6 = 68 pF

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 ST25TB04K command brief

Figure 47. Initiate frame exchange between reader and ST25TB04K

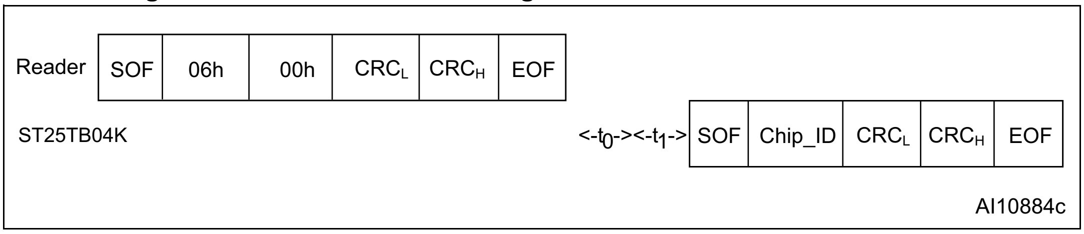


Figure 48. Pcall16 frame exchange between reader and ST25TB04K

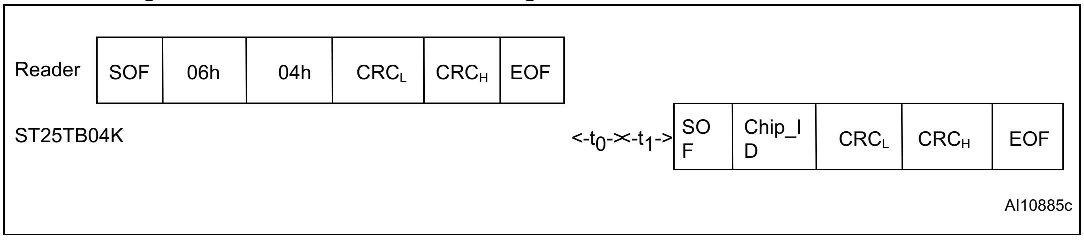


Figure 49. Slot_marker frame exchange between reader and ST25TB04K

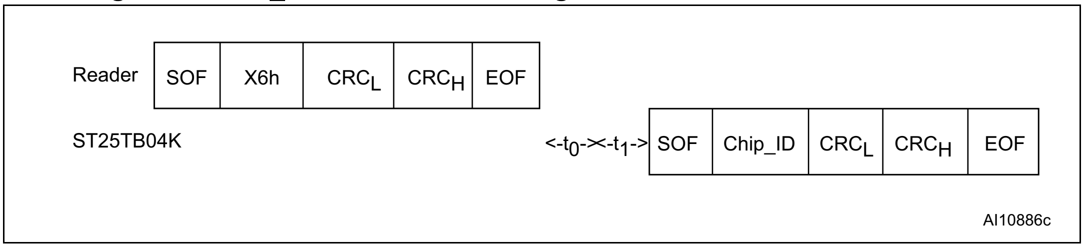


Figure 50. Select frame exchange between reader and ST25TB04K

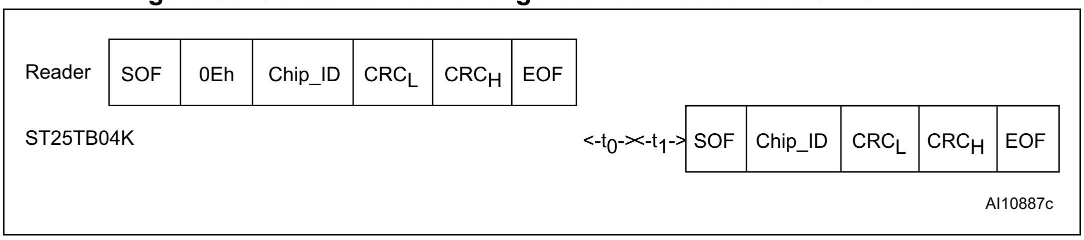


Figure 51. Completion frame exchange between reader and ST25TB04K

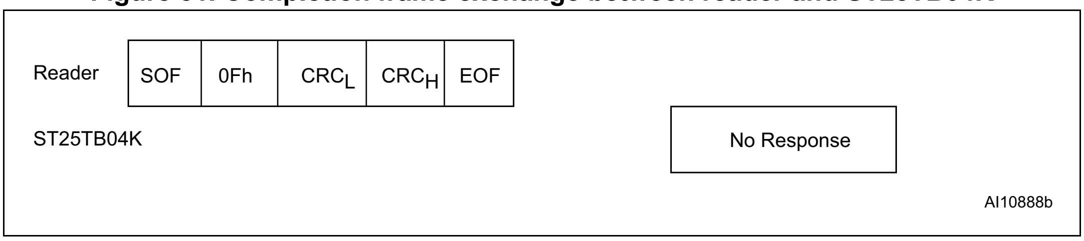


Figure 52. Reset_to_inventory frame exchange between reader and ST25TB04K

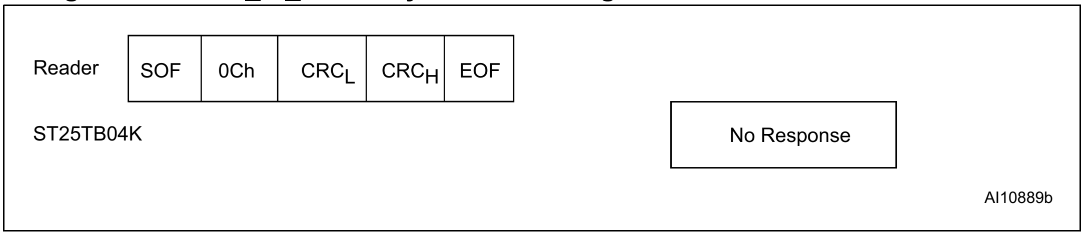


Figure 53. Read_block frame exchange between reader and ST25TB04K

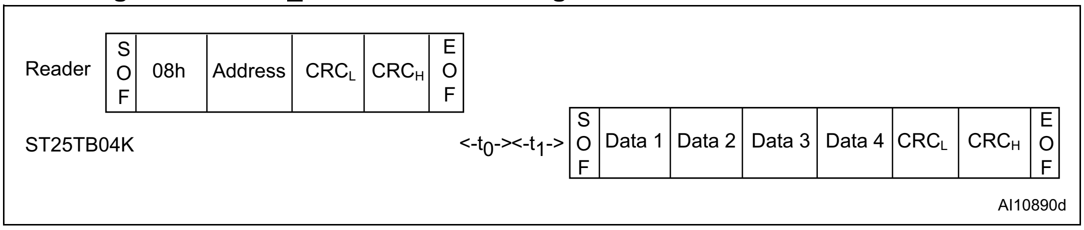


Figure 54. Write_block frame exchange between reader and ST25TB04K

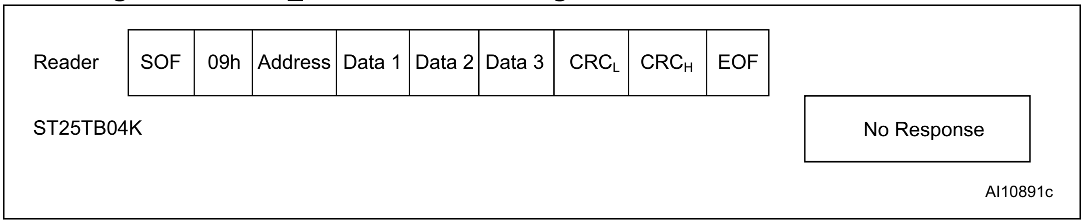
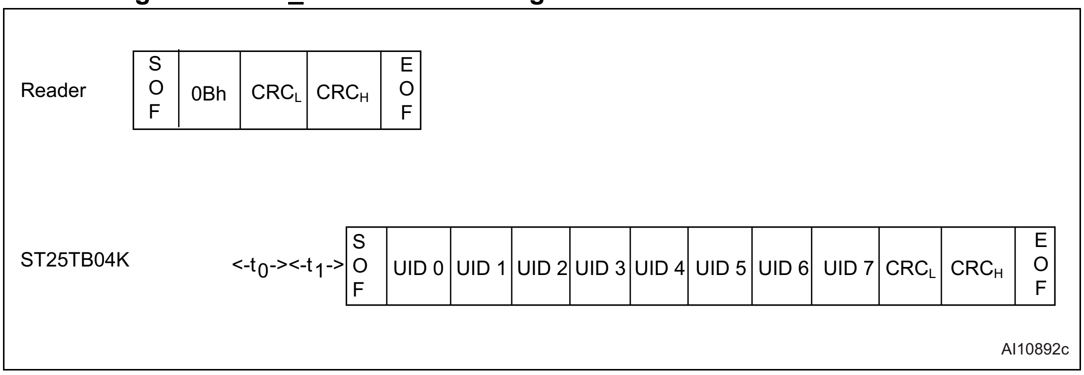


Figure 55. Get_UID frame exchange between reader and ST25TB04K



Revision history

Table 16. Document revision history

Date	Version	Changes
09-Jan-2016	1	Initial release
03-Mar-2016	2	Updated *Figure 28* and *Figure 41*.
19-Apr-2016	3	Changed confidentiality level from ST restricted to public.
21-Sept-2016	4	Updated *Figure 44: 64-bit unique identifier of the ST25TB04K*, *Figure 46: ST25TB04K synchronous timing, transmit and receive*, *Table 11: Absolute maximum ratings*, *Table 15: Ordering information scheme (bumped and sawn wafer)* and *Section 8.9: Get_UID() command*
24-Nov-2016	5	Updated *Features* in cover page