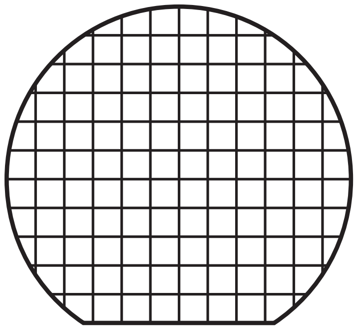
# SRT512

## 13.56 MHz short-range contactless memory chip with 512-bit EEPROM and anticollision functions

**Datasheet − production data**

## 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
- 512-bit EEPROM with write protect feature
- Read_block and Write_block (32 bits)
- Internal tuning capacitor
- 1 million erase/write cycles
- 40-year data retention
- Self-timed programming cycle
- 5 ms typical programming time



– Unsawn wafer
– Bumped and sawn wafer

## Applications

- Transport



This is information on a product in full production. www.st.com

# Contents

# List of tables

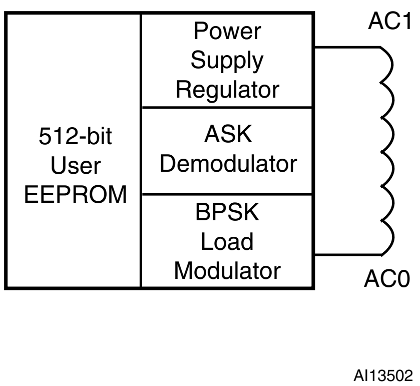
# List of figures

# 1 Description

The SRT512 is a contactless memory, powered by an externally transmitted radio wave. It contains a 512-bit user EEPROM. The memory is organized as 16 blocks of 32 bits. The SRT512 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 SRT512 and the reader is 106 Kbit/s in both reception and emission modes.

The SRT512 follows the ISO 14443-2 Type B recommendation for the radio-frequency power and signal interface.

**Figure 1. Logic diagram**




The SRT512 is specifically designed for short range applications that need re-usable products. The SRT512 includes an anticollision mechanism that allows it to detect and select tags present at the same time within range of the reader.

**Table 1. Signal names**

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

The SRT512 contactless EEPROM can be randomly read and written in block mode (each block containing 32 bits). The instruction set includes the following nine commands:

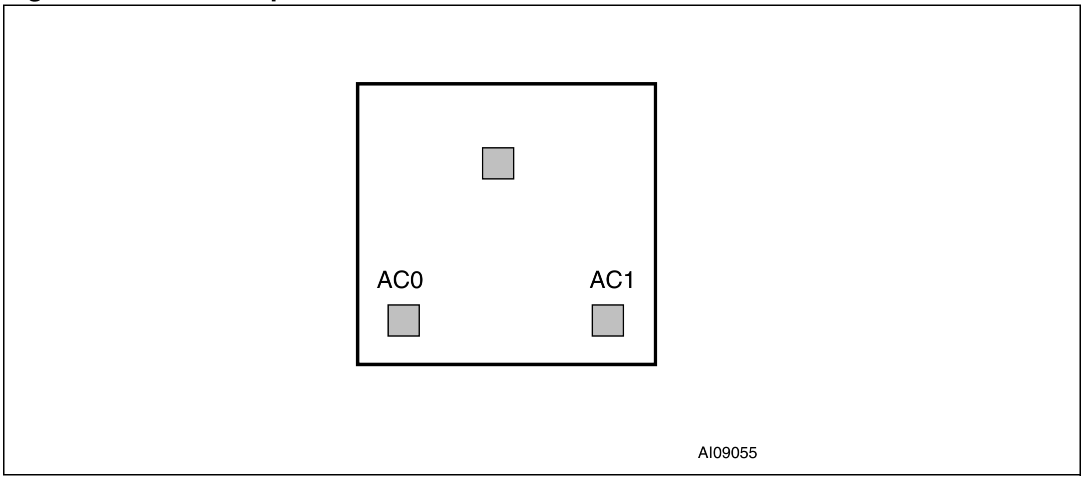
- Read_block
- Write_block
- Initiate
- Pcall16
- Slot_marker
- Select
- Completion
- Reset_to_inventory
- Get_UID

The SRT512 memory is organized in three areas, as described in *Table 12*. The first area is an EEPROM area where all blocks behave as User blocks.

The second area provides two 32-bit binary counters that 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

## 2.1 AC1, AC0

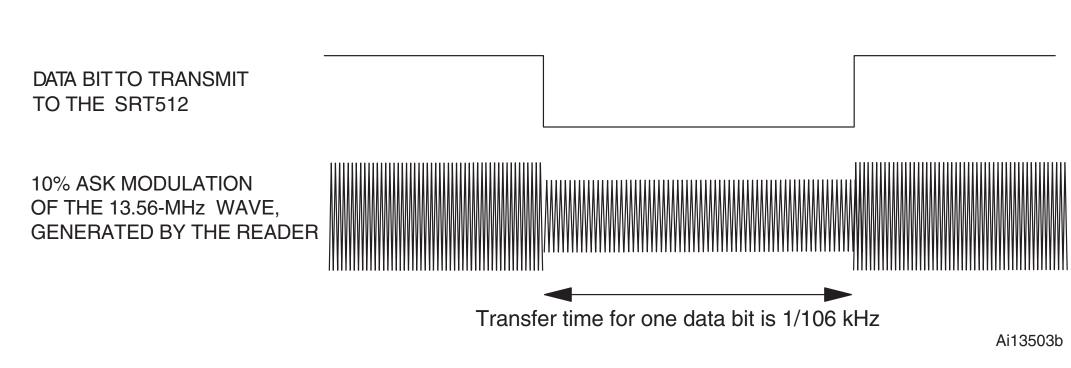
The pads for the antenna coil. AC1 and AC0 must be directly bonded to the antenna.

# 3 Data transfer

## 3.1 Input data transfer from reader to SRT512 (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 SRT512’s antenna is transformed into a supply voltage by a regulator, and into data bits by the ASK demodulator. For the SRT512 to decode correctly the information it receives, the reader must 10% amplitude-modulate the 13.56 MHz wave before sending it to the SRT512. 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 SRT512 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 SRT512 does not execute the command, but it does not generate an error frame.

**Figure 4. SRT512 request frame character format**

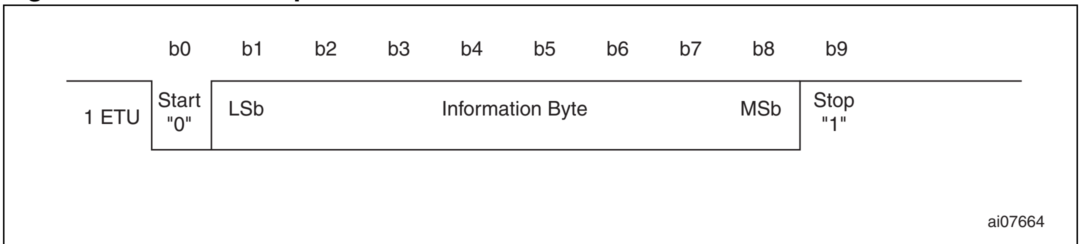


Table 2. Bit description

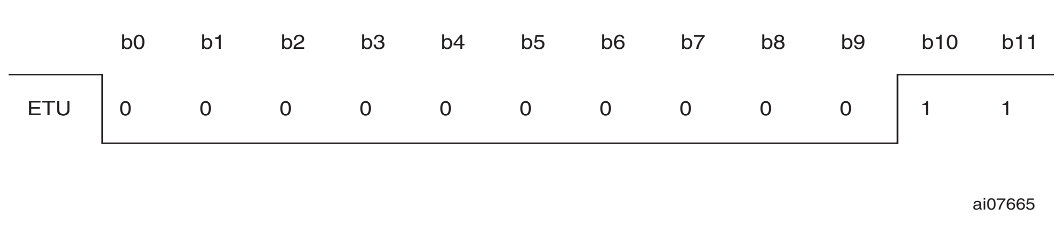
| Bit | Description | Value |
|---|---|---|
| $b_0$ | Start bit used to synchronize the transmission | $b_0 = 0$ |
| $b_1$ to $b_8$ | Information byte (command, address or data) | The information byte is sent with the least significant bit first |
| $b_9$ | Stop bit used to indicate the end of a character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

### 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

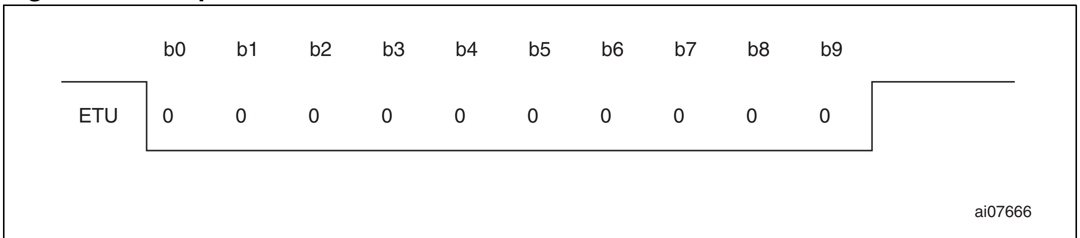



### 3.1.3 Request end of frame

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.

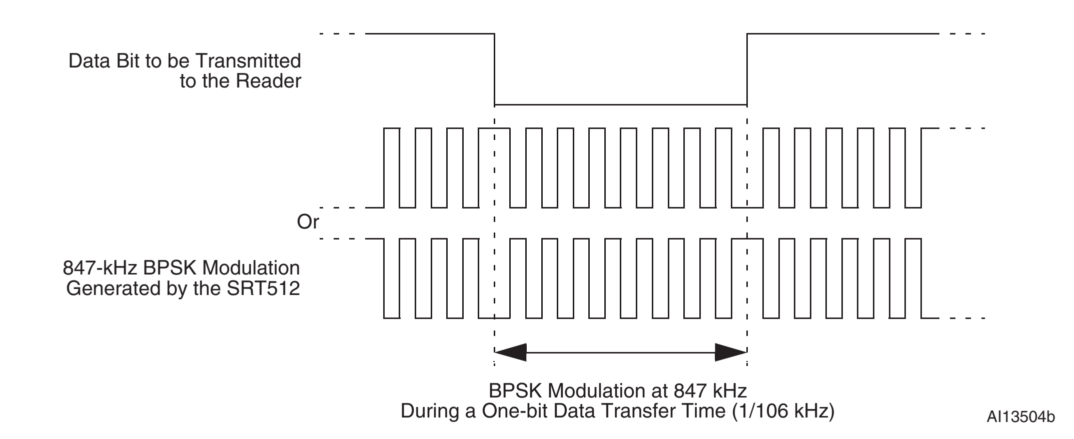
Figure 6. Request end of frame



## 3.2 Output data transfer from SRT512 to reader (answer frame)

The data bits issued by the SRT512 use back-scattering. Back-scattering is obtained by modifying the SRT512 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 SRT512. 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

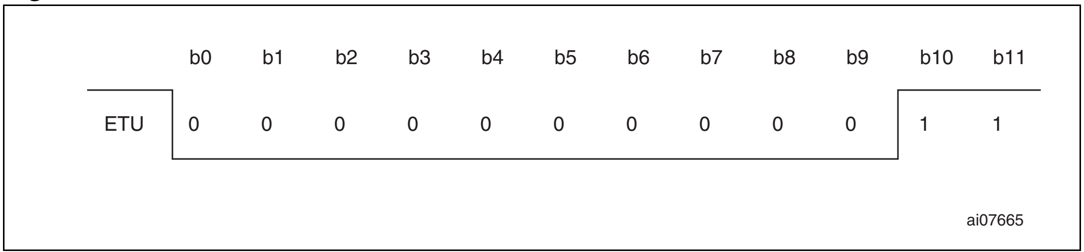
The character format is the same as for input data transfer (*Figure 4*). The transmitted frames are made up of an SOF, data, a CRC and an EOF (*Figure 10*). As with an input data transfer, if an error occurs, the reader does not issue an error code to the SRT512, but it should be able to detect it and 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**

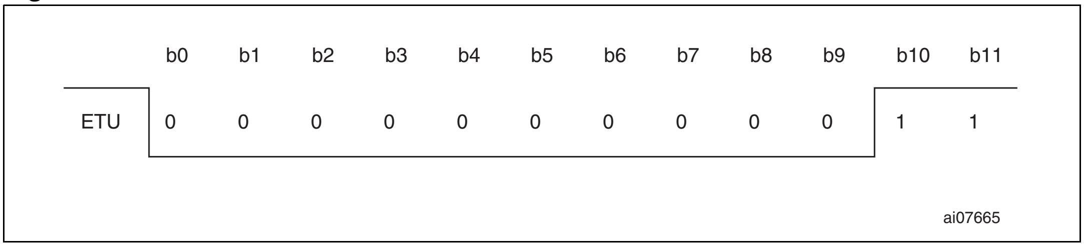


### 3.2.3 Answer end of frame

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

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

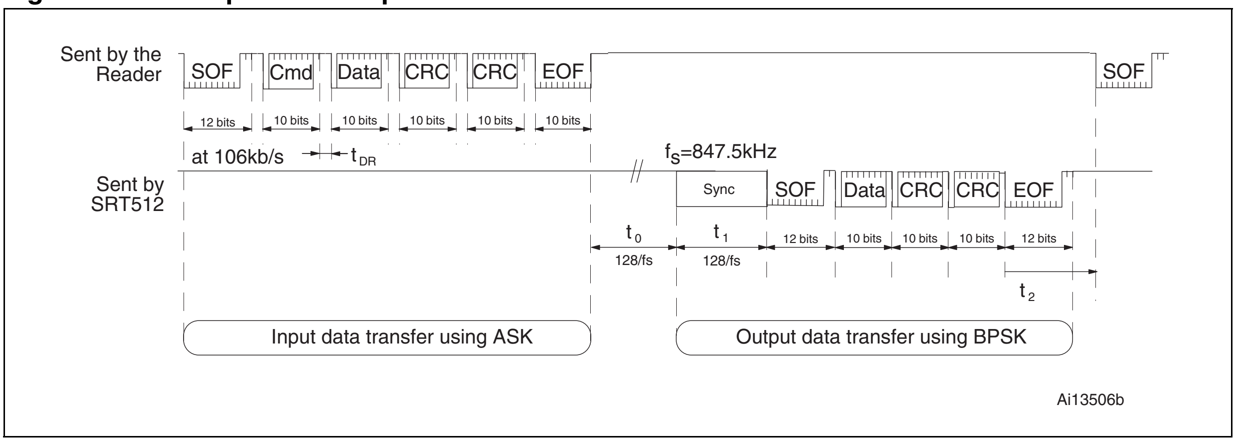
**Figure 9. Answer end of frame**




## 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 SRT512 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 SRT512 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 SRT512.

**Figure 10. Example of a complete transmission frame**



## 3.4 CRC

The 16-bit CRC used by the SRT512 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.

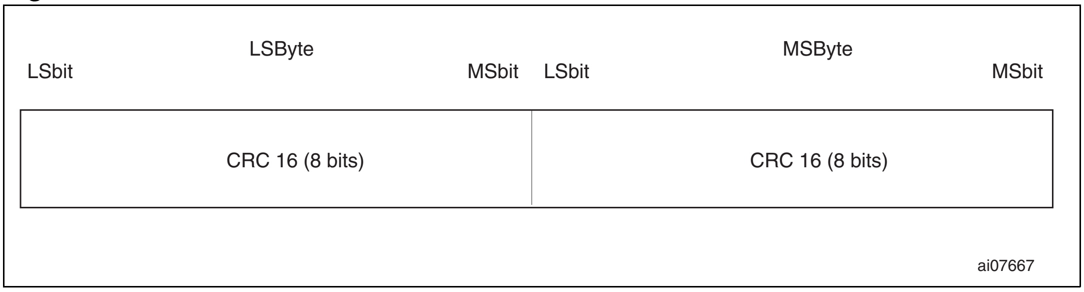
The two-byte CRC is present in every request and in every answer frame, before the EOF. The CRC is calculated on all the bytes between SOF (not included) and the CRC field.

Upon reception of a request from a reader, the SRT512 verifies that the CRC value is valid. If it is invalid, the SRT512 discards the frame and does not answer the reader.

Upon reception of an answer from the SRT512, 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 SRT512 is organized as 16 blocks of 32 bits as shown in *Table 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. SRT512 memory mapping**

| Block Addr | MSB $b_{31}$ — 32-bit block — $b_{16}$ $b_{15}$ $b_{14}$ $b_8$ $b_7$ — LSB $b_0$ | | | | Description |
|---|---|---|---|---|---|
| 0 | User area | | | | lockable EEPROM |
| 1 | User area | | | | |
| 2 | User area | | | | |
| 3 | User area | | | | |
| 4 | User area | | | | |
| 5 | 32 bits binary counter | | | | Count down counter |
| 6 | 32 bits binary counter | | | | |
| 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 | | | | |
| | | | | | |
| 255 | OTP_Lock_Reg | 1 | ST Reserved | Fixed Chip_ID (Option) | System OTP bits |
| | | | | | |
| UID0 | 64 bits UID area | | | | ROM |
| UID1 | | | | | |

## 4.1 EEPROM area

Blocks 0 to 4 define a User area. They behave as standard EEPROM blocks, like blocks 7 to 15 as described in *Figure 13*. Each block can be individually write-protected using the OTP_Lock_Reg bits of the system area. Once a block has been protected, it can no longer be unprotected.

**Figure 13. Lockable EEPROM area (addresses 0 to 4)**

| Block address | MSb b31 — 32-bit block b16 b15 b14 — b8 b7 — LSb b0 | Description |
|---|---|---|
| 0 | User area | Lockable EEPROM |
| 1 | User area | |
| 2 | User area | |
| 3 | User area | |
| 4 | User area | |

ai12382b

## 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 SRT512 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 is FFFF FFFEh in counter 5 and, 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 15* 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.

Blocks 5 and 6 can be write-protected using the OTP_Lock_Reg bits (block 255). Once a block has been protected, its contents cannot be modified. A protected counter block behaves like a ROM block.

**Figure 14. Binary counter (addresses 5 to 6)**

| Block address | MSb b31 — 32-bit block b16 b15 b14 — b8 b7 — LSb b0 | Description |
|---|---|---|
| 5 | 32-bit binary counter | Count down counter |
| 6 | 32-bit binary counter | |

ai12384b

**Figure 15. Count down 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

## 4.3 EEPROM area

The 9 blocks between addresses 7 and 15 are EEPROM blocks of 32 bits each (36 bytes in total). (See *Figure 16* 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 9 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 16. EEPROM (addresses 7 to 15)**

| Block address | MSb b31 — 32-bit block b16 b15 b14 — b8 b7 — LSb b0 | Description |
|---|---|---|
| 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 | |

Ai12383b

## 4.4 System area

This area is used to modify the settings of the SRT512. It contains 3 registers: OTP_Lock_Reg, Fixed Chip_ID and ST Reserved. See *Figure 17* 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 17. System area**

| Block address | MSB | 32-bit block | | LSB | Description |
|---|---|---|---|---|---|
| | b31 ... b16 | b15 | b14 ... b8 | b7 ... b0 | |
| 255 | OTP_Lock_Reg | 1 | ST reserved | Fixed Chip_ID (Option) | OTP |

ai13505b

### 4.4.1 OTP_Lock_Reg

The 16 bits, b31 to b16, of the System area (block address 255) are used as OTP_Lock_Reg bits in the SRT512. They control the write access to the 16 blocks 0 to 15 as follows:

- When b16 is at 0, block 0 is write-protected
- When b17 is at 0, block 1 is write-protected
- When b18 is at 0, block 2 is write-protected
- When b19 is at 0, block 3 is write-protected
- When b20 is at 0, block 4 is write-protected
- When b21 is at 0, block 5 is write-protected
- When b22 is at 0, block 6 is write-protected
- When b23 is at 0, block 7 is write-protected
- When b24 is at 0, block 8 is write-protected
- When b25 is at 0, block 9 is write-protected
- When b26 is at 0, block 10 is write-protected
- When b27 is at 0, block 11 is write-protected
- When b28 is at 0, block 12 is write-protected
- When b29 is at 0, block 13 is write-protected
- When b30 is at 0, block 14 is write-protected
- When b31 is at 0, block 15 is write-protected.

The OTP_Lock_Reg bits cannot be erased. Once write-protected, the 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 SRT512 in order to load the block write protection into the logic.

This bit is set by ST during production tests on customer request. It cannot be modified by the user.

### 4.4.2 Fixed Chip_ID (Option)

The SRT512 is provided with an anticollision feature based on a random 8-bit Chip_ID. Prior to selecting an SRT512, an anticollision sequence has to be run to search for the Chip_ID of the SRT512. 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 SRT512 Chip_ID beforehand, so that the SRT512 can be identified and selected directly without having to run an anticollision sequence. This is why the SRT512 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 SRT512 operation

All commands, data and CRC are transmitted to the SRT512 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 SRT512 at the antenna is demodulated by the 10% ASK demodulator, and decoded by the internal logic. Prior to any operation, the SRT512 must have been selected by a Select command. Each frame transmitted to the SRT512 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 SRT512 (wrong command or CRC error), the memory does not return any error code.

When a valid frame is received, the SRT512 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 SRT512 sending the 2 CRC bytes and the EOF.

# 6 SRT512 states

The SRT512 can be switched into different states. Depending on the current state of the SRT512, its logic will only answer to specific commands. These states are mainly used during the anticollision sequence, to identify and to access the SRT512 in a very short time. The SRT512 provides 6 different states, as described in the following paragraphs and in *Figure 18*.

## 6.1 Power-off state

The SRT512 is in Power-off state when the electromagnetic field around the tag is not strong enough. In this state, the SRT512 does not respond to any command.

## 6.2 Ready state

When the electromagnetic field is strong enough, the SRT512 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 SRT512.

## 6.3 Inventory state

The SRT512 switches from the Ready to the Inventory state after an Initiate() command has been issued. In Inventory state, the SRT512 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 SRT512 is active and responds to all Read_block(), Write_block(), and Get_UID() commands. When an SRT512 has entered the Selected state, it no longer responds to anticollision commands. So that the reader can access another tag, the SRT512 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 SRT512 can be in Selected state at a time.

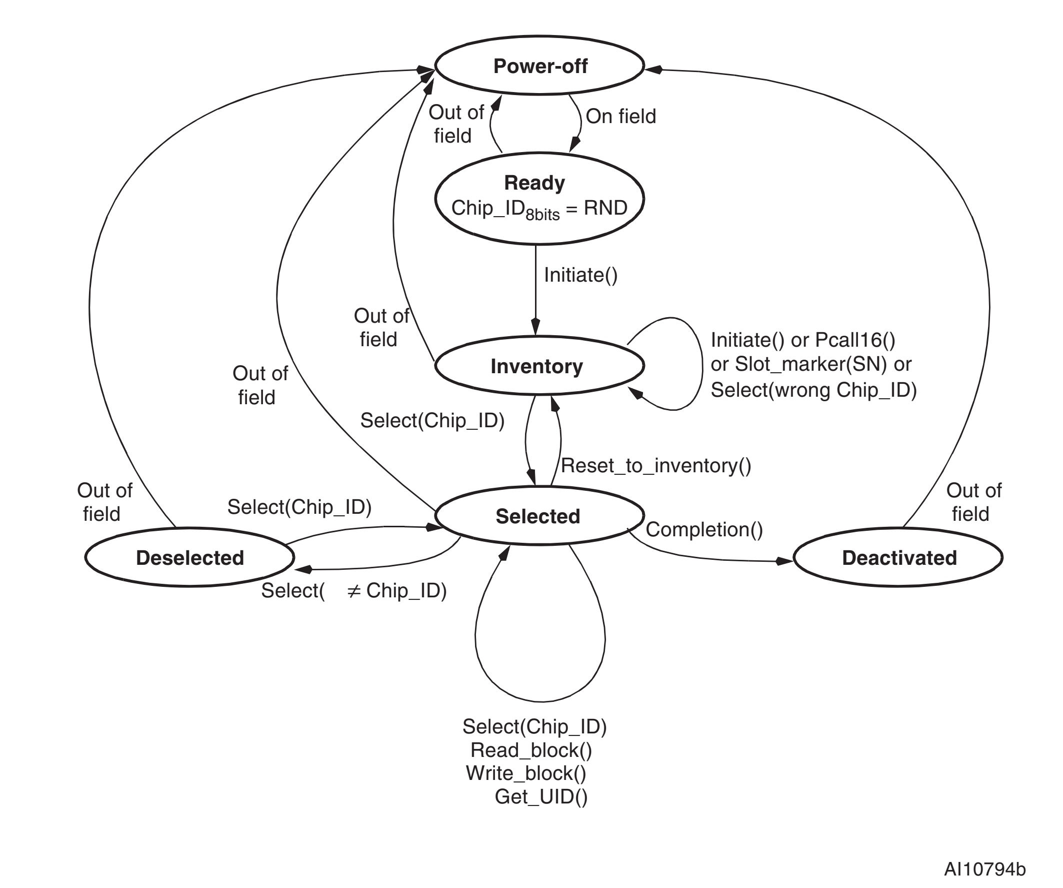
## 6.5 Deselected state

Once the SRT512 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 SRT512 can only be turned off. All commands are ignored.

**Figure 18. State transition diagram**



# 7 Anticollision

The SRT512 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 SRT512 individually, and access its memory. The anticollision sequence is managed by the reader through a set of commands described in *Section 5: SRT512 operation*:

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

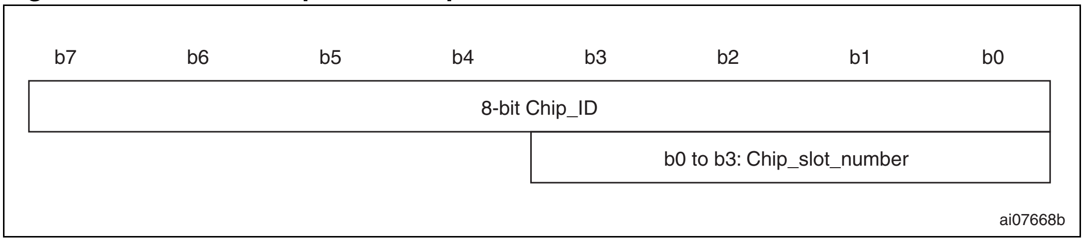
The reader is the master of the communication with one or more SRT512 device(s). It initiates the tag communication activity by issuing an Initiate(), Pcall16() or Slot_marker() command to prompt the SRT512 to answer. During the anticollision sequence, it might happen that two or more SRT512 devices respond simultaneously, so causing a collision. The command set allows the reader to handle the sequence, to separate SRT512 transmissions into different time slots. Once the anticollision sequence has completed, SRT512 communication is fully under the control of the reader, allowing only one SRT512 to transmit at a time.

The Anticollision scheme is based on the definition of time slots during which the SRT512 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 SRT512 answers after the command is issued. SRT512 devices are allowed to answer only once during the anticollision sequence. Consequently, even if there are several SRT512 devices present in the reader field, there will probably be a slot in which only one SRT512 answers, allowing the reader to capture its Chip_ID. Using the Chip_ID, the reader can then establish a communication channel with the identified SRT512. The purpose of the anticollision sequence is to allow the reader to select one SRT512 at a time.

The SRT512 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.

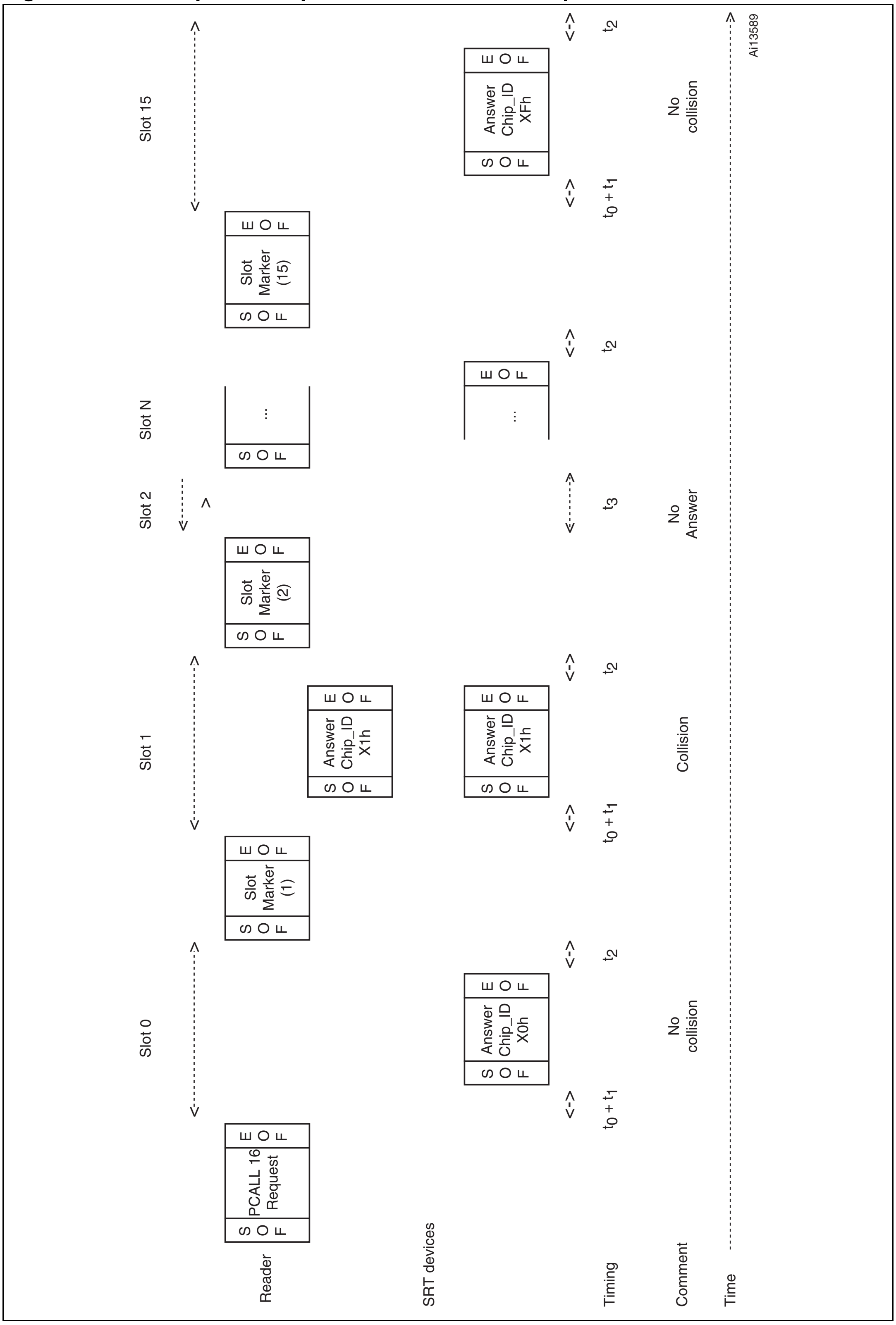
The four least significant bits ($b_0$ to $b_3$) of the Chip_ID are also known as the Chip_slot_number. This 4-bit value is used by the Pcall16() and Slot_marker() commands during the anticollision sequence in the Inventory state.

**Figure 19. SRT512 Chip_ID description**




Each time the SRT512 receives a Pcall16() command, the Chip_slot_number is given a new 4-bit random value. If the new value is $0000_b$, the SRT512 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 SRT512 receives the Slot_marker(SN) command, it compares its Chip_slot_number with the Slot_number parameter (SN). If they match, the SRT512 returns its Chip_ID as a response to the command. If they do not, the SRT512 does not answer. The Slot_marker(SN) command is used to define all the anticollision slot numbers from 1 to 15.

**Figure 20. 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 SRT512 devices that are present in the reader field range, and that are in Inventory state. Only SRT512 devices in Inventory state will respond to the Pcall16() and Slot_marker(SN) anticollision commands.

A new SRT512 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 21* 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 SRT512. After accessing the SRT512, 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 SRT512 one by one. After accessing each SRT512, 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 SRT512 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 SRT512 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 SRT512 devices in the field. The anticollision sequence can stop when all SRT512 devices have been identified.

## Figure 21. Example of an anticollision sequence

| Command | Tag 1 Chip_ID | Tag 2 Chip_ID | Tag 3 Chip_ID | Tag 4 Chip_ID | Tag 5 Chip_ID | Tag 6 Chip_ID | Tag 7 Chip_ID | Tag 8 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 a new random value |
| | | | 30h | | | | | | 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 a new random value |
| | 40h | | | | | | 50h | | Slot0: collisions |
| 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: no 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 |

ai07669

# 8 SRT512 commands

See the paragraphs below for a detailed description of the Commands available on the SRT512. 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) |
| 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 SRT512. On receiving the Initiate() command, all SRT512 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 SRT512 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 22. Initiate request format**

| SOF | Initiate | | $CRC_L$ | $CRC_H$ | EOF |
|---|---|---|---|---|---|
| | 06h | 00h | 8 bits | 8 bits | |

AI07670b

Request parameter:

- No parameter

**Figure 23. Initiate response format**

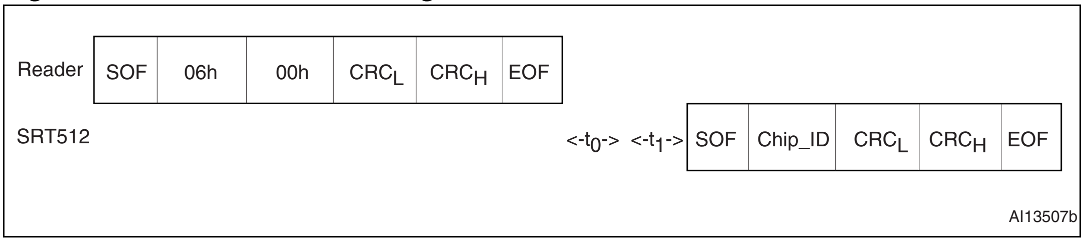
| SOF | Chip_ID | $CRC_L$ | $CRC_H$ | EOF |
|---|---|---|---|---|
| | 8 bits | 8 bits | 8 bits | |

AI07671

Response parameter:

- Chip_ID of the SRT512

**Figure 24. Initiate frame exchange between reader and SRT512**



## 8.2 Pcall16() command

Command code = 06h - 04h

The SRT512 must be in Inventory state to interpret the Pcall16() command.

On receiving the Pcall16() command, the SRT512 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 SRT512 is powered off. The new Chip_slot_number value is then compared with the value $0000_b$. If they match, the SRT512 returns its Chip_ID value. If not, the SRT512 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 SRT512 device in Inventory state present in the reader field range.

**Figure 25. Pcall16 request format**

| SOF | Pcall16 | | $CRC_L$ | $CRC_H$ | EOF |
|---|---|---|---|---|---|
| | 06h | 04h | 8 bits | 8 bits | |

AI07673b

Request parameter:

- No parameter

**Figure 26. Pcall16 response format**

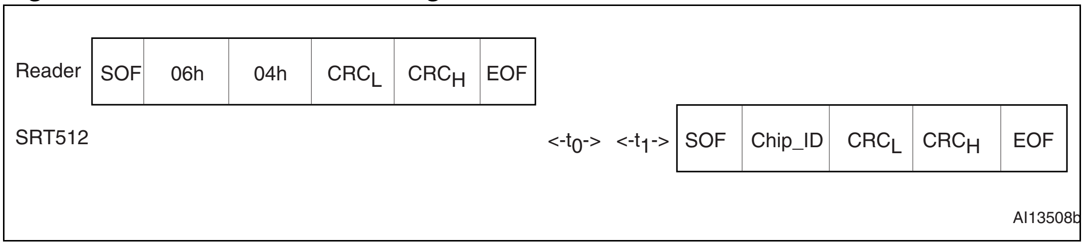
| SOF | Chip_ID | $CRC_L$ | $CRC_H$ | EOF |
|---|---|---|---|---|
| | 8 bits | 8 bits | 8 bits | |

AI07671

Response parameter:

- Chip_ID of the SRT512

**Figure 27. Pcall16 frame exchange between reader and SRT512**



## 8.3 Slot_marker(SN) command

Command code = x6h

The SRT512 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 SRT512 compares its Chip_slot_number value with the Slot_number value given in the command code. If they match, the SRT512 returns its Chip_ID value. If not, the SRT512 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 SRT512 device in Inventory state present in the reader field range.

**Figure 28. Slot_marker request format**

| SOF | Slot_marker | $CRC_L$ | $CRC_H$ | EOF |
|---|---|---|---|---|
| | X6h | 8 bits | 8 bits | |

AI07675b

Request parameter:

- x: Slot number

**Figure 29. 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 SRT512

**Figure 30. Slot_marker frame exchange between reader and SRT512**

| | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reader | SOF | X6h | $CRC_L$ | $CRC_H$ | EOF | | | | | | | |
| SRT512 | | | | | | <-$t_0$-> | <-$t_1$-> | SOF | Chip_ID | $CRC_L$ | $CRC_H$ | EOF |

AI13509b

## 8.4 Select(Chip_ID) command

Command code = 0Eh

The Select() command allows the SRT512 to enter the Selected state. Until this command is issued, the SRT512 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 SRT512 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 31. 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 32. Select response format**

| SOF | Chip_ID | $CRC_L$ | $CRC_H$ | EOF |
|---|---|---|---|---|
| | 8 bits | 8 bits | 8 bits | |

AI07671

Response parameters:

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

**Figure 33. Select frame exchange between reader and SRT512**

| | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Reader | SOF | 0Eh | Chip_ID | $CRC_L$ | $CRC_H$ | EOF | | | | | | | |
| SRT512 | | | | | | | <-$t_0$-> | <-$t_1$-> | SOF | Chip_ID | $CRC_L$ | $CRC_H$ | EOF |

AI13510b

## 8.5 Completion() command

Command code = 0Fh

On receiving the Completion() command, an SRT512 in Selected state switches to Deactivated state and stops decoding any new commands. The SRT512 is then locked in this state until a complete reset (tag out of the field range). A new SRT512 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 SRT512 devices not in Selected state ignore the Completion() command.

**Figure 34. Completion request format**

| SOF | Completion | $CRC_L$ | $CRC_H$ | EOF |
|---|---|---|---|---|
| | 0Fh | 8 bits | 8 bits | |

AI07679b

Request parameters:

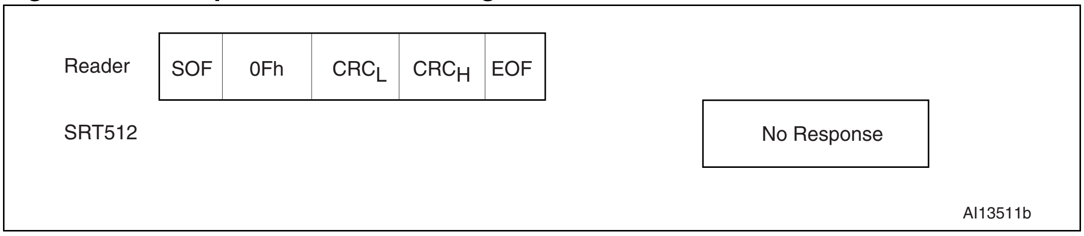
- No parameter

**Figure 35. Completion response format**

| No Response |
|---|
| |

AI07680

**Figure 36. Completion frame exchange between reader and SRT512**



## 8.6 Reset_to_inventory() command

Command code = 0Ch

On receiving the Reset_to_inventory() command, all SRT512 devices in Selected state revert to Inventory state. The concerned SRT512 devices are thus resubmitted to the anticollision sequence. This command is useful when two SRT512 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 SRT512 devices that are not in Selected state ignore the Reset_to_inventory() command.

**Figure 37. Reset_to_inventory request format**

| SOF | Reset_to_inventory | $CRC_L$ | $CRC_H$ | EOF |
|---|---|---|---|---|
| | 0Ch | 8 bits | 8 bits | |

AI07682b

Request parameter:

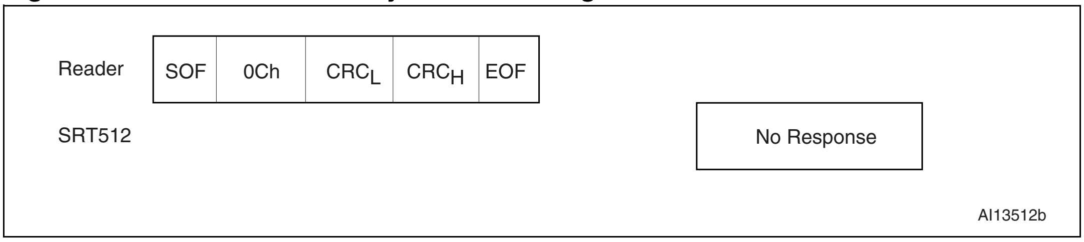
- No parameter

**Figure 38. Reset_to_inventory response format**

| No Response |
|---|
| |

AI07680

**Figure 39. Reset_to_inventory frame exchange between reader and SRT512**



## 8.7 Read_block(Addr) command

Command code = 08h

On receiving the Read_block command, the SRT512 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 16 blocks of the SRT512 (addresses 0 to 15). Read_block commands issued with a block address above 15 will not be interpreted and the SRT512 will not return any response, except for the System area located at address 255.

The SRT512 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 SRT512 before a Select() command is issued are ignored.

**Figure 40. Read_block request format**

| SOF | Read_block | Address | $CRC_L$ | $CRC_H$ | EOF |
|---|---|---|---|---|---|
| | 08h | 8 blts | 8 bits | 8 bits | |

AI07684b

Request parameter:

- Address: block addresses from 0 to 15, or 255

**Figure 41. Read_block response format**

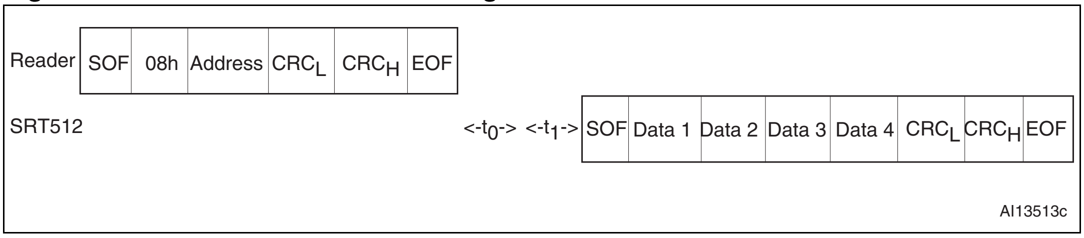
| SOF | Data 1 | Data 2 | Data 3 | Data 4 | $CRC_L$ | $CRC_H$ | EOF |
|---|---|---|---|---|---|---|---|
| | 8 blts | 8 blts | 8 blts | 8 blts | 8 bits | 8 blts | |

AI07685b

Response parameters:

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

**Figure 42. Read_block frame exchange between reader and SRT512**



## 8.8 Write_block (Addr, Data) command

Command code = 09h

On receiving the Write_block command, the SRT512 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 16 blocks of the SRT512 (addresses 0 to 15). Write_block commands issued with a block address above 15 will not be interpreted and the SRT512 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 paragraphs for a complete description of the Write_block command:

- *Figure 13: Lockable EEPROM area (addresses 0 to 4)*
- *Figure 14: Binary counter (addresses 5 to 6)*.
- *Figure 16: EEPROM (addresses 7 to 15)*.

The Write_block command does not give rise to a response from the SRT512. The reader must check after the programming time, $t_W$, that the data was correctly programmed. The SRT512 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 SRT512 before a Select() command is issued, are ignored.

**Figure 43. Write_block request format**

| SOF | Write_block | Address | Data 1 | Data 2 | Data 3 | Data 4 | $CRC_L$ | $CRC_H$ | EOF |
|---|---|---|---|---|---|---|---|---|---|
| | 09h | 8 blts | 8 blts | 8 blts | 8 blts | 8 blts | 8 bits | 8 blts | |

AI07687c

Request parameters:

- Address: block addresses from 0 to 15, or 255
- Data 1: Less significant data byte
- Data 2: Data byte
- Data 3: Data byte
- Data 4: Most significant data byte.

**Figure 44. Write_block response format**

| No response |
|---|
| |

AI07680b

**Figure 45. Write_block frame exchange between reader and SRT512**

| | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| Reader | SOF | 09h | Address | Data 1 | Data 2 | Data 3 | Data 4 | $CRC_L$ | $CRC_H$ | EOF |
| SRT512 | | | | | | | | | | No response |

AI13514d

## 8.9 Get_UID() command

Command code = 0Bh

On receiving the Get_UID command, the SRT512 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 SRT512 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 SRT512 before a Select() command is issued, are ignored.

**Figure 46. Get_UID request format**

| SOF | Get_UID | $CRC_L$ | $CRC_H$ | EOF |
|---|---|---|---|---|
| | 0Bh | 8 bits | 8 bits | |

AI07693b

Request parameter:

- No parameter

**Figure 47. 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 SRT512 family are uniquely identified by a 64-bit unique identifier (UID). This is used for addressing each SRT512 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 48*):

- 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 1100b = 12d for SRT512
- a 42-bit unique serial number

**Figure 48. 64-bit unique identifier of SRT512**

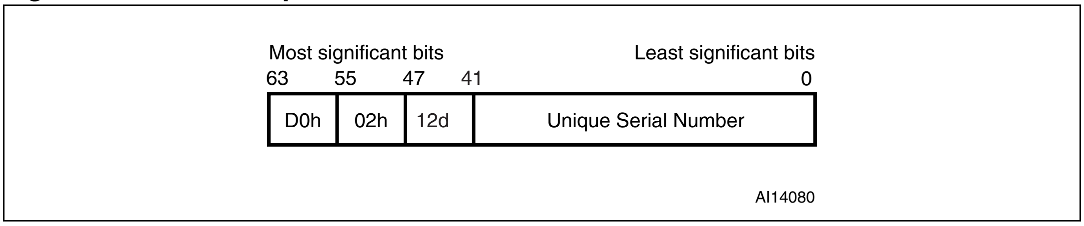
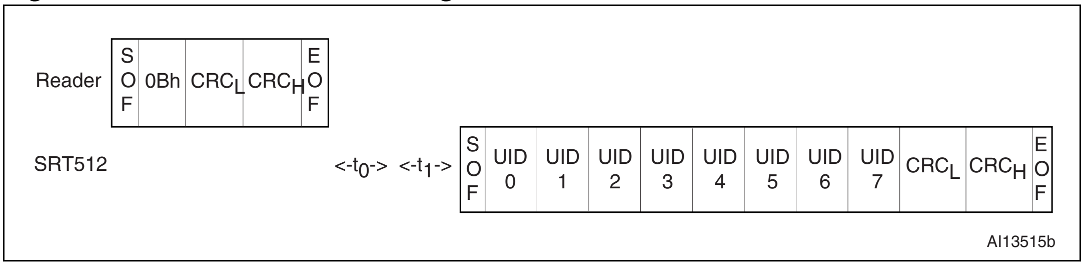



**Figure 49. Get_UID frame exchange between reader and SRT512**




## 8.10 Power-on state

After power-on, the SRT512 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 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**

<table>
<tr><th>Symbol</th><th colspan="2">Parameter</th><th>Min.</th><th>Max.</th><th>Unit</th></tr>
<tr><td>$T_{STG}$</td><td rowspan="2">Storage conditions</td><td rowspan="2">Wafer<br>(kept in its antistatic bag)</td><td>15</td><td>25</td><td>°C</td></tr>
<tr><td>$t_{STG}$</td><td>-</td><td>23</td><td>months</td></tr>
<tr><td>$I_{CC}$</td><td>Supply current on AC0 / AC1</td><td></td><td>–20</td><td>20</td><td>mA</td></tr>
<tr><td>$V_{MAX}$</td><td>Input voltage on AC0 / AC1</td><td></td><td>–7</td><td>7</td><td>V</td></tr>
<tr><td rowspan="2">$V_{ESD}$</td><td rowspan="2">Electrostatic discharge voltage</td><td>Machine model(1)</td><td>–100</td><td>100</td><td>V</td></tr>
<tr><td>Human body model(1)</td><td>–1000</td><td>1000</td><td>V</td></tr>
</table>

1. Mil. Std. 883 - Method 3015

# 10 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 V | - | - | 100 | µA |
| $I_{CC}$ | Supply current (active in write) | $V_{CC}$ = 3.0 V | - | - | 250 | µA |
| $V_{RET}$ | Backscattering-induced voltage | ISO10373-6 | 20 | - | - | mV |
| $C_{TUN}$ | Internal tuning capacitor | (1) | –3% | 64.0 | +3% | pF |

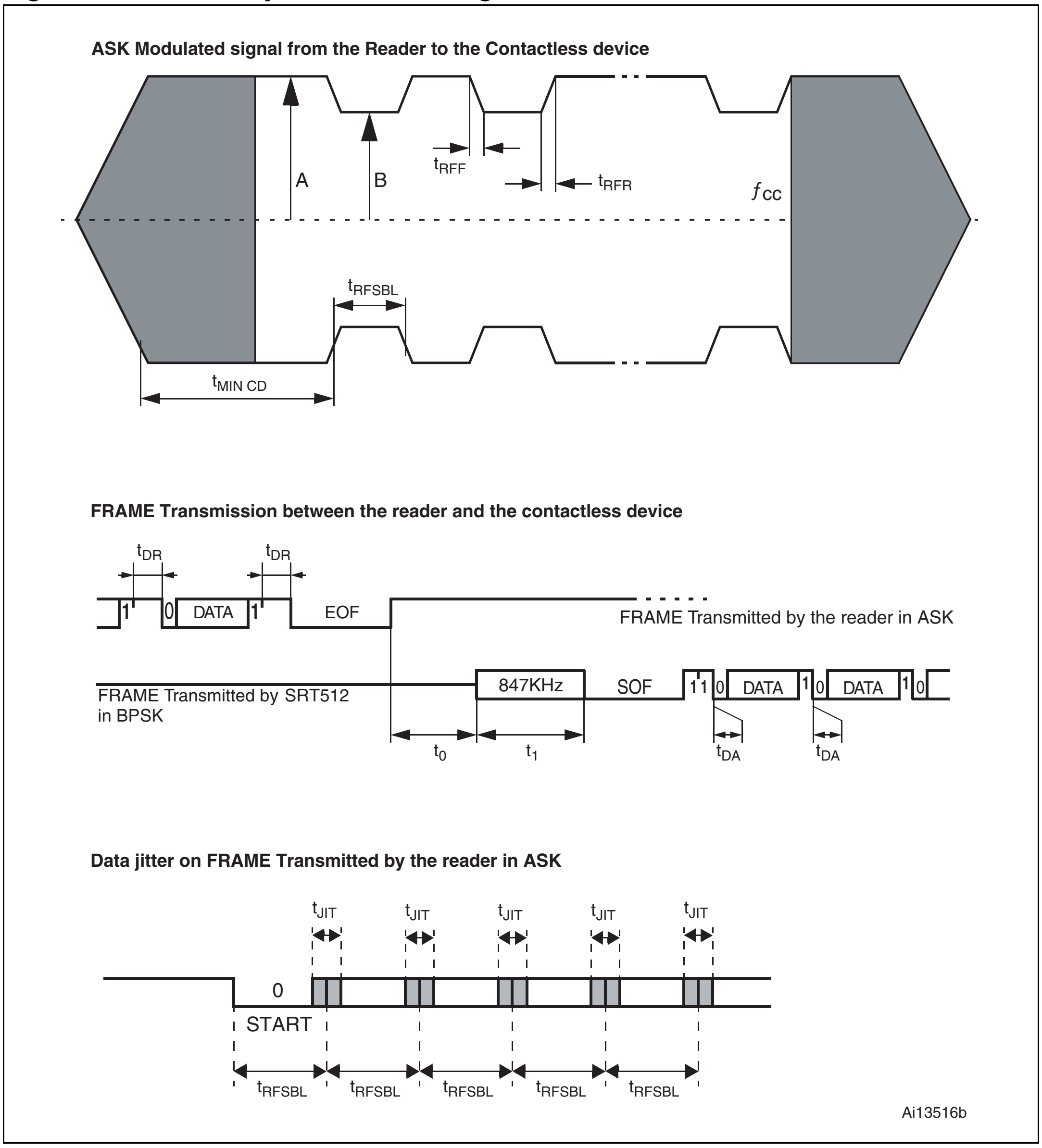
1. Based on automated production test equipment for which the typical value is 66.2 pF.

**Table 8. AC characteristics(1)**

| 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 | µs |
| $t_{RFSBL}$ | Minimum pulse width for start bit | ETU = 128/$f_{CC}$ | 9.44 | | µs |
| $t_{JIT}$ | ASK modulation data jitter | Coupler to SRT512 | –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 | 128/$f_S$ | 151 | | µs |
| $t_1$ | Synchronization delay | 128/$f_S$ | 151 | | µs |
| $t_2$ | Answer to new request delay | 14 ETU | 132 | | µs |
| $t_{DR}$ | Time between request characters | Coupler to SRT512 | 0 | 57 | µs |
| $t_{DA}$ | Time between answer characters | SRT512 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 | - | 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 µH
Tuning Frequency: 14.4 MHz.

### Figure 50. SRT512 synchronous timing, transmit and receive



# 11 Part numbering

**Table 9. Ordering information scheme**

| Example: | SRT512 | – | W4 | / | 1GE |
|---|---|---|---|---|---|

**Device type**

SRT512

**Package**

W4 =180 µm ± 15 µm unsawn wafer

SBN18 = 180 µm ± 15 µm bumped and sawn wafer on 8-inch frame

**Customer code**

1GE = generic product

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 ISO14443 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 SRT512 command brief

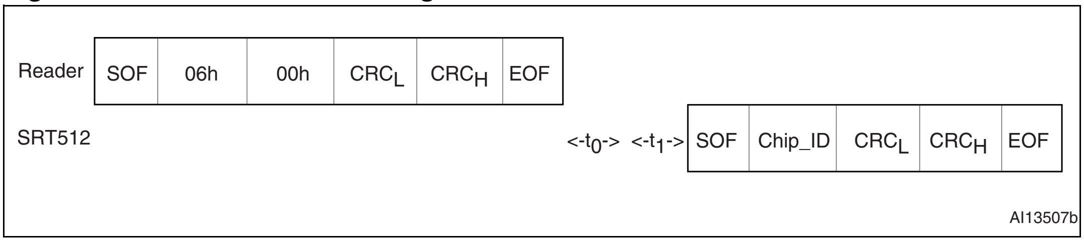



Figure 51. Initiate frame exchange between reader and SRT512

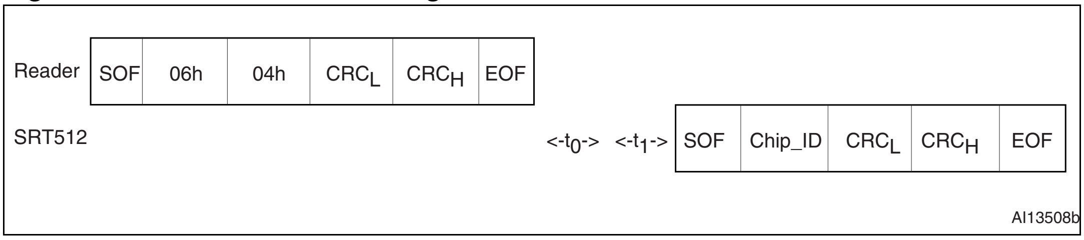



Figure 52. Pcall16 frame exchange between reader and SRT512

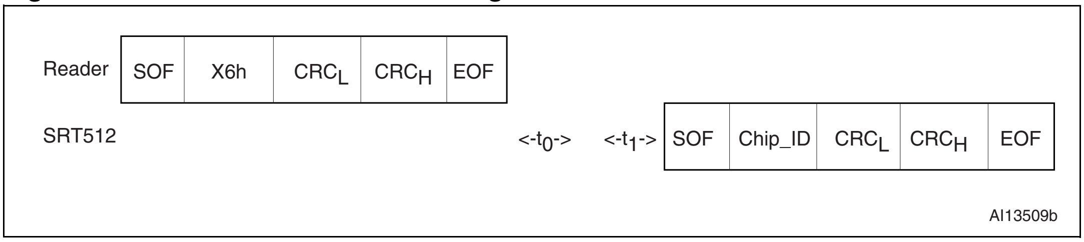



Figure 53. Slot_marker frame exchange between reader and SRT512

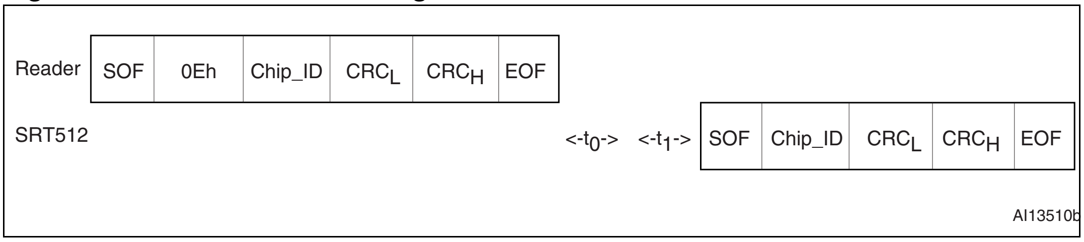



Figure 54. Select frame exchange between reader and SRT512

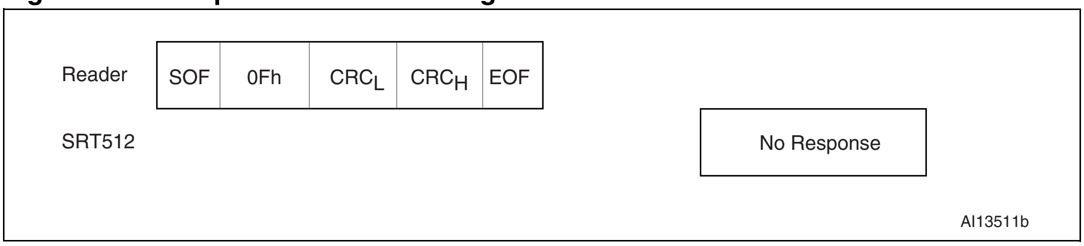
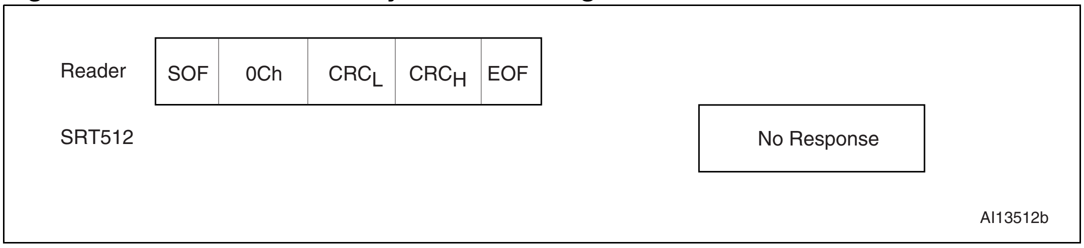
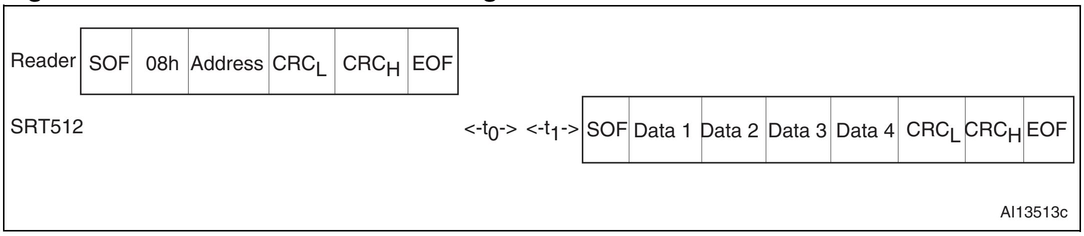



Figure 55. Completion frame exchange between reader and SRT512

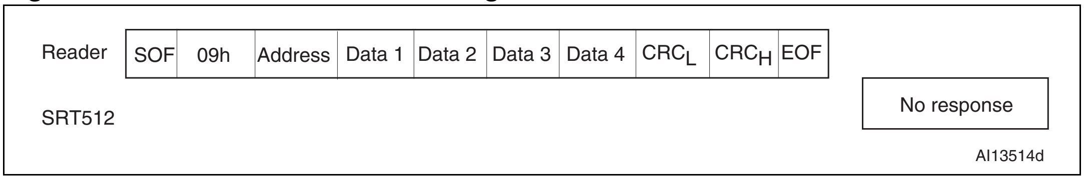
### Figure 56. Reset_to_inventory frame exchange between reader and SRT512



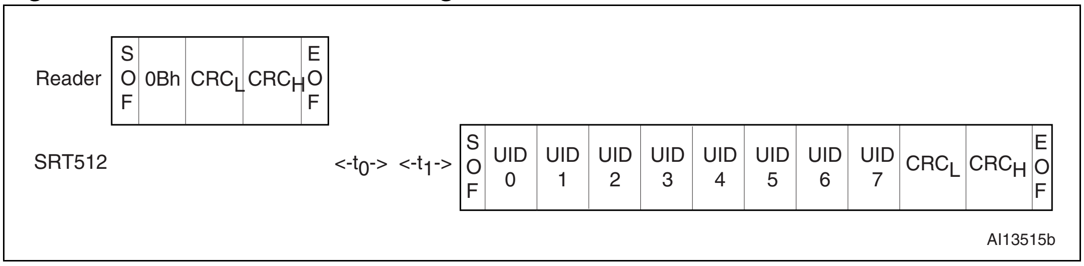

### Figure 57. Read_block frame exchange between reader and SRT512




### Figure 58. Write_block frame exchange between reader and SRT512




### Figure 59. Get_UID frame exchange between reader and SRT512



# Revision history

**Table 10. Document revision history**

| Date | Revision | Changes |
|---|---|---|
| 12-Dec-2006 | 0.1 | Initial release. |
| 22-Feb-2007 | 1 | Document status promoted from Target Specification to Preliminary Data. |
| 05-Apr-2007 | 2 | Document status promoted from Preliminary Data to full Datasheet.<br>A3, A4 and A5 antennas added (see *Package mechanical on page 41*).<br>6-bit IC code changed under *Unique identifier (UID) on page 37*.<br>$C_{TUN}$ min and max values removed, typical value added in *Table 7: DC characteristics*. Small text changes.<br>All antennas are ECOPACK® compliant. |
| 28-Aug-2008 | 3 | SRT512 products no longer delivered with A3, A4 and A5 antennas.<br>*Table 5: Absolute maximum ratings* and *Table 9: Ordering information scheme* clarified. Small text changes. |
| 28-Jul-2009 | 4 | Initial counter values corrected in *Section 4.2: 32-bit binary counters*. Small text changes. |
| 19-Sep-2011 | 5 | Updated *Section 1: Description*.<br>Modified disclaimer on last page. |
| 22-Oct-2012 | 5 | Improved internal tuning capacitor tolerance in *Table 7: DC characteristics*. |