



Chipsmall Limited consists of a professional team with an average of over 10 year of expertise in the distribution of electronic components. Based in Hongkong, we have already established firm and mutual-benefit business relationships with customers from,Europe,America and south Asia,supplying obsolete and hard-to-find components to meet their specific needs.

With the principle of "Quality Parts,Customers Priority,Honest Operation,and Considerate Service",our business mainly focus on the distribution of electronic components. Line cards we deal with include Microchip,ALPS,ROHM,Xilinx,Pulse,ON,Everlight and Freescale. Main products comprise IC,Modules,Potentiometer,IC Socket,Relay,Connector.Our parts cover such applications as commercial,industrial, and automotives areas.

We are looking forward to setting up business relationship with you and hope to provide you with the best service and solution. Let us make a better world for our industry!



## Contact us

Tel: +86-755-8981 8866 Fax: +86-755-8427 6832

Email & Skype: info@chipsmall.com Web: www.chipsmall.com

Address: A1208, Overseas Decoration Building, #122 Zhenhua RD., Futian, Shenzhen, China



**NOT RECOMMENDED  
FOR NEW DESIGNS**

Replaced by  
ATSHA204A

## Atmel CryptoAuthentication

### DATASHEET

### Features

- Secure authentication and validation device
- Integrated capability for both Host and Client operations
- Superior SHA-256 Hash algorithm with Message Authentication Code (MAC) and Hash-Based Message Authentication Code (HMAC) options
- Best-in-class, 256-bit key length; storage for up to 16 keys
- Guaranteed unique 72-bit serial number
- Internal, high-quality Random Number Generator (RNG)
- 4.5Kb EEPROM for keys and data
- 512 OTP (One Time Programmable) bits for fixed information
- Multiple I/O options
  - High-Speed, Single-Wire Interface
  - 1MHz I<sup>2</sup>C interface
- 2.0V to 5.5V supply voltage range
- 1.8V to 5.5V communications
- <150nA sleep current
- Extended, multi-level hardware security
- 8-lead SOIC, 8-lead TSSOP, 3-lead SOT23, 8-pad UDFN, and 3-lead Contact packages

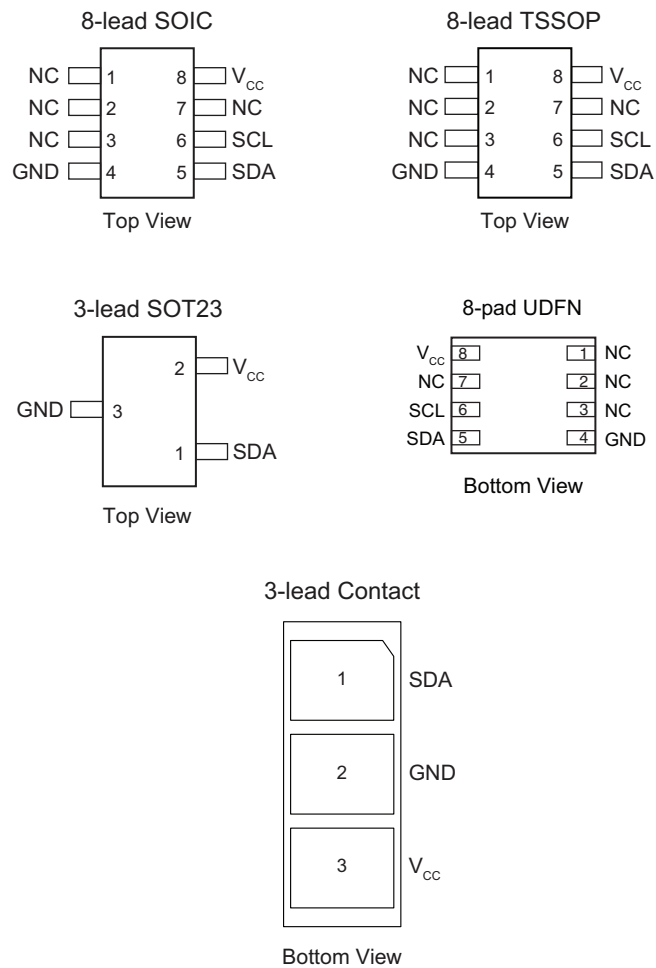
### Applications

- Anti-clone protection for accessories, daughter cards, and consumables
- Secure boot validation, software anti-piracy
- Network and computer access control
- Key exchange for encrypted downloads
- Authenticated/encrypted communications for control networks

**Table 1. Pin Configurations**

Pin Name	Function
NC	No Connect
GND	Ground
SDA	Serial Data
SCL	Serial Clock Input
V <sub>CC</sub>	Power Supply

**Figure 1. Pinouts**



## Table of Contents

---

<b>1. Introduction</b> . . . . .	<b>6</b>
1.1 Applications . . . . .	6
1.2 Device Features . . . . .	6
1.3 Cryptographic Operation . . . . .	7
<b>2. Device Organization</b> . . . . .	<b>8</b>
2.1 EEPROM Organization . . . . .	8
2.1.1 Data Zone . . . . .	9
2.1.2 Configuration Zone . . . . .	10
2.1.3 One Time Programmable (OTP) Zone . . . . .	14
2.1.4 Device Locking . . . . .	15
2.2 Static RAM (SRAM) . . . . .	17
2.2.1 TempKey . . . . .	17
<b>3. Security Features</b> . . . . .	<b>18</b>
3.1 Physical Security . . . . .	18
3.2 Random Number Generator (RNG) . . . . .	18
<b>4. General I/O Information</b> . . . . .	<b>19</b>
4.1 Byte and Bit Ordering . . . . .	19
4.1.1 Output Example . . . . .	19
4.1.2 MAC Message Example . . . . .	19
<b>5. Single-Wire Interface</b> . . . . .	<b>20</b>
5.1 I/O Tokens . . . . .	21
5.2 I/O Flags . . . . .	21
5.2.1 Transmit Flag . . . . .	21
5.3 Synchronization . . . . .	22
5.3.1 I/O Timeout . . . . .	22
5.3.2 Synchronization Procedures . . . . .	22
5.4 Sharing the Interface . . . . .	22
5.5 Transaction Example . . . . .	23
5.6 Wiring Configuration for Single-Wire Interface . . . . .	24
<b>6. I<sup>2</sup>C Interface</b> . . . . .	<b>25</b>
6.1 I/O Conditions . . . . .	25
6.1.1 Device is Asleep . . . . .	25
6.1.2 Device is Awake . . . . .	25
6.2 I <sup>2</sup> C Transmission to the ATSHA204 Device . . . . .	27
6.2.1 Word Address Values . . . . .	28
6.2.2 Command Completion Polling . . . . .	28
6.3 I <sup>2</sup> C Transmission from the ATSHA204 Device . . . . .	28
6.4 Address Counter . . . . .	29
6.5 I <sup>2</sup> C Synchronization . . . . .	29
6.6 Transaction Example . . . . .	30

<b>7. Electrical Characteristics</b> . . . . .	<b>31</b>
7.1 Absolute Maximum Ratings . . . . .	31
7.2 Reliability . . . . .	31
7.3 AC Parameters — All I/O Interfaces . . . . .	31
7.3.1 AC Parameters — Single-Wire Interface . . . . .	32
7.3.2 AC Parameters — I <sup>2</sup> C Interface . . . . .	34
7.4 DC Parameters — All I/O Interfaces . . . . .	35
7.4.1 V <sub>IH</sub> and V <sub>IL</sub> Specifications . . . . .	35
<b>8. Control Flags</b> . . . . .	<b>37</b>
8.1 I/O Blocks . . . . .	37
8.1.1 Status/Error Codes . . . . .	38
8.2 Sleep Sequence . . . . .	39
8.3 Idle Sequence . . . . .	39
8.4 Wake Sequence . . . . .	39
8.5 Watchdog Failsafe . . . . .	39
8.6 Command Sequence . . . . .	40
8.6.1 Command Packets . . . . .	40
8.6.2 Command Opcodes, Short Descriptions, and Execution Times . . . . .	41
8.6.3 Zone Encoding . . . . .	42
8.6.4 Address Encoding . . . . .	43
8.6.5 CheckMac Command . . . . .	44
8.6.6 DeriveKey Command . . . . .	46
8.6.7 DevRev Command . . . . .	48
8.6.8 GenDig Command . . . . .	49
8.6.9 HMAC Command . . . . .	51
8.6.10 Lock Command . . . . .	53
8.6.11 MAC Command . . . . .	54
8.6.12 Nonce Command . . . . .	56
8.6.13 Pause Command . . . . .	58
8.6.14 Random Command . . . . .	59
8.6.15 Read Command . . . . .	60
8.6.16 UpdateExtra Command . . . . .	62
8.6.17 Write Command . . . . .	63
<b>9. Compatibility</b> . . . . .	<b>65</b>
<b>10. Mechanical</b> . . . . .	<b>66</b>
10.1 Pinout . . . . .	66
<b>11. Package Drawings</b> . . . . .	<b>67</b>
11.1 8-lead SOIC . . . . .	67
11.2 8-lead TSSOP . . . . .	68
11.3 3-lead SOT23 . . . . .	69
11.4 8-pad UDFN . . . . .	70
11.5 3-lead Contact . . . . .	71
<b>12. Ordering Information</b> . . . . .	<b>72</b>
<b>13. Errata</b> . . . . .	<b>73</b>



14. Reference and Application Notes	74
14.1 SHA-256	74
14.2 HMAC/SHA-256	74
14.3 Key Values	74
14.3.1 Diversified Keys	75
14.3.2 Rolled Keys	75
14.3.3 Created Keys	75
14.3.4 Single-use Keys	75
14.3.5 Limited-use Key	76
14.3.6 Password Checking	77
14.3.7 Transport Keys	78
15. Revision History	79

# 1. Introduction

The following sections introduce the features and functions of the Atmel® ATSHA204 authentication device.

## 1.1 Applications

The ATSHA204 is a member of the Atmel CryptoAuthentication™ family of high-security hardware authentication devices. It has a flexible command set that allows use for many applications, including the following:

- **Anti-counterfeiting**  
Validate that a removable, replaceable, or consumable Client is authentic. Example Clients could be printer ink tanks, electronic daughter cards, or other spare parts. It can also be used to validate a software/firmware module or memory storage element.
- **Protection for Firmware or Media**  
Validate code stored in flash memory at boot to prevent unauthorized modifications (also known as secure boot), encrypt downloaded media files, and uniquely encrypt code images to be usable on a single system only.
- **Session Key Exchange**  
Securely and easily exchange stream encryption keys for use by an encryption/decryption engine in the system microprocessor to manage such things as a confidential communications channel or an encrypted download.
- **Secure Data Storage**  
Store secret keys for use by crypto accelerators in standard microprocessors. It can also be used to store small quantities of data necessary for configuration, calibration, ePurse value, consumption data, or other secrets. Programmable protection up through encrypted/authenticated reads and writes.
- **User Password Checking**  
Validate user entered passwords without letting the expected value become known, map simple passwords to complex ones, and securely exchange password values with remote system.

## 1.2 Device Features

The ATSHA204 device includes an Electrically Erasable Programmable Read-Only Memory (EEPROM) array that can be used for storage of keys, miscellaneous read/write, read-only or secret data, consumption logging, and security configuration. Access to the various sections of memory can be restricted in a variety of ways and the configuration then locked to prevent changes. See [Section 2.1, “EEPROM Organization”](#) for more details.

The ATSHA204 features a wide array of defensive mechanisms specifically designed to prevent physical attacks on the device itself or logical attacks on the data transmitted between the device and the system see [Section 3., “Security Features”](#) for more details. Hardware restrictions on the ways in which keys are used or generated provide further defense against certain styles of attack.

Access to the device is through a standard I<sup>2</sup>C interface at speeds up to 1Mb/s. see [Section 6., “I<sup>2</sup>C Interface”](#) for more details. It is compatible with the I<sup>2</sup>C interface specifications. The device also supports a Single-Wire Interface that can reduce the number of GPIOs required on the system processor and/or reduce the number of pins on connectors. See [Section 5., “Single-Wire Interface”](#) for more details.

Using the Single-Wire Interface, multiple ATSHA204 devices can share the same bus, which saves processor GPIO usage in systems with multiple Clients such as different color ink tanks or multiple spare parts. See [Section 5.4, “Sharing the Interface”](#) and [Section 8.6.13, “Pause Command”](#) for more details on the way in which this is implemented.

Each ATSHA204 ships with a guaranteed unique 9-byte (72-bit) serial number. Using the cryptographic protocols supported by the device, a Host system or remote server can prove that the serial number is both authentic and is not a copy. Serial numbers are often stored in a standard Serial EEPROM, but these can be easily copied, and there is no way for the Host to know if the serial number is authentic or a clone. The entire serial number must be utilized to guarantee uniqueness.

The ATSHA204 can generate high-quality random numbers and employ them for any purpose, including as part of the crypto protocols of this device. Because each 32-byte (256-bit) random number is not dependent on passed numbers generated on this or any other device, their inclusion in the protocol calculation ensures that replay attacks

(re-transmitting a previously successful transaction) always fail. See [Section 3.2, “Random Number Generator \(RNG\)”](#) and [Section 8.6.14, “Random Command”](#).

System integration is eased with a wide supply voltage range (2.0V through 5.5V) and an ultra-low sleep current of <150nA. Complete DC parameters are found in [Section 7., “Electrical Characteristics”](#), which describes multiple package options, including a tiny SOT23 package with a footprint of only 2.5mm x 3mm. See [Section 11., “Package Drawings”](#) for more details and for ordering codes.

See [Section 9., “Compatibility”](#) for information regarding compatibility with the Atmel AT88SA102S and the Atmel AT88SA10HS (previous members of the Atmel CryptoAuthentication family).

## 1.3 Cryptographic Operation

The ATSHA204 supports a standard challenge-response protocol to simplify programming. At its most basic, the Host system sends a challenge to the device in the Client, which combines that challenge with a secret key via the Message Authentication Code (MAC) command from the system, described in [Section 8.6.11, “MAC Command”](#) and sends the response back to the system. The device uses a cryptographic hash algorithm for the combination, which prevents an observer on the bus from deriving the value of the secret key, but allows the recipient to verify that the response is correct by performing the same calculation (combining the challenge with the secret) with a stored copy of the secret.

This basic operation can be expanded in many ways with the flexible command set of the ATSHA204. Using the GenDig command ([Section 8.6.8, “GenDig Command”](#)), the values in other slots can be included in the response digest which provides an effective way of proving that a data read really did come from the device, as opposed to being inserted by a man-in-the-middle attacker. This same command can be used to combine two keys with the challenge, which is useful when there are multiple layers of authentication to be performed.

The DeriveKey command ([Section 8.6.6, “DeriveKey Command”](#)) implements a key rolling scheme. Depending on the command mode parameter, the resulting operation can be similar to that implemented in a remote-controlled garage door opener. Each time the key is used, the current value of the key is cryptographically combined with a value specific to that system, and the result forms the key for the next cryptographic operation. Even if an attacker gets the value of one key, that key will be gone forever with the next use.

DeriveKey can also be used to generate new random keys that might be valid only for a particular Host ID, for a particular time period, or for some other restricted environment. Each generated key is different from any other key ever generated on any device. By “activating” a Host-Client pair in the field in this manner, a clone of a single Client would not work on any other Host.

In a Host-Client configuration where the Host (e.g. a mobile phone) needs to verify a Client (e.g. an OEM battery), there is a need to store the secret in the Host in order to validate the response from the Client. The CheckMac command ([Section 8.6.5, “CheckMac Command”](#)) allows the Host device to securely store the Client secret and hide the correct response value from the pins, returning only a yes/no answer to the system.

Where a user-entered password is a requirement, the CheckMac command also provides a way to both verify the password without exposing it on the communications bus as well as map the password to a stored value that can have much higher entropy. See [Section 14.3.6, “Password Checking”](#) for more details.

Finally, the hash combination of a challenge and secret key can be kept on the device and XORed with the contents of a slot to implement an encrypted read ([Section 8.6.15, “Read Command”](#)), or it can be XORed with encrypted input data to implement an encrypted write ([Section 8.6.17, “Write Command”](#)).

Each of these operations can be protected against replay attacks by including a random nonce ([Section 8.6.12, “Nonce Command”](#)) in the calculation.

All security functions are implemented using the industry-standard SHA-256 secure hash algorithm, which is part of the latest set of high-security cryptographic algorithms recommended by various governments and cryptographic experts. [Section 14.1, “SHA-256”](#) includes a reference to the algorithm details. If desired, the SHA-256 algorithm can also be included in an HMAC sequence (See [Section 8.6.9, “HMAC Command”](#)). The ATSHA204 employs full-sized, 256-bit secret keys to prevent any kind of exhaustive attack.



## 2. Device Organization

The device contains the following memory blocks:

- EEPROM
- SRAM

### 2.1 EEPROM Organization

The EEPROM contains a total of 664-bytes (5312-bits), and is divided into the following zones:

- **Data**  
A 512-byte (4Kb) zone split into 16 general-purpose, read-only, or read/write memory slots of 32 bytes (256 bits) each that can be used to store keys, calibration data, model number, or other information related to the item to which the ATSHA204 device is attached. Each slot may have different access restrictions based on the values stored in the Configuration zone. Within this document the nomenclature slot[yy] indicates the 32-byte value stored in slot yy of the Data zone.
- **Configuration**  
An 88-byte (704-bit) zone that contains serial number and other ID information as well as access permission information for each slot of the data memory. Within this document the nomenclature SN[a:b] indicates a range of bytes within a field of the configuration section. The 88-bytes are accessible from within a three-block address space.
- **OTP (One Time Programmable)**  
A 64-byte (512-bit) zone which can be used to store read-only data. Prior to locking the OTP zone, the bits may be freely written using the standard Write command. The OTP zone is accessible from within a two-block address space. Within this document the nomenclature OTP[bb] indicates a byte within the OTP zone, while OTP[aa:bb] indicates a range of bytes.

Within this document, the terms “slot” and “block” are used interchangeably to mean a single, 256-bit (32-byte) area of a particular memory zone. The industry SHA-256 documentation uses the term “block” to indicate a 512-bit section of the message input. In addition, the I/O section of this document uses the term “block” to indicate a variable-length aggregate element transferred between the system and the device.

In this specification, the nomenclature *mode:b* indicates bit b of the parameter mode.

On shipment from Atmel, the EEPROM contains factory test data that can be used for fixed-value board testing. This data must be overwritten with the desired contents prior to locking the configuration and/or data sections of the device. See the [Atmel website](#) for the document containing the specific shipment values.

## 2.1.1 Data Zone

This zone of 512-bytes (4Kb) is part of the EEPROM array, and can be used for secure storage purposes.

Prior to locking the configuration section (using lockConfig), the Data zone is inaccessible and can be neither read nor written. After configuration locking, the entire Data zone can be written using the Write command. If desired, the data to be written can be encrypted.

In [Table 2-1](#), “Byte Address” is the byte address within the Data zone for the first byte in the respective slot. Because all Reads and Writes with the ATSHA204 are performed on a word (4-byte or 32-byte) basis, the word address in the table below should be used for the address parameter passed to the Read and Write commands.

**Table 2-1. Data Zone Slots**

Slot	Byte Address (Hex)	Word Address (Hex)
0	0x0000	0x0000
1	0x0020	0x0008
2	0x0040	0x0010
3	0x0060	0x0018
4	0x0080	0x0020
5	0x00A0	0x0028
6	0x00C0	0x0030
7	0x00E0	0x0038

Slot	Byte Address (Hex)	Word Address (Hex)
8	0x0100	0x0040
9	0x0120	0x0048
10	0x0140	0x0050
11	0x0160	0x0058
12	0x0180	0x0060
13	0x01A0	0x0068
14	0x01C0	0x0070
15	0x01E0	0x0078

## 2.1.2 Configuration Zone

The 88-bytes (704-bits) in the Configuration zone contain manufacturing identification data, general device, and system configuration, and access restriction control values for the slots within the data zone. The values of these bytes can always be obtained using the Read command. The bytes of this zone are arranged as shown in [Table 2-2](#).

**Table 2-2. Configuration Zone**

Word	Byte 0	Byte 1	Byte 2	Byte 3	Default	Write Access	Read Access
0x00	Serial Number[0:3]				01 23 xx xx	Never	Always
0x01	Revision Number				xx xx xx xx	Never	Always
0x02	Serial Number[4:7]				xx xx xx xx	Never	Always
0x03	SN[8]	Reserved	I2C Enable	Reserved	EE 55 xx 00	Never	Always
0x04	I2C Address	Must be 0	OTP Mode	Selector Mode	C8 00 55 00	If Config Is unlocked	Always
0x05	Slot Configuration 0		Slot Configuration 1		8F 80 80 A1	If Config Is unlocked	Always
0x06	Slot Configuration 2		Slot Configuration 3		82 E0 A3 60	If Config Is unlocked	Always
0x07	Slot Configuration 4		Slot Configuration 5		94 40 A0 85	If Config Is unlocked	Always
0x08	Slot Configuration 6		Slot Configuration 7		86 40 87 07	If Config Is unlocked	Always
0x09	Slot Configuration 8		Slot Configuration 9		0F 00 89 F2	If Config Is unlocked	Always
0x0A	Slot Configuration 10		Slot Configuration 11		8A 7A 0B 8B	If Config Is unlocked	Always
0x0B	Slot Configuration 12		Slot Configuration 13		0C 4C DD 4D	If Config Is unlocked	Always
0x0C	Slot Configuration 14		Slot Configuration 15		C2 42 AF 8F	If Config Is unlocked	Always
0x0D	Use Flag 0	Update Count 0	Use Flag 1	Update Count 1	FF 00 FF 00	If Config Is unlocked	Always
0x0E	Use Flag 2	Update Count 2	Use Flag 3	Update Count 3	FF 00 FF 00	If Config Is unlocked	Always
0x0F	Use Flag 4	Update Count 4	Use Flag 5	Update Count 5	FF 00 FF 00	If Config Is unlocked	Always
0x10	Use Flag 6	Update Count 6	Use Flag 7	Update Count 7	FF 00 FF 00	If Config Is unlocked	Always
0x11	Last Key Use 0	Last Key Use 1	Last Key Use 2	Last Key Use 3	FF FF FF FF	If Config Is unlocked	Always
0x12	Last Key Use 4	Last Key Use 5	Last Key Use 6	Last Key Use 7	FF FF FF FF	If Config Is unlocked	Always
0x13	Last Key Use 8	Last Key Use 9	Last Key Use 10	Last Key Use 11	FF FF FF FF	If Config Is unlocked	Always
0x14	Last Key Use 12	Last Key Use 13	Last Key Use 14	Last Key Use 15	FF FF FF FF	If Config Is unlocked	Always
0x15	User Extra	Selector	Lock Data	Lock Config	00 00 55 55	Via Update Extra Command Only	Always

### 2.1.2.1 I2C Enable

Bit 0:

0 = Single-Wire Interface

1 = I<sup>2</sup>C interface

Bit 1–7: Ignored and set by Atmel.

### 2.1.2.2 I<sup>2</sup>C Address

I<sup>2</sup>C

Bit 0 = Ignored

Bits 1 – 7 = I<sup>2</sup>C device address

Single-Wire:

Bits 0 – 2 = Ignored

Bit 3 = TTL Enable

0 = Input level uses a fixed reference.

1 = Input level uses V<sub>CC</sub> reference

Bits 4 – 7 = Ignored

### 2.1.2.3 OTP Mode

0xAA (Read-only mode) = When OTP zone is locked, writes are disabled.

0x55 (Consumption mode) = Not supported at this time. Please contact Atmel for more information.

0x00 (Legacy mode) = When OTP zone is locked, writes are disabled, reads to Words 0 and 1, and 32-byte reads are disabled.

All other modes are reserved.

### 2.1.2.4 Selector Mode

0 = Selector can be updated with UpdateExtra.

All other values only allow Selector to be updated if its value is zero.

### 2.1.2.5 Slot Config

See [Table 2-3, SlotConfig Bits \(Per Slot\)](#).

### 2.1.2.6 Use Flag

For uses with “single-use slots”. The quantity of “1” bits represents the number of times slots 0 – 7 may be used before being disabled.

### 2.1.2.7 Update Count

Indicates how many times slots 0 – 7 have been updated with DeriveKey.

### 2.1.2.8 Last Key Use

Used to control limited use for Slot 15. Each “1” bit represents a remaining use for Slot 15.

### 2.1.2.9 UserExtra

Can be modified via the UpdateExtra command.

### 2.1.2.10 Selector

Selects which device will remain in active mode after the execution of the Pause command.

### 2.1.2.11 Lock Data

Controls the Data and OTP zone access.

0x55 = The Data and OTP zone is unlocked and has write access.

0x00 = The Data and OTP zone can only be updated with the Write command. Slots in the Data zone can only be modified based on the corresponding WriteConfig fields. The OTP zone can only be modified based on the OTP mode.

### 2.1.2.12 Lock Config

Configuration zone access.

0x55 = The Configuration zone has write access (unlocked).

0x00 = The Configuration zone does not have write access (locked).

### 2.1.2.13 SlotConfig (Bytes 20 – 51)

The 16 SlotConfig elements are used to configure the access protections for each of the 16 slots within the ATSHA204. Each configuration element consists of 16 bits, which control the usage and access for that particular slot/key. The SlotConfig field is interpreted according to [Table 2-3](#) when the Data zone is locked. When the Data zone is unlocked, these restrictions do not apply — all slots may be freely written and none may be read.

**Table 2-3. SlotConfig Bits (Per Slot)**

Bit	Name	Description
0 – 3	ReadKey	Slot of the key to be used for encrypted reads. If 0x0, then this slot can be used as the source slot for the CheckMac Copy Command.
4	CheckOnly	0 = This slot can be used for all crypto commands. 1 = This slot can only be used for CheckMac and GenDig followed by CheckMac Commands.
5	SingleUse	0 = No limit on the number of time the key can be used. 1 = Limit on the number of time the key can be used based on the UseFlag (or last key used) for the slot.
6	EncryptRead	0 = Clear reads are permitted. 1 = Requires the slot to be Secret and encrypted read to access.
7	IsSecret	0 = The slot is not secret and requires clear read, clear write, no MAC check, and no Derivekey Command. 1 = The slot is secret and requires encrypted reads and/or writes.
8 – 11	WriteKey	Slot of the key to be used to validate encrypted writes.
12 – 15	Write Config	See detailed function definition for use.



**Table 2-4. Write Configuration Bits — Derivekey Command**

Bit 15	Bit 14	Bit 13	Bit 12	Source Key <sup>(1)</sup>	Description
0	X	1	0	Target	DeriveKey command can be run without authorizing MAC (Roll).
1	X	1	0	Target	Authorizing MAC required for DeriveKey command (Roll).
0	X	1	1	Parent	DeriveKey command can be run without authorizing MAC (Create).
1	X	1	1	Parent	Authorizing MAC required for DeriveKey command (Create).
X	X	0	X	—	Slots with this value in the WriteConfig field may not be used as the target of the DeriveKey command.

Note: 1. The source key for the computation performed by the DeriveKey command can either be the key directly specified in Param2 (the “Target”) or the key at slotConfig[Param2].WriteKey (the “Parent”). See [Section 14.3, “Key Values”](#) for more details.

**Table 2-5. Write Configuration Bits — Write Command**

Bit 15	Bit 14	Bit 13	Mode Name	Description
0	0	0	Always	Clear text writes are always permitted on this slot. Slots set to “always” should never be used as key storage. Either 4 or 32 bytes may be written to this slot.
X	0	1	Never	Writes are never permitted on this slot using the Write command Slots set to “never” can still be used as key storage.
1	0	X	Never	Writes are never permitted on this slot using the Write command Slots set to “never” can still be used as key storage.
X	1	X	Encrypt	Writes to this slot require a properly computed MAC, and the input data must be encrypted by the system with WriteKey using the encryption algorithm documented in the Write command description ( <a href="#">Section 8.6.16, “UpdateExtra Command”</a> ). 4-byte writes to this slot are prohibited.

The 4-bit WriteConfig field is interpreted by the DeriveKey command as shown in [Table 2-4](#), where X means *don’t care*.

Note: The tables overlap. For example, a code of 0110 indicates a slot that can be written in encrypted form using the Write command and also can be the target of an unauthorized DeriveKey command with the target as the source.

The 4-bit WriteConfig field is interpreted by the Write command as shown in [Table 2-5](#), where X means *don’t care*.

Note: The tables overlaps with the DeriveKey Command. For example, a code of 0110 indicates a slot can be written in encrypted form using the Write command, and can be the target of an unauthorized DeriveKey command with the target as the source.

The IsSecret bit controls internal circuitry necessary for proper security for slots in which reads and/or writes must be encrypted or are prohibited altogether. It must also be set for all slots that are to be used as keys, including those created or modified with DeriveKey. Specifically, to enable proper device operation, this bit must be set unless WriteConfig is “Always”. 4-byte accesses are prohibited to/from slots in which this bit is set.

Slots used to store key values should always have IsSecret set to one and EncryptRead set to zero (reads prohibited) for maximum security. For fixed key values, WriteConfig should be set to “Never”. When configured in this way, there is no way to read or write the key after the Data zone is locked; it may only be used for crypto operations.

Some security policies require that secrets be updated from time to time. The ATSHA204 supports this capability in the following way: WriteConfig for the particular slot should be set to “Encrypt”, and SlotConfig.WriteKey should point back to the same slot by setting WriteKey to the slot ID. A standard Write command can be then used to write a new value to this slot provided that the authentication MAC is computed using the old (current) key value.

### 2.1.2.14 Special Memory Values in the Configuration Zone (Bytes 0 – 12)

Various fixed information is included in the ATSHA204 that can never be written under any circumstances and can always be read, regardless of the state of the lock bits.

- **SerialNum**

Nine bytes (SN[0:8]) which together form a unique value that is never repeated for any device in the CryptoAuthentication family. The serial number is divided into two groups:

1. **SN[0:1] and SN[8]**

The values of these bits are fixed at manufacturing time in most versions of the ATSHA204. Their default value is (0x01 0x23 0xEE). These 24 bits are always included in the SHA-256 computations made by the ATSHA204.

2. **SN[2:7]**

The values of these bits are programmed by Atmel during the manufacturing process and are different for every die. These 6-bytes (48-bits) are optionally included in some SHA-256 computations made by the ATSHA204.

- **RevNum**

4-bytes of information that are used by Atmel to provide manufacturing revision information. These bytes can be freely read as RevNum[0:3], but should never be used by system software, as they may vary from time to time.

### 2.1.3 One Time Programmable (OTP) Zone

The OTP zone of 64 bytes (512 bits) is part of the EEPROM array, and can be used for read-only storage.

Prior to locking the configuration section (using lockConfig), the OTP zone is inaccessible and can be neither read nor written. After configuration locking, but prior to locking of the OTP zone (using lockData), the entire OTP zone can be written using the Write command. If desired, the data to be written can be encrypted. When unlocked the OTP zone cannot be read.

Once the OTP zone is locked, the OTPmode byte in the Configuration zone controls the permissions of this zone, as follows:

- **Read-only Mode**

In this mode, the data cannot be modified, and would be used to store fixed model numbers, calibration information, manufacturing history, or other data that should never change. The Write command will always return an error and leave the memory unmodified.

All 64-bytes within the OTP section are always available for reading using either 4-byte or 32-byte reads.

- **Consumption Mode**

This mode is not currently supported. Contact Atmel for further information.

- **Legacy Mode**

In the Legacy mode, the operation of the OTP zone is consistent the fuse array on the Atmel ATSA102S. Reads of words zero and one are always prohibited, while reads of the remaining 14 words are always permitted. Only 4-byte (32-bit) reads are permitted, and any attempt to execute a 32-byte (256-bit) read will result in an error return code. All Write operations to the OTP zone are prohibited. See [Section 9.](#), “Compatibility” for more of the Atmel ATSA102S compatibility details.

All OTP zone bits have a value of one on shipment from the Atmel factory.

**Table 2-6. OTP Zone**

Word (HEX)	Address (HEX)	Default
0x00	0x00	0xFFFFFFFF
0x01	0x04	0xFFFFFFFF
0x02	0x08	0xFFFFFFFF
0x03	0x0C	0xFFFFFFFF
0x04	0x10	0xFFFFFFFF
0x05	0x14	0xFFFFFFFF
0x06	0x18	0xFFFFFFFF
0x07	0x1C	0xFFFFFFFF
0x08	0x20	0xFFFFFFFF
0x09	0x24	0xFFFFFFFF
0x0A	0x28	0xFFFFFFFF
0x0B	0x2C	0xFFFFFFFF
0x0C	0x30	0xFFFFFFFF
0x0D	0x34	0xFFFFFFFF
0x0E	0x38	0xFFFFFFFF
0x0F	0x3C	0xFFFFFFFF

#### 2.1.4 Device Locking

There are two separate lock bytes for the device:

- Lock the Configuration zone (controlled by LockConfig, byte 87).
- Lock both the Data and OTP zones (controlled by LockData, byte 86).

These locks are stored within separate bytes in the Configuration zone, and can be modified only through the Lock command. After a memory zone is locked, there is no way to unlock it.

The device should be personalized at the system manufacturer with the desired configuration information, and the Configuration zone should be locked. When this lock is complete, all necessary writes of public and secret information into the EEPROM slots should be performed using encrypted writes if appropriate. Upon completion of writes to the data and OTP zones, the Data and OTP zones should be locked.

It is vital that the Data and OTP zones be locked prior to release of the system containing the device into the field. Failure to lock these zones may permit modification of any secret keys and may lead to other security problems.

Any attempt to read or write the Data or OTP sections prior to locking the configuration section causes the device to return an error.

Contact Atmel for optional secure personalization services.

### 2.1.4.1 Configuration Zone Locking

Certain bytes within the Configuration zone cannot be modified, regardless of the state of LockConfig. Access to the remainder of the bytes within the zone is controlled using the LockConfig byte in the Configuration zone, as shown in [Table 2-7](#). Throughout this document, if LockConfig is 0x55, the Configuration zone is said to be unlocked; otherwise it is locked.

**Table 2-7. Configuration Zone Locking**

	Read Access	Write Access
LockConfig == 0x55 (unlocked)	Read	Write
LockConfig != 0x55 (locked)	Read	<never>

### 2.1.4.2 Data and OTP Zone Locking

Throughout this document, if LockData is 0x55, then both the Data and OTP zones are said to be unlocked; otherwise they are locked.

Note: There is neither read nor write access to the Data and OTP zones prior to locking of the Configuration zone.

**Table 2-8. Data and OTP Zone Access Restrictions**

	Read Access	Write Access
LockData == 0x55 (unlocked)	<never>	Write
LockData != 0x55 (locked)	Read <sup>(1)</sup>	Write <sup>(1)</sup>

Note: 1. Based on Slot Configuration for a given slot.

### 2.1.4.3 OTP Zone Locking

Reads and writes of the OTP zone depend on the state of the LockConfig, LockData, and OTP mode bytes in the Configuration zone.

## 2.2 Static RAM (SRAM)

The device includes an SRAM Array that is used to store the input command or output result, intermediate computation values, and/or an ephemeral key. The entire contents of this memory are always invalidated whenever the device goes into sleep mode or the power is removed. The ephemeral key is named TempKey, and can be used as an input to the MAC, HMAC, CheckMac, GenDig, and DeriveKey commands. It is also used as the Data protection (encryption or decryption) key by the Read and Write commands. See the below, [Section 2.2.1, “TempKey”](#).

### 2.2.1 TempKey

TempKey is a storage register in the SRAM array that can be used to store an ephemeral result value from the Nonce or GenDig commands. The contents of this register can never be read from the device (although the device itself can read and use the contents internally).

This register contains the elements shown in [Table 2-9](#).

**Table 2-9. TempKey Storage Register**

Name	Bit Length	Description
TempKey	256 (32-bytes)	Nonce (from Nonce command) or Digest (from GenDig command).
SlotID	4	If TempKey was generated by GenDig (see the GenData and CheckFlag bits), these bits indicate which key was used in its computation. The four bits represent one of the slots of the Data zone.
SourceFlag	1	The source of the randomness in TempKey: 0 = Internally generated random number ( <i>Rand</i> ). 1 = Input seed only, no internal random generation ( <i>Input</i> ).
GenData	1	0 = TempKey.SlotID is not meaningful, and is ignored. 1 = The contents of TempKey were generated by GenDig using one of the slots in the Data zone (and TempKey.SlotID will be meaningful).
CheckFlag	1	If one, the contents of TempKey were generated by the GenDig command and at least one of the keys used in that generation is restricted to the CheckMac command (SlotConfig.CheckOnly is 1); otherwise, this bit will be zero.
Valid	1	0 = The information in TempKey is invalid. 1 = The information in TempKey is valid.

In this specification, the name “TempKey” refers to the contents of the 32-byte (256-bit) Data register. The remaining bit fields are referred to as TempKey.SourceFlag, TempKey.GenData, and so on.

The TempKey.Valid bit is cleared to zero under any of the following circumstances:

- Power-up, sleep, brown-out, watchdog expiration, or tamper detection. The contents of TempKey are however retained when the device enters idle mode.
- After the execution of any command other than Nonce or GenDig, regardless of whether or not the command execution succeeds. It may be cleared by the CheckMac command unless a successful copy takes place. It is *not* cleared if there is a communications problem, as evidenced by a cyclic redundancy check (CRC) error.
- An error during the parsing or execution of GenDig and/or Nonce.
- Execution of GenDig replaces any previous output of the Nonce command with the output of the GenDig command. Execution of the Nonce command likewise replaces any previous output of the GenDig command.



## 3. Security Features

### 3.1 Physical Security

The ATSHA204 incorporates a number of physical security features designed to protect the EEPROM contents from unauthorized exposure. The security measures include:

- An Active Shield over the part
- Internal Memory Encryption
- Secure Test Modes
- Glitch Protection
- Voltage Tamper Detection

Pre-programmed transport keys stored on the ATSHA204 are encrypted in such a way as to make retrieval of their values using outside analysis very difficult.

Both the logic clock and logic supply voltage are internally generated, preventing any direct attack on these two signals using the pins of the device.

### 3.2 Random Number Generator (RNG)

The ATSHA204 includes a high-quality random number generator that returns a 32-byte random number to the system. The device combines this generated number with a separate input number to form a nonce that is stored within the device in TempKey and may be used by subsequent commands.

The system may use this random number generator for any purpose. One common purpose would be as the input challenge to the MAC command on a separate CryptoAuthentication device. The device provides a special Random command for such purposes, which does not affect the internally stored nonce.

To simplify system testing, prior to locking the Configuration zone the random number generator always returns the following value:

```
ff ff 00 00 ff ff 00 00 ...
```

where ff is the first byte read from the device and is used for the SHA message.

To prevent replay attacks on encrypted data that is passed to or from the ATSHA204, the device requires that a new, internally generated nonce be included as part of the encryption sequence used to protect the data being read or written. To implement this requirement, the data protection key generated by GenDig and used by the Read or Write command must use the internal random number generator during the creation of the nonce.

Random numbers are generated from a combination of the output of a hardware random number generator and an internal seed value, which is not externally accessible. The internal seed is stored in the EEPROM, and is normally updated once after every power-up or sleep/wake cycle. After the update, this seed value is retained in registers within the device that are invalidated if the device enters sleep mode or the power is removed.

Because there is an EEPROM endurance specification that limits the number of times the EEPROM seed can be updated, the Host system should manage power cycles to minimize the number of required updates. In certain circumstances, the system may choose to suppress the EEPROM seed update using the mode parameter to the Nonce and Random commands. Because this may affect the security of the system, it should be used with caution. See [Section 8.6.12, “Nonce Command”](#) and [Section 8.6.14, “Random Command”](#) for more information about how the EEPROM seed update is controlled.

## 4. General I/O Information

Communication with the ATSHA204 is achieved through one of two different protocols (I<sup>2</sup>C or Single-Wire) and is selected based on the device ordered:

- **Single-Wire Interface**  
This mode uses a single GPIO connection on the system microprocessor connected to the SDA pin on the device. It permits the fewest number of connector pins to any removable/replaceable entity. The bit rate is up to 26Kb/s and is compatible with standard UART signaling.
- **I<sup>2</sup>C Interface**  
This mode is compatible with the Atmel AT24C16B Serial EEPROM interface. Two pins are required, Serial Data (SDA) and Serial Clock (SCL). The I<sup>2</sup>C interface supports a bit rate of up to 1Mb/s.

The lowest levels of the I/O protocols are described in [Section 5., “Single-Wire Interface”](#) and [Section 6., “I<sup>2</sup>C Interface”](#). On top of the I/O protocol level, both interfaces transmit exactly the same bytes to and from the device to implement the cryptographic commands and error codes documented in [Section 8., “Control Flags”](#).

Note: The device implements a failsafe internal watchdog timer that forces it into a very low-power mode after a certain time interval, regardless of any current activity. System programming must take this into consideration. See [Section 8.5, “Watchdog Failsafe”](#) for more details.

### 4.1 Byte and Bit Ordering

CryptoAuthentication uses a common ordering scheme for bytes and also for the way in which numbers and arrays are represented in this datasheet:

- All multi-byte aggregate elements are treated as arrays of bytes and are processed in the order received or transmitted with index #0 first.
- 2-byte (16-bit) integers, typically Param2 appear on the bus least-significant byte first.

The bit order is different depending on the I/O channel used:

- On the Single-Wire Interface, data is transferred to/from the ATSHA204 least-significant bit first on the bus.
- On the I<sup>2</sup>C interface, data is transferred to/from the ATSHA204 most-significant bit first on the bus.

#### 4.1.1 Output Example

The following bytes will be returned in this order on the bus by a 32-byte read of the configuration section with an input address of 0x0000:

SN[0], SN[1], SN[2], SN[3], RevNum[0], RevNum[1], RevNum[2], RevNum[3], SN[4], SN[5], SN[6], SN[7], SN[8], reserved, I2C Enable, reserved, I2C\_Address, TempOffset, OTPmode, SelectorMode, SlotConfig[0].Read, SlotConfig[0].Write, SlotConfig[1].Read, SlotConfig[1].Write, SlotConfig[2].Read, SlotConfig[2].Write, SlotConfig[3].Read, SlotConfig[3].Write, SlotConfig[4].Read, SlotConfig[4].Write, SlotConfig[5].Read, SlotConfig[5].Write

#### 4.1.2 MAC Message Example

The following bytes will be passed to the SHA engine for a MAC command using a mode value of 0x71 and a SlotID of slot x. In the example below, K[x] indicates the SlotID of slot x in the Data zone, with K[0] being the first byte on the bus for a read from or write to that slot. OTP[0] indicates the first byte on the bus for a read of the OTP zone at address zero, and so on.

K[0], K[1], K[2], K[3] ... K[31], TempKey[0], TempKey[1], TempKey[2], TempKey[3] ... TempKey[31], Opcode (=0x08), Mode (=0x71), Param2(LSB = x), Param2(MSB = 0), OTP[0], OTP[1], OTP[2], OTP[3], OTP[4], OTP[5], OTP[6], OTP[7], OTP[8], OTP[9], OTP[10], SN[8], SN[4], SN[5], SN[6], SN[7], SN[0], SN[1], SN[2], SN[3].

For more details regarding MAC messages, see [Section 8.6.11, “MAC Command”](#).

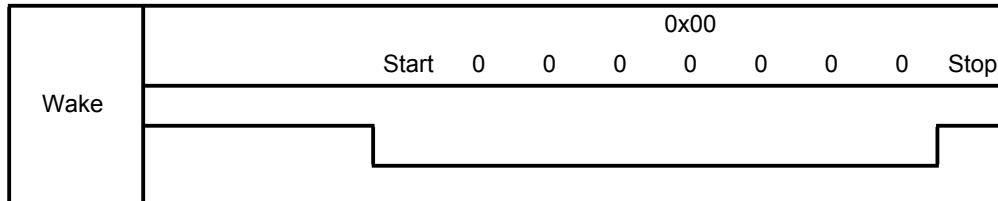
## 5. Single-Wire Interface

In the Single-Wire Interface mode, communications to and from the ATSHA204 take place over the SDA pin, a single, asynchronously timed wire, and the SCL pin is ignored.

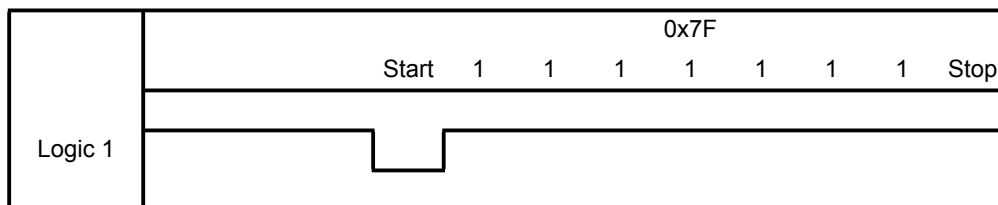
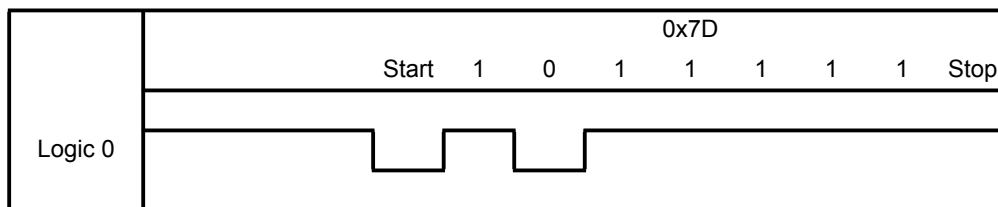
Note: The sleep current specification values are guaranteed only if the SCL pin is held low or left unconnected.

The overall communications structure is a hierarchy:

### Wake Tokens



**I/O Tokens** I/O tokens implement a single data bit transmitted on the bus, or the wake-up event. The figure below shows how to use the Single-Wire Interface with a standard RS-232 port. The port should be set to 7-bit addressing.



**I/O Flags** Flags consist of eight tokens (bits) that convey the direction and meaning of the next group of bits (if any) that may be transmitted. Flags are always transmitted LSB first.

**Blocks** Blocks of data follow the command and transmit flags. They incorporate both a byte count and a checksum to ensure proper data transmission.

**Packets** Packets of bytes form the core of the block (minus the byte count and CRC). They are either the input or output parameters of a CryptoAuthentication command or status information from the ATSHA204.

## 5.1 I/O Tokens

There are a number of I/O tokens that may be transmitted over the Single-Wire Interface:

**Input:** (to the ATSHA204)

- Wake** Wake the device up from either sleep or idle states.
- Zero** Send a single bit from the system to the device with a value of zero.
- One** Send a single bit from the system to the device with a value of one.

**Output:** (from the ATSHA204)

- ZeroOut** Send a single bit from the device to the system with a value of zero.
- OneOut** Send a single bit from the device to the system with a value of one.

The waveforms are the same in either direction. There are some differences in timing; however, based on the expectation that the Host has a very accurate and consistent clock, while the ATSHA204 has significant part-to-part variability in its internal clock generator, due to normal manufacturing and environmental fluctuations.

The bit timing is designed to permit a standard UART running at 230.4Kbaud to transmit and receive the tokens efficiently. Each byte transmitted or received by the UART corresponds to a single bit received or transmitted by the device. The UART needs to be configured with 7-bits of data having 0x7F corresponding to a Logic 1 and 0x7D corresponding to a Logic 0.

The Wake token is special in that it requires an extra long low pulse of 8 $\mu$ s on the SDA pin (see [Table 7-3](#)), which cannot be confused with the shorter low pulses that occur during a Data token (Zero, One, ZeroOut, OneOut). Devices that are in either the idle or sleep state ignore all data tokens until they receive a legal Wake token. Do not send a Wake token to devices that are awake, as they will lose synchronization because the waveform can be resolved to neither a legal one nor zero. See [Section 5.3.2, “Synchronization Procedures”](#) for the procedure to regain synchronization.

## 5.2 I/O Flags

The system is always the bus master; so before any I/O transaction, the system must send an 8-bit flag to the device to indicate the I/O operation to be subsequently performed, as shown in [Table 5-1](#).

**Table 5-1. I/O Flags**

Value	Name	Meaning
0x88	Transmit	Communicates to the device to wait for a bus turnaround time and then start transmitting its response to the previously transmitted command block.

### 5.2.1 Transmit Flag

The transmit flag is used to turn the bus around so that the ATSHA204 can send data back to the system. The bytes that the device returns to the system depend on the current state of the device, and may include either status, error code, or command results.

When the device is busy executing a command, it ignores the SDA pin and any flags sent by the system. See [Section 8.6.2, “Command Opcodes, Short Descriptions, and Execution Times”](#) for execution delays in the device for each command type. The system must observe these delays before trying to communicate with the device after sending a command.

## 5.3 Synchronization

Because the communications protocol is half-duplex, there is the possibility that the system and the ATSHA204 will fall out of synchronization with each other. In order to speed recovery, the device implements a timeout that forces it to sleep under certain circumstances.

### 5.3.1 I/O Timeout

After a leading transition for any data token has been received, the ATSHA204 will expect the remaining bits of the token to be properly received by the device within the  $t_{\text{TIMEOUT}}$  interval. Failure to send enough bits or the transmission of an illegal token (a low pulse exceeding  $t_{\text{ZLO}}$ ) will cause the device to enter the sleep state after the  $t_{\text{TIMEOUT}}$  interval.

The same timeout applies during the transmission of the command block. After the transmission of a legal command flag, the I/O timeout circuitry is enabled until the last expected data bit is received.

Note: The timeout counter is reset after every legal token, and the total time to transmit the command may exceed the  $t_{\text{TIMEOUT}}$  interval while the time between bits may not.

The I/O timeout circuitry is disabled when the device is busy executing a command.

### 5.3.2 Synchronization Procedures

If the device is not busy when the system sends a transmit flag, the device should respond within  $t_{\text{TURNAROUND}}$ . If  $t_{\text{EXEC}}$  time has not already passed, the device may be busy, and the system should poll or wait until the maximum  $t_{\text{EXEC}}$  time has elapsed. If the device still does not respond to a second transmit flag within  $t_{\text{TURNAROUND}}$ , it may be out of synchronization. At this point, the system may take the following steps to reestablish communication:

1. Wait  $t_{\text{TIMEOUT}}$ .
2. Send the transmit flag.
3. If the device responds within  $t_{\text{TURNAROUND}}$ , then the system may proceed with more commands.
4. Send a Wake token.
5. Wait  $t_{\text{WHI}}$ .
6. Send the transmit flag.
7. The device should respond with a 0x11 status within  $t_{\text{TURNAROUND}}$ , at which time system may proceed with commands.

Any command results in the I/O buffer may be lost when the system and device lose synchronization.

## 5.4 Sharing the Interface

Multiple CryptoAuthentication devices may share the same interface, as follows:

1. System issues a Wake token ([Section 8.4, “Wake Sequence”](#)) to wake-up all devices.
2. The system issues the Pause command to put all but one of the devices into idle mode. Only the remaining device then sees any commands the system sends. When the system has completed talking to the one active device, it sends an idle flag, which the idle devices ignore but puts the single remaining active device into the idle mode. See [Section 8.6.13, “Pause Command”](#) for more details.

Steps 1 and 2 are repeated for each device on the wire. If the system has completed communications with the final device, it should wake all the devices up and then put all the devices to sleep to reduce total power consumption.

The device uses the selector byte within the configuration zone to determine which device stays awake – only that device with a selector value that matches the input parameter of the Pause command stays awake. In order to facilitate late configuration of systems that use the multi-device sharing mode, the following three update capabilities for the selector byte are supported:



- Unlimited Updates**  
At any time, the UpdateExtra command can be executed to write the value in the selector field of the Configuration zone. To enable this mode, set the SelectorMode byte in the Configuration zone to zero.
- One-time Field Update**  
If the SelectorMode byte is set to a non-zero value and the selector byte is set to a zero value prior to locking the Configuration zone. Then, at any time after the Configuration zone is locked the UpdateExtra command can be used one time to set Selector to a non-zero value. The UpdateExtra command is not affected by the LockData byte.
- Fixed Selector Value**  
The selector byte can never be modified after the Configuration zone is locked if both SelectorMode and Selector are set to non-zero values. The UpdateExtra command will always return an error code.

## 5.5 Transaction Example

Figure 5-1. Wake (Single-Wire)

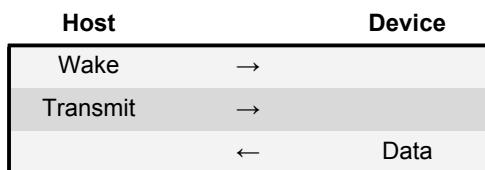


Figure 5-2. Example (Single-Wire)

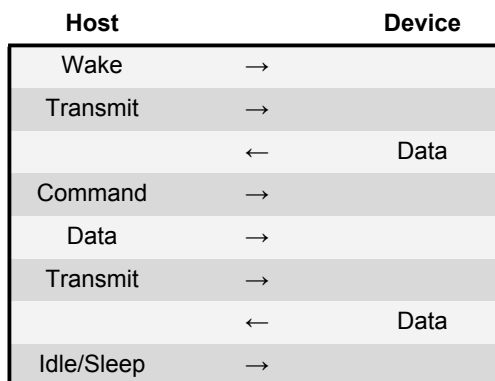
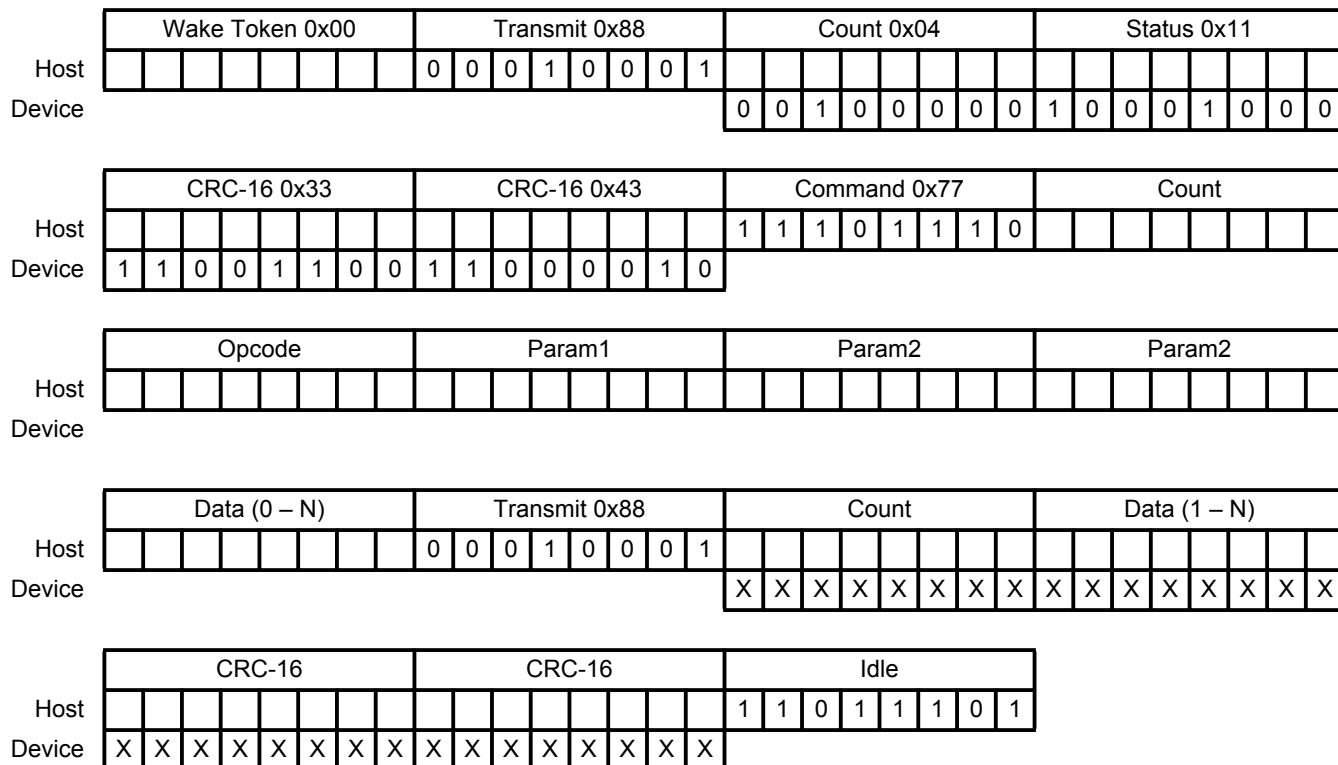


Figure 5-3. Example (Single-Wire)



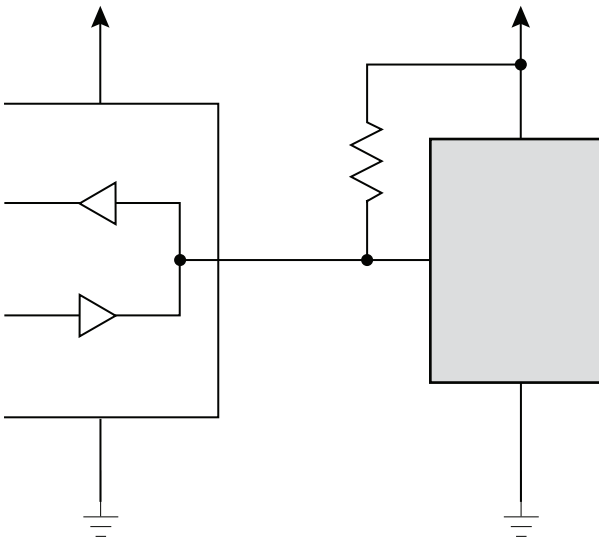
## 5.6 Wiring Configuration for Single-Wire Interface

Using the Single-Wire Interface allows the connection of the ATSHA204 to a Host using only a single SDA pin to transfer data in both directions. This interface does not use the SCL pin. In this configuration, no bypass capacitor is required to connect the device to the system (bypass capacitor is recommended).

To prevent forward biasing the internal diode and drawing current across power planes in the system, the resistor pull-up on the SDA pin should either be connected to the same supply that is connected to the  $V_{CC}$  pin or to a lower voltage rail.

If the signal levels for SDA are different from the  $V_{CC}$  voltage, consult the parametric specifications section of this document to ensure that the signal levels are such that excessive leakage current will be minimized when in sleep modes. This situation might occur if the ATSHA204 device is physically distant from the bus master device, or the supply voltage for the bus master is different from the supply voltage for the ATSHA204.

**Figure 5-4. 3-wire Configuration for Single-Wire Interface**



## 6. I<sup>2</sup>C Interface

The I<sup>2</sup>C interface uses the SDA and SCL pins to indicate various I/O states to the ATSHA204. This interface is designed to be compatible at the protocol level with other I<sup>2</sup>C devices operating up to 1MHz.

The SDA pin is normally pulled high with an external pull-up resistor, as the ATSHA204 includes only an open-drain driver on its output pin. The bus master may be either open-drain or totem pole, and if the latter, then it should be tri-stated when the ATSHA204 is driving results on the bus. The SCL pin is an input, and must be driven both high and low at all times by an external device or pull-up.

### 6.1 I/O Conditions

The device responds to the following I/O conditions:

#### 6.1.1 Device is Asleep

When the device is asleep, it ignores all but the wake condition

**Wake:** If SDA is held low for a period greater than  $t_{WLO}$ , the device exits low-power mode and, after a delay of  $t_{WHI}$ , is ready to receive I<sup>2</sup>C commands. The device ignores any levels or transitions on the SCL pin when the device is idle or asleep and during  $t_{WLO}$ . At some point during  $t_{WHI}$ , the SCL pin is enabled and the conditions listed in Section 6.1.2, “Device is Awake”, are honored.

The Wake condition requires either that the system processor manually drive the SDA pin low for  $t_{WLO}$ , or that a data byte of 0x00 is transmitted at a clock rate sufficiently slow that SDA is low for a minimum period of  $t_{WLO}$ . When the device is awake, the normal processor I<sup>2</sup>C hardware and/or software can be used for device communications up to and including the I/O sequence required to put the device back into low-power (sleep) mode.

When there are multiple ATSHA204 devices on the bus and the I<sup>2</sup>C interface is run at 133KHz or slower, the transmission of certain data patterns (such as 0x00) will cause all the ATSHA204 devices on the bus to wake-up. Because subsequent device addresses transmitted along the bus will only match the desired devices, the unused devices will remain inactive and not cause any bus conflicts.

In I<sup>2</sup>C mode, the device will ignore a wake sequence sent when the device is already awake.

#### 6.1.2 Device is Awake

When the device is awake, it honors the conditions listed below.

**Data Zero:** If SDA is low and stable while SCL goes from low to high to low, then a zero bit is being transferred on the bus. SDA can change while SCL is low.

**Data One:** If SDA is high and stable while SCL goes from low to high to low, then a one bit is being transferred on the bus. SDA can change while SCL is low.

Figure 6-1. Data Bit Transfer on I<sup>2</sup>C Interface

