Atmel ATSHA204

Atmel CryptoAuthentication Chip

DATASHEET

Features

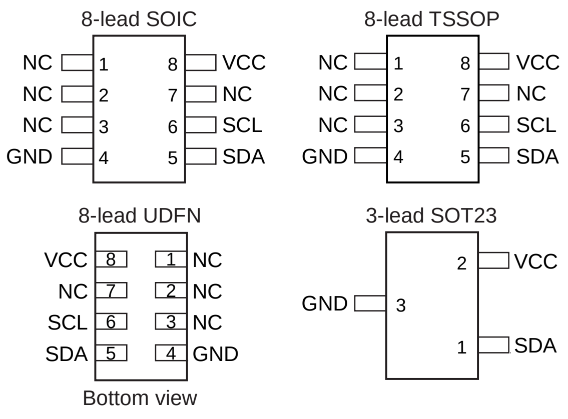
- Secure authentication and validation device
- Integrated capability for both host and client operations
- Superior SHA-256 hash algorithm, HMAC option
- Best-in-class, 256-bit key length; storage for up to 16 keys
- Guaranteed unique 72-bit serial number
- Internal, high-quality random number generator
- 4.5-Kbit EEPROM for keys and data
- 512 OTP bits for fixed information
- Multiple I/O options
    - High-speed, single-pin interface
    - 1MHz I^{2}C interface
- 2.0V – 5.5V supply voltage range
- 1.8V – 5.5V communications
- <150nA sleep current
- Extended, multi-level hardware security
- 8-lead SOIC, 8-lead TSSOP, 3-lead SOT23, and 8-pad UDFN 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

Figure 1. Pin Configurations

Pin Name	Function
SDA	Serial data
SCL	Serial clock input
GND	Ground
VCC	Power supply



1. Introduction

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

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 (aka 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. 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 Chip Features

The ATSHA204 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 on the EEPROM organization.

The ATSHA204 features a wide array of defensive mechanisms specifically designed to prevent physical attacks on the chip itself or logical attacks on the data transmitted between the chip and the system. See Section 3.4, “Security Features,” for more details. Hardware restrictions on the ways in which keys are used or generated, described in Section 3.3, “Key Values,” provide further defense against certain styles of attack.

Access to the chip is through a standard I^2C interface at speeds up to 1Mbit/sec (See Section 6 for details on this interface). It is compatible with standard serial EEPROM I^2C interface specifications. The chip 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. The single-wire interface is described in more detail in Section 5, “Single-wire Interface.”

Using either the I^2C or single-wire interface, multiple ATSHA204 chips 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 4.2, “Sharing the Interface,” and Section 8.10, “Pause Command,” for more details on the way in which this is implemented.

Each ATSHA204 ships with a guaranteed unique 72-bit serial number. Using the cryptographic protocols supported by the chip, a host system or remote server can prove that the serial number is both authentic and 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 Atmel ATSHA204 can generate high-quality random numbers and employ them for any purpose, including as part of the crypto protocols of this chip. Because each 256-bit random number is guaranteed to be unique from all numbers ever generated on this or any other chip, their inclusion in the protocol calculation ensures that replay attacks (re-transmitting a previously successful transaction) always fail. Further information is found in Section 3.4.2, “Random Number Generator,” and Section 8.11, “Random Command.”

System integration is eased with a wide supply voltage range (2.0V through 5.5V) and an ultra-low sleep current of <100nA. Complete DC parameters are found in Section 7, 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 ordering codes.

See Section 9, “Compatibility,” for information regarding compatibility with the Atmel AT88SA102S and 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 chip in the client, which combines that challenge with a secret key via the MAC command from the system, described in Section 8.8, “MAC Command,” and sends the response back to the system. The chip 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.

Due to the flexible command set of the ATSHA204, however, this basic operation can be expanded in many ways. Using the GenDig command (Section 8.5, “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 chip, 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 8.3, “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 chip. 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 (for instance, a mobile phone) needs to verify a client (for instance, 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 8.2, “CheckMac Command”) allows the chip 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 3.3.6 for more details.

Finally, the hash combination of a challenge and secret key can be kept on the chip and XORed with the contents of a slot to implement an encrypted read (Section 8.12, “Read Command”) , or it can be XORed with encrypted input data to implement an encrypted write (Section 8.14, “Write Command”).

Each of these operations can be protected against replay attacks by including a random nonce (Section 8.9, “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 3.1, “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 3.2, “HMAC/SHA-256,” and Section 8.6, “HMAC Command”). The ATSHA204 employs full-sized, 256-bit secret keys to prevent any kind of exhaustive attack.

2. Chip Organization

The chip contains the following memory blocks:

- EEPROM
- SRAM

2.1 EEPROM Organization

The EEPROM contains a total of 5312 bits, and is divided into the following zones:

- **Data**
  A 512-byte (4Kbit) 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 Atmel ATSHA204 chip is attached. 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. The nomenclature SN[a:b] indicates a range of bytes within a field of the configuration section.
- **OTP**
  A 64-byte (512-bit) zone of one-time programmable (OTP) bits. Prior to locking the OTP zone, the bits may be freely written using the standard Write command. The OTP zone can be used to store read-only data. 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 different zones are broken down as follows:

- **Data Section**
  16 slots (512 bytes/32 bytes = 16)
- **Configuration Section**
  Three blocks (88 bytes/32 bytes = 3)
  The 88 bytes are accessible from within a three-block address space
- **OTP Section**
  Two blocks (64 bytes/32 bytes = 2)

Each slot or block has potentially different access permissions.

Note: 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 chip.

Many sections of this document refer to a keyID, which is equivalent to the slot number for those slots designated to hold key values. Key #1 (sometimes referred to as key[1]) is stored in Slot[1], and so on. While all 16 slots can potentially hold keys, those slots for which clear reads are permitted would not normally be used as keys by the crypto commands.

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 chip. See the Atmel website for the document containing the specific shipment values.

2.1.1 Configuration Zone

The 88 bytes in the configuration zone contain manufacturing identification data, general chip 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 2-1.

Table 2-1. Configuration Zone

Byte #	Name	Description	Write	Read
0 → 3	SN[0:3]	Part of the serial number value See the Section 2.1.1.2, "Special Memory Values in the Config Zone"	Never	Always
4 → 7	RevNum	Chip revision number See the Section 2.1.1.2, "Special Memory Values in the Config Zone"	Never	Always
8 → 12	SN[4:8]	Part of the serial number value See the Section 2.1.1.2, "Special Memory Values in the Config Zone"	Never	Always
13	Reserved	Set by Atmel	Never	Always
14	I2C_Enable	Bit 0: 0 = the chip operates in single-wire interface mode 1 = the chip operates in I^2C interface mode Bits 1-7 are ignored: These bits are set at the Atmel factory, depending on the part number, and cannot be changed	Never	Always
15	Reserved	Set by Atmel	Never	Always
16	I2C_Address	Bit 0: Ignored Bits 1-7: For I^2C interface parts, the most-significant seven bits of this byte form the device address value to which this chip will respond. Bits 1, 2, and 4-7 are ignored for single-wire interface parts. Bit 3 (TTLenable): 1 = the input levels are V_{CC} referenced 0 = the input levels use a fixed reference See Section 7.4.1 for more information	If config unlocked	Always
17	RFU	Reserved for future use, must be written to 0x00	If config unlocked	Always
18	OTPmode	0xAA (Read-only mode) = Writes to the OTP zone are forbidden when the OTP zone is locked. Reads of all words are permitted 0x55 (Consumption mode) = Not supported at this time Contact Atmel for more information 0x00 (Legacy mode) = When the OTP zone is locked, writes to the OTP zone, reads of words 0 and 1, and 32-byte reads are all forbidden. See Section 9 for more information on compatibility with the Atmel AT88SA102S device All other values of OTPmode are reserved, and should not be used	If config unlocked	Always
19	SelectorMode	If 0, then Selector can always be written with UpdateExtra. If non-zero, Selector can only be written if it currently has a value of zero See Section 8.13 for more details	If config unlocked	Always
20 → 51	SlotConfig	Two bytes of access and usage permissions and controls for each slot of the data zone. See the SlotConfig (bytes 20 – 51) section below for more details.	If config unlocked	Always

Byte #	Name	Description	Write	Read
52,54,56,58 60,62,64,66	UseFlag	For “single use” keys (Section 3.3), this byte indicates how many times a key may be used before such use is disabled. Applies to keys #0-7 only (byte 52 corresponds to Key0, 54 to Key1, and so on). Initialized to 0xFF. See Section 3.3	If config unlocked	Always
53,55,57,59 61,63,65,67	UpdateCount	For keys that can be updated with DeriveKey, these bytes indicate how many times this operation has been performed. Applies to keys #0-7 only, (byte 53 corresponds to Key0, 55 to Key1 and so on) Initialized to 0x00 See Sections 3.3 and 8.2	If config unlocked	Always
68 → 83	LastKeyUse	128 bits to control limited use for KeyID 15 Initialized to 0xFF See Section 3.3	If config unlocked	Always
84	UserExtra	1-byte value that can be modified via the UpdateExtra command after the data zone has been locked	Via update extra cmd only	Always
85	Selector	Selects which chip will remain in active mode after the execution of the pause command See Sections 8.10 and 8.13 for more details	Via update extra cmd only	Always
86	LockValue	Controls the ability to write the OTP and data zones 0x55 = unlocked 0x00 = locked On shipment from Atmel, this byte has a value of 0x55 corresponding to the “unlocked” state. After the lock command has been run, this byte will have a value of 0x00, corresponding to “locked” See Sections 2.1.2 and 8.7 for more details When locked, the OTP zone can be modified only with the Write command, and slots in the data zone can be modified only if the corresponding WriteConfig field so indicates. When unlocked, the Read command is prohibited within these two zones	Via lock command only	Always
87	LockConfig	Controls the ability to modify the configuration zone 0x55 = unlocked 0x00 = locked On shipment from Atmel this byte has a value of 0x55 corresponding to the “unlocked” state. After the lock command has been run, this byte will have a value of 0x00, corresponding to”locked” See Sections 2.1.2 and 8.7 for more details	Via lock command only	Always

2.1.1.1 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 the following table 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 2-2. SlotConfig Bits (per slot)

Bit #	Name	Description
0 → 3	ReadKey	Use this keyID to encrypt data being read from this slot using the read command. If zero, then this slot can be the source for the CheckMac copy operation (see Section 3.3.6). See the description for bit 6 in this table, the Read command description in Section 8.12, and Table 2-3 for more details. Note: Do *not* use zero as a default. Do *not* set this field to zero unless the CheckMac copy operation is explicitly desired, regardless of any other read/write restrictions
4	CheckOnly	1 = The key stored in the slot can be used only for CheckMac command. It can also be used for GenDig if the subsequent use of TempKey is CheckMac. The key cannot be used by any other command, either directly as KeyID in the input parameter list or via TempKey. 0 = The key stored in the slot can be used by all crypto commands
5	SingleUse	1 = The key stored in the slot is “single use.” See Section 3.3 for more details Ignored for keys in slots 8-14 0 = There are no usage limitations
6	EncryptRead	1 = Reads from this slot will be encrypted using the procedure specified in the Read command description (Section 8.12) using ReadKey (bits 0 – 3 in this table) to generate the encryption key. No input MAC is required. If this bit is set, then IsSecret must also be set (see also Table 2-3) 0 = Clear text reads may be permitted
7	IsSecret	1 = The contents of this slot are secret: Clear text reads are prohibited, and both 4-byte reads and writes are prohibited. This bit *must* be set if EncryptRead is one or if WriteConfig has any value other than <Always> to ensure proper operation of the chip 0 = The contents of this slot should not contain confidential data or keys See Table 2-3 for additional information
8 → 11	WriteKey	Use this key to validate and encrypt data written to this slot See Section 8.14 for additional information
12 → 15	WriteConfig	Controls the ability to modify the data in this slot See Table 2-4 and Section 8.14 for additional information

Read operations depend on the state of IsSecret and EncryptRead according to the following table:

Table 2-3. Read Operation Permission

IsSecret	EncryptRead	Description
0	0	Clear text reads are always permitted from this slot Slots set to this state should never be used as key storage Either 4 or 32 bytes may be read at a time
0	1	Prohibited. No security is guaranteed for slots using this code
1	0	Reads are never permitted from this slot Slots set to this state can still be used for key storage
1	1	Reads from this slot are encrypted using the encryption algorithm documented in the Read command description (See Section 8.12) The encryption key is in the slot specified by ReadKey. 4-byte reads and writes are prohibited

The 4-bit WriteConfig field is interpreted by the Write and DeriveKey commands as shown in Table 2-4 and Table 2-5, 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

Table 2-4. Write Configuration Bits – Write Command

Bit 15	Bit 14	Bit 13	Bit 12	Mode Name	Description
0	0	0	X	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
0	0	1	X	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	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	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 (Section 8.13). 4-byte writes to this slot are prohibited

Table 2-5. Write Configuration Bits – Derivekey Command

Bit 15	Bit 14	Bit 13	Bit 12	Source Key[(1)]	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 3.3 for more details

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 chip 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.

2.1.1.2 Special Memory Values in the Config 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 chip 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 Atmel ATSHA204. Their default value is 0x01 23 EE. These 24 bits are always included in the SHA-256 computations made by the Atmel ATSHA204
  2. **SN[2:3] and SN[4:7]**
     The values of these bits are programmed by Atmel during the manufacturing process and are different for every die. These 48 bits are optionally included in some SHA-256 computations made by the Atmel ATSHA204
- **RevNum**
  Four 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.

2.1.2 Chip Locking

There are two separate lock states for the chip:

1. One to lock the configuration zone (controlled by LockConfig, byte 87)
2. Second to lock both the OTP and data zones (controlled by LockValue, byte 86)

These lock bits 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 chip should be personalized at the system manufacturer with the desired configuration information, after which 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 any writes, the data and OTP sections should be locked. Contact Atmel for optional secure personalization services.

It is vital that the data and OTP sections be locked prior to release of the system containing the chip 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 chip to return an error.

2.1.2.2 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 LockValue byte in the configuration zone, as shown in the following table. Throughout this document, if LockConfig is 0x55, the configuration zone is said to be unlocked; otherwise it is locked.

Table 2-6. Configuration Zone Locking

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

2.1.2.3 Data Zone Locking

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

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

Table 2-7. Data zone access restrictions

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

2.1.2.4 OTP Zone Locking

Reads and writes of the OTP zone depend on the state of the LockConfig, LockValue, and OTPmode bytes in the configuration zone. See Section 2.1.3 for more information.

2.1.3 One Time Programmable (OTP) Zone

This zone of 64 bytes (512 bits) is part of the EEPROM array, and can be used for read-only storage or consumption logging purposes.

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 lockValue), the entire OTP zone can be written using the Write command. If desired, the data to be written can be encrypted. When unlocked, this zone cannot be read at all.

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- or 32-byte reads.
- **Consumption Mode**
  This mode is not currently supported. Contact Atmel for further information.
- **Legacy Mode**
  In this 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 32-bit reads are permitted, and any attempt to execute a 256-bit read will result in an error return code. All write operations to the OTP zone are prohibited. See Section 9 or more of the Atmel ATSA102S compatibility details.

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

2.2 Static RAM (SRAM)

The chip 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 chip 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 below for more details on 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 chip (although the chip itself can read and use the contents internally).

This register contains the elements shown in Table 2-8.

Table 2-8. TempKey Storage Register

Name	Length	Description
TempKey	256 bits (32 bytes)	Nonce (from nonce command) or Digest (from GenDig command)
KeyID	4 bits	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 bit	The source of the randomness in TempKey: 0 = Internally generated random number (*Rand*) 1 = Input seed only, no internal random generation (*Input*)
GenData	1 bit	0 = TempKey.KeyID 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.KeyID will be meaningful)
CheckFlag	1 bit	If 1, 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 0.
Valid	1 bit	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 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, however, are retained when the chip 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. Cryptographic Information

The ATSHA204 implements a challenge-response protocol using either SHA-256 or HMAC/SHA-256, details of which are below. The response is always a 256-bit digest.

The Nonce command (see Section 8.9) accepts an input challenge from the system and optionally combines it with an internally generated random number to generate a nonce (number used once) for the calculation. This seed is then combined with a secret key as part of the authentication calculation for any of the crypto commands (MAC, HMAC, Read, Write, or GenDig). For compatibility reasons, the input challenge may be passed directly to the MAC command; however, this operation is deprecated.

The chip can guarantee the uniqueness of the Nonce only if the chip has included the output of its random number generator in the calculation, because the system input may or may not be unique. Every random Nonce is guaranteed to be unique when compared to all previous nonces, ensuring that each transaction is unique over all time.

3.1 SHA-256

The ATSHA204 MAC command calculates the digest of a secret key concatenated with the challenge or nonce. It optionally includes various other pieces of information stored on the chip within the digested message.

The ATSHA204 computes the SHA-256 digest based on the algorithm documented here:

http://csrc.nist.gov/publications/fips/fips180-2/fips180-2.pdf

The complete SHA-256 message processed by the ATSHA204 is listed in Sections 8.5 and 8.9 for each of the particular commands (GenDig and Nonce) that use the algorithm. Most standard software implementations of the algorithm automatically add the appropriate number of pad and length bits to this message to match the operation the chip performs internally.

The SHA-256 algorithm is also used for encryption by taking the output digest of the hash algorithm and XORing it with the plain text data to produce the ciphertext. Decryption is the reverse – the ciphertext is XORed with the digest, and the result is the plain text.

3.2 HMAC/SHA-256

The response to the challenge can also be computed using the HMAC algorithm based on SHA-256 documented here:

http://csrc.nist.gov/publications/fips/fips198/fips-198a.pdf

Because of the increased computation complexity, the HMAC command is not as flexible as the MAC command and the computation time for HMAC is extended. While the HMAC sequence is not necessary to ensure the security of the digest, it is included for compatibility with various software packages.

3.3 Key Values

All keys within the CryptoAuthentication family are 256 bits long. The ATSHA204 uses these keys as part of the messages hashed with the MAC, CheckMac, HMAC, and GenDig commands. Any slot in the data zone of the EEPROM can be used to store a key, though the value will be secret only if the read and write permissions are properly set within SlotConfig (including the IsSecret bit).

Except for the GenDig command, all but the least-significant four bits of the KeyID parameter are ignored in determining the source of key data. Only the least-significant four bits are used to select one of the slots of the data zone. See Section 3.3.7, below, for information on how GenDig uses other KeyID values.

In all cases for which a SHA-256 calculation is performed using Param2, the entire 16-bit KeyID as input is included in the message.

3.3.1 Diversified Keys

If the host or validating entity has a place to securely store secrets, the key values stored in the EEPROM slot(s) can be diversified with the serial number embedded in the chip (SN[0:8]). In this manner, every client chip can have a unique key, which can provide extra protection against known plaintext attacks and permit compromised serial numbers to be identified and blacklisted.

To implement this, a root secret is externally combined with the chip serial number during personalization using some cryptographic algorithm and the result written to the ATSHA204 key slot.

The ATSHA204 CheckMac command provides a mechanism of securely generating and comparing diversified keys, eliminating this requirement from the host system.

Consult the following application note for more details:

http://www.atmel.com/dyn/resources/prod_documents/doc8666.pdf

3.3.2 Rolled Keys

In order to prevent repeated use of the same key value, the ATSHA204 supports key rolling. Normally, after a certain number of uses (perhaps as few as one), the current key value is replaced with the SHA-256 digest of its current value combined with some offset, which may either be a constant, something related to the current system (for example, a serial number or model number), or a random number.

This capability is implemented using the DeriveKey command. Prior to execution of the DeriveKey command, the Nonce command must be run to load the offset into TempKey. Each time the roll operation is performed on slots 0-7, the UpdateCount field for that slot is incremented.

One use for this capability is to permanently remove the original key from the device, replacing it with a key that is only useful in a particular environment. After the key is rolled, there is no possible way to retrieve the old value, which improves the security of the system.

Note: Any power interruption during the execution of the DeriveKey command in Roll mode may cause either the key or the UpdateCount to have an unknown value. If writing to a slot is enabled using bit number 14 of SlotConfig, such keys can be written in encrypted and authenticated form using the Write command. Alternatively, multiple copies of the key can be stored in multiple slots so that failure of a single slot does not incapacitate the system.

3.3.3 Created Keys

In order to support unique ephemeral keys for every client, the ATSHA204 also supports key creation. In this mechanism, a “parent” key (specified by slotConfig.writeKey) is combined with a fixed or random nonce to create a unique key, which is then used for any cryptographic purpose.

The ability to create unique keys is especially useful if the parent key has usage restrictions (see “Single-use Keys” and “Limited-use Key” in the following sections). In this mode, the limited use parent key can be employed to create an unlimited use child key. Because the child key is useful only for this particular host-client pair, attacks on its value are less valuable.

This capability is also implemented using the DeriveKey command. Prior to execution of the DeriveKey command, the Nonce command must be run to load the Nonce value into TempKey. Each time the create operation is performed on slots 0-7; the UpdateCount field for that slot is incremented.

3.3.4 Single-use Keys

For the KeyID values corresponding to slots 0-7 in the data section of the EEPROM, repeated usage of the key stored in the slot can be strictly limited. This feature is enabled if the SingleUse bit is set in the SlotConfig field. The SingleUse bit is ignored for slots 8-14. The number of remaining uses is stored as a bit map in the UseFlag byte corresponding to the slot in question.

Prior to execution of any cryptographic command that uses this slot as a key, the following takes place:

- If SlotConfig[keyId].SingleUse is set and UseFlag[KeyID] is 0x00, the chip returns an error
- Starting at bit seven of UseFlag[keyID], clear to zero the first bit that is currently a one

In practice, this procedure permits SingleUse keys to be used eight times between “refreshes” using the DeriveKey command. If power is lost during the execution of any command referencing a key that has this feature enabled, one of the use bits in UseFlag may still be cleared even though the command did not complete. For this reason, Atmel recommends that the key be used a single time only, with the other bits providing a safety margin for errors.

Under normal circumstances, all eight UseFlag bytes should be initialized to 0xFF. If it is the intention to permit fewer than eight uses of a particular key, these bytes should be initialized to 0x7F (seven uses), 0x3F (six uses), 0x1F (five uses), 0x0F (four uses), 0x07 (three uses), 0x03 (two uses), or 0x01 (one use). Initialization to any other value besides these values or 0xFF is prohibited.

The Read, Write, and DeriveKey commands operate slightly differently:

- **Read and Write**

  These commands ignore the state of the SingleUse bit and the UseFlag byte does not change as a result of their execution. SingleUse slots in which the UseFlag is exhausted (value of 0x00) can still be read or written (subject to the appropriate SlotConfig limitations) although the value in the slot cannot ever be used as a key for cryptographic commands.

  If SlotConfig.WriteKey for slot X points back to X, but UseFlag[X] is exhausted, encrypted writes to the slot will never succeed because the prior GenDig command will have returned an error due to the usage limitation. A similar situation occurs with reads and ReadKey. Slots used as keys should never have IsSecret set to zero or WriteConfig set to Always.

- **DeriveKey**

  If the parent key is used for either authentication or as the source, then if SingleUse (for the parent) is set and UseFlag (also for the parent) is 0x00, the DeriveKey command returns an error. The SingleUse and UseFlag bits are ignored for the target key. When successfully executed, DeriveKey always resets the UseFlag to 0xFF for the target key – this is the only mechanism to reset the UseFlag bits.

  Use of the DeriveKey command is optional – it is legal to be run only if WriteConfig: 13 is set for this slot. In some situations, it may be advantageous to simply have a key that can be used eight times, in which case the other crypto commands will clear the bits in UseFlag one at a time until all are cleared, and at which time the key is disabled.

3.3.5 Limited-use Key

If Slot[15].SingleUse is set, usage of key number 15 is limited through a different mechanism than the single-use limitation described above, which applies only to slots 0-7.

Prior to any use of key 15 by a cryptographic command, the following takes place:

- If all bytes in LastKeyUse are 0x00, return error
- Starting at bit seven of the first byte of LastKeyUse (byte 68 in config zone), clear to zero the first bit that is currently a zero. If byte 68 is 0x00, check bit seven of byte 69, and so on up through byte 83. Only a single bit is cleared each time prior to using key 15

There is no reset mechanism for this limitation – after 128 uses (or the number of one bits set in LastKeyUse on personalization), key 15 is permanently disabled. This capability is not susceptible to power interruptions – even if the power is interrupted during execution of the command, only a single bit in LastKeyUse will be unknown; all other bits in LastKeyUse will be unchanged and the key will remain unchanged.

If fewer than 128 uses are desired for key 15, then some of the bytes within this array should not be initialized to 0xFF. As with UseFlag, the only legal values for bytes within this field (besides 0xFF) are 0x7F, 0x3F, 0x1F, 0x0F, 0x07, 0x03, 0x01, or 0x00. The total number of bits set to one indicates the number of uses. One example of how to set 16 uses is as follows: 0xFF, 0xFF, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00, 0x00.

The SingleUse bit is ignored by the Read and Write commands, and lastKeyUse does not change as a result of their execution. The SingleUse bit is ignored by the copy function of the CheckMac command. The SingleUse bit is honored for the parent key in the DeriveKey command, but is ignored for the target key.

3.3.6 Password Checking

Many applications require a user to enter a password to enable features, decrypt stored data, or some other purpose. Typically, the expected password has to be stored somewhere in memory and is, therefore, subject to discovery. The ATSHA204 can securely store the expected password and perform a number of useful operations on it. The password is never passed in the clear to the chip, nor can it be read from the chip. It is hashed with a random number in the system software before being passed to the chip.

The copy capability of the CheckMac command enables the following types of password checking options:

1. CheckMac does an internal comparison with the expected password and returns a Boolean to the system to indicate whether the password was correctly entered or not
2. If the chip determines that the correct password has been entered, then the value of the password can optionally be combined with a stored or ephemeral value to create a key that can be used by the system for data protection purposes
3. If the chip determines that the correct password has been entered, the chip can use this fact to optionally release a secondary, high entropy secret, which can be used for data protection without risk of any exhaustive dictionary attack
4. If the password has been lost, an entity with knowledge of a parent key value can optionally write a new password into the slot. Also optionally, the current value can be encrypted with a parent key and read from the chip

Passwords should be stored in even-numbered slots. If the password is to be mapped to a secondary value (use #3 above), then the target slot containing this value is located in the next higher slot number (the password slot number plus one). Otherwise, the target slot is the same as the password slot.

ReadKey for the target slot must be set to zero to enable this capability. In order to prevent fraudulent or unintended usage of this capability, do not set ReadKey for any slot to zero unless this CheckMac/copy capability is specifically required. In particular, do not assume that other bits in the configuration word for a particular slot override the enablement of this capability specified by ReadKey = 0.

This capability is enabled only if the mode parameter to CheckMac has a value of 0x01, indicating that

a. The first 32 bytes of the SHA-256 message are stored in a data slot in the EEPROM (the password)
b. The second 32 bytes of the SHA-256 message must be a randomly generated nonce in the TempKey register

If the above conditions are met and the input response matches the internally generated digest, then the contents of the target key are copied to TempKey. The other TempKey register bits are set as follows:

- SourceFlag is set to one (not random)
- GenData is set to zero (not generate by the GenData command)
- CheckFlag is set to zero (TempKey is not restricted to the CheckMac command)
- Valid is set to one

3.3.7 Transport Keys

The ATSHA204 chip includes an internal hardware array of keys (transport keys) that are intended for secure personalization prior to locking of the data section. The values of the hardware keys are kept secret, and are made available to qualified customers upon request to Atmel. These keys can be used with the GenDig command only, and are indicated by a KeyID value ≥ 0x8000.

This is the intended personalization command flow:

1. Write intended values to the configuration zone, and then lock the configuration zone.
2. Write non-secret slots and OTP zone, data should be passed to the chip in the clear.
3. Generate a random personalization key in any one of the secret slots with the following sequence:
   a. Nonce command to generate a random nonce in TempKey
   b. Gendig specifying a transport key ≥ 0x8000
   c. Gendig using the compliance (default) value stored in the slot to be used for personalization
   d. Encrypted write to that same slot (overwrites the compliance value)

4. Use that personalization key to write all the secret slots, ending with the final value of the personalization key slot itself, using the following sequence repeated as necessary:
    a. Nonce command to generate a random nonce in TempKey
    b. Gendig specifying the personalization key
    c. Encrypted write to the target slot
5. Lock the data zone

For GenDig and all other commands, KeyID values less than 0x8000 always reference keys that are stored in the data zone of the EEPROM. In these cases, only the four least-significant bits of KeyID are used to determine the slot number, while the entire 16-bit KeyID as input is used in any SHA-256 message calculation.

3.4 Security Features

3.4.1 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 chip.

3.4.2 Random Number Generator

The ATSHA204 includes a high-quality random number generator that returns 32 random bytes to the system. The chip combines this generated number with a separate input number to form a nonce that is stored within the chip 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 chip. The chip provides a special Random command for such purposes, which does do not affect the internally stored nonce.

To simplify system testing, prior to config locking 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 chip and the first byte into the SHA message.

To prevent replay attacks on encrypted data that is passed to or from the ATSHA204, the chip requires that a new, internally generated nonce be included as part of the encryption sequence used to protect the data being written or read. 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 chip that are invalidated if the chip 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 8.9 and Section 8.11 for more information about how the EEPROM seed update is controlled.

4. General I/O Information

Communications with the ATSHA204 are achieved through one of two different protocols, and selected using the part number that is ordered:

- **Single-wire Interface**
  Uses a single GPIO connection on the system microprocessor connected to SDA on the chip. It permits the fewest number of connector pins to any removable/replaceable entity. The bit rate is up to 26Kbits/sec
- **I^{2}C Interface**
  This mode is compatible with the Atmel AT24C16B serial EEPROM interface. Two pins, serial data (SDA), and serial clock (SCL) are required. The I^{2}C interface supports a bit rate of up to 1Mbit/sec.

The lowest levels of the I/O protocols are described below in Sections 5 and 6. Above the I/O protocol level, however, exactly the same bytes are transferred to and from the chip to implement the cryptographic commands and error codes documented in Section 8.

Note: The chip 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 8.1.6 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
- 16-bit (2-byte) integers, typically Param2 appear on the bus least-significant byte first

In this document, the most-significant bit or nibble of a byte or 16-bit word appears towards the left hand side of the page.

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

- On the one-wire interface, data is transferred to/from the Atmel ATSHA204 least-significant bit first on the bus
- On the I^{2}C interface, data is transferred to/from the Atmel 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 KeyID of slot x. In the example below, K[x] indicates the KeyID 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 8.8, “MAC Command.”

4.2 Sharing the Interface

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

a. System issues a Wake token (see Section 5.1) to wake up all chips
b. The system issues the Pause command to put all but one of the devices into idle mode. Only the remaining chip 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 chips ignore but which puts the single remaining active chip into the idle mode. See Section 8.10, "Pause Command," for more details.

Steps a and b are repeated for each chip on the wire. If the system has completed communications with the final chip, it should wake all the chips up and then put all the chips to sleep to reduce total power consumption.

The chip uses the selector byte within the configuration zone to determine which chip stays awake – only that chip 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-chip sharing mode, the following three update capabilities for the selector byte are supported:

1. **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.
2. **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 LockValue byte.
3. **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. Single-wire Interface

In this mode, communications to and from the ATSHA204 take place over SDA, 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:

Tokens	I/O tokens implement a single data bit transmitted on the bus, or the wake-up event.
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.
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 Atmel ATSHA204.

See the Atmel website for the appropriate application notes for more details on how to use any microprocessor to easily generate the signaling necessary to send these elements to the chip, including C source code libraries. Also, see Section 10.2, "Wiring Configuration for Single-wire Interface," for more information about how to connect the chip in the single-wire interface mode.

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 Atmel ATSHA204)

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

Output: (from the Atmel ATSHA204)

ZeroOut	Send a single bit from the chip to the system with a value of zero
OneOut	Send a single bit from the chip 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 timings are designed to permit a standard UART running at 230.4K baud 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 chip.

The Wake token is special in that it requires an extra long low pulse on the SDA pin, which cannot be confused with the shorter low pulses that occur during a data token (Zero, One, ZeroOut, OneOut). Chips 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 chips 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 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 chip 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
0x77	Command	After this flag, the system starts sending a command block to the chip The first bit of the block can follow immediately after the last bit of the flag
0x88	Transmit	This command tells the chip to wait for a bus turnaround time and then start transmitting its response to the previously transmitted command block
0xBB	Idle	Upon receipt of an idle flag, the chip goes into the idle mode and remains there until the next wake token is received
0xCC	Sleep	Upon receipt of a sleep flag, the chip enters the low-power sleep mode until the next Wake token is received
All other values are reserved and should not be used		

5.2.1.1 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 chip returns to the system depend on the current state of the chip, and may include either status, error code, or command results.

When the chip is busy executing a command, it ignores the SDA pin and any flags sent by the system. See Table 8-4 for execution delays in the chip for each command type. The system must observe these delays after sending a command to the chip.

5.2.1.2 Idle Flag

The idle flag is used to transition the ATSHA204 to the idle state, which causes the input/output buffer to be flushed. It does *not* invalidate the contents of the TempKey and random number generator (RNG) seed registers. This flag can be sent to the chip at any time when it will accept a flag. When the chip is in the idle state, the watchdog timer is disabled.

5.2.1.3 Sleep Flag

The sleep flag transitions the ATSHA204 to the low-power sleep state, which causes a complete reset of the chip, including invalidation of the contents of the SRAM and all volatile registers. This flag can be sent to the chip at any time when it will accept a flag.

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 chip 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 chip within the $t_{TIMEOUT}$ interval. Failure to send enough bits or the transmission of an illegal token (a low pulse exceeding t_{ZLO}) will cause the chip to enter the sleep state after the $t_{TIMEOUT}$ 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 so the total time to transmit the command may exceed the $t_{TIMEOUT}$ interval while the time between bits may not

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

5.3.2 Synchronization Procedures

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

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

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

6. I^{2}C Interface

The I^{2}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 the AT24C16B serial EEPROM operating at 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

6.1 I/O Conditions

The chip responds to the following I/O conditions:

6.1.1 Chip is Asleep

When the chip is asleep, it ignores all but the WAKE condition

Wake: If SDA is held low for a period greater than t_{WLO}, the chip exits low-power mode and, after a delay of tWHI, is ready to receive I^{2}C commands. The chip ignores any levels or transitions on the SCL pin when the chip 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, “Chip 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 chip is awake, the normal processor I^{2}C hardware and/or software can be used for chip communications up to and including the I/O sequence required toput the chip back into low power (sleep) mode.

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

In I^{2}C mode, the chip will ignore a wake sequence sent when the chip is already awake.

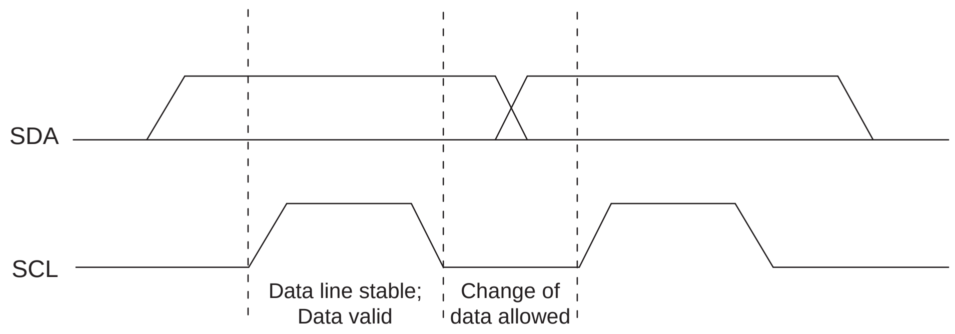
6.1.2 Chip is Awake

When the chip 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.

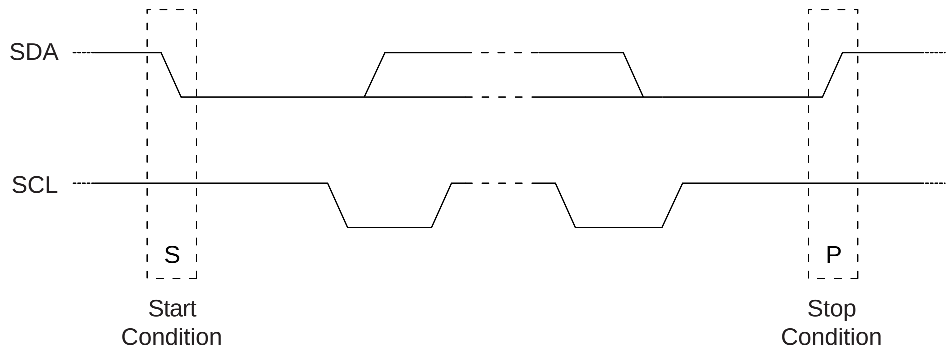
Figure 6-1. Data Bit Transfer on I^{2}C Interface



Start: A high-to-low transition of SDA with SCL high is a start condition, which must precede all commands.

Stop: A low-to-high transition of SDA with SCL high is a stop condition. After this condition is received by the chip, the current I/O transaction ends. On input, if the chip has sufficient bytes to execute a command, the chip transitions to the busy state and begins execution. The stop condition should always be sent after any packet sent to the chip.

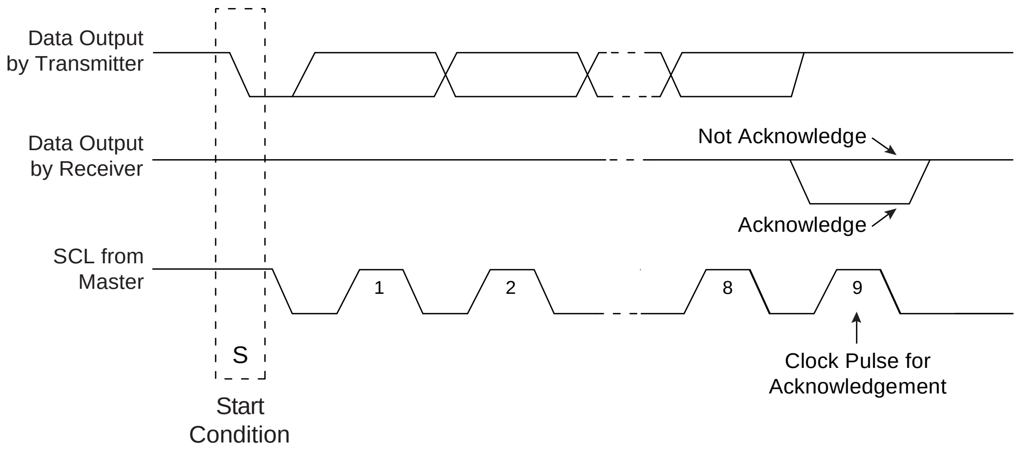
Figure 6-2. Start and Stop Conditions on I^{2}C Interface



Acknowledge (ACK): On the ninth clock cycle after every address or data byte is transferred, the receiver will pull the SDA pin low to acknowledge proper reception of the byte.

Not Acknowledge (NACK): Alternatively, on the ninth clock cycle after every address or data byte is transferred, the receiver can leave the SDA pin high to indicate that there was a problem with the reception of the byte or that this byte completes the block transfer.

Figure 6-3. NACK and ACK Conditions on I^{2}C Interface



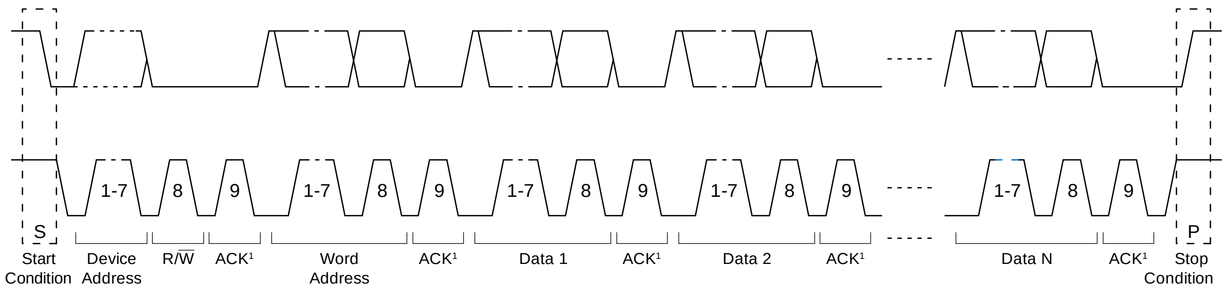
Multiple ATSHA204 chips can easily share the same I^{2}C interface if the I2C_Device address byte is programmed differently for each device on the bus. Because six of the bits of the device address are programmable, the ATSHA204 can also share the I^{2}C interface with any standard I^{2}C chip, including any serial EEPROM. Bit 3 (also known as TTL Enable) must be programmed according the input thresholds desired, and so is fixed in a particular application.

6.2 I^{2}C Transmission to the Atmel ATSHA204 Device

The transmission of data from the system to the AT88SA102S is summarized in Figure 6-4. The order of transmission is:

- Start condition
- Device address byte
- Word address byte
- Optional data bytes (1 through N)
- Stop condition

Figure 6-4. Normal I^{2}C Transmission to an Atmel ATSHA204



Note: SDA is driven low by the Atmel ATSHA204 during the ACK periods

The tables below label the bytes of the I/O transaction. The I^{2}C name column provides the names of the bytes as described in the AT24C16B datasheet.

Table 6-1. I^{2}C Transmission to Atmel ATSHA204

Atmel ATSHA204	I^{2}C Name	Description
Device address	Device address	This byte selects a particular chip on the I^{2}C interface. The Atmel ATSHA204 is selected if bits 1-7 of this byte match bits 1-7 of the I2C_Address byte in the configuration zone. Bit 0 of this byte is the standard I^{2}C R/W bit, and should be zero to indicate a write operation (the bytes following the device address travel from the master to the slave).
Word address	Word address	This byte should have a value of 0x03 for normal operation. See Sections 6.2.2 and 6.6, below, for more information.
Command	Data1-n	The command block, consisting of the count, command packet, and the two-byte CRC. The CRC is calculated over the size and packet bytes. See Section 8.1.

Because the chip treats the command input buffer as a FIFO, the input block can be sent to the chip in one or many I^{2}C command blocks. The first byte sent to the chip is the count, and so after the chip receives that number of bytes, it will ignore any subsequently received bytes until execution is finished.

The system *must* send a stop condition after the last command byte to ensure that the ATSHA204 will start the computation of the command. Failure to send a stop condition may eventually result in a loss of synchronization (See Section 6.7 for recovery procedures).

6.2.1 Word Address Values

During an I^2C write packet, the ATSHA204 interprets the second byte sent as the word address, which indicates the packet function, as described in the following table.

Table 6-2. Word Address Values

Name	Value	Description
Reset	0x00	Reset the address counter. The next read or write transaction will start with the beginning of the I/O buffer
Sleep (low power)	0x01	The Atmel ATSHA204 goes into the low-power sleep mode and ignores all subsequent I/O transitions until the next Wake flag. The entire volatile state of the chip is reset.
Idle	0x02	The Atmel ATSHA204 goes into the idle state and ignores all subsequent I/O transitions until the next Wake flag. The contents of TempKey and RNG seed registers are retained
Command	0x03	Write subsequent bytes to sequential addresses in the input command buffer that follow previous writes. This is the normal operation
Reserved	0x04 – 0xFF	These addresses should not be sent to the chip

6.2.2 Command Completion Polling

After a complete command has been sent to the ATSHA204, the chip will be busy until the command computation completes. The system has two options for this delay:

- **Polling**
  The system should wait t_{EXEC} (typical) and then send a read sequence (See Section 6.5). If the chip NACKs the device address, then it is still busy. The system may delay for some time or immediately send another read sequence, again looping on NACK. After a total delay of t_{EXEC} (max), the chip will have completed the computation and return the results.
- **Single Delay**
  The system should wait t_{EXEC} (max), after which the chip will have completed execution and the result can be read from the chip using a normal read sequence.

6.3 Sleep Sequence

Upon completion of system use of the ATSHA204, the system should issue a sleep sequence to put the chip into low-power mode. This sequence consists of the proper device address followed by the value of 0x01 as the word address followed by a stop condition. This transition to the low-power state causes a complete reset of the chip internal command engine and input/output buffer. It can be sent to the chip at any time when it is awake and not busy.

6.4 Idle Sequence

If the total sequence of required commands exceeds $t_{WATCHDOG}$, then the chip will automatically go to sleep and lose any information stored in the volatile registers. This action can be prevented by putting the chip into the idle state prior to completion of the watchdog interval. When the chip receives the wake token, it will then restart the watchdog timer, and execution can be continued.

The idle sequence consists of the proper device address followed by the value of 0x02 as the word address followed by a stop condition. It can be sent to the chip at any time when it is awake and not busy.

If TempKey was created as a result of the copy mode of the CheckMac command, it will *not* be retained when the part goes into an idle state.

6.5 I²C Transmission from the Atmel ATSHA204 Device

When the ATSHA204 is awake and not busy, the bus master can retrieve the current buffer contents from the chip using an I²C read. If valid command results are available, the size of the block returned is determined by the particular command that has been run (See Section 8. "Commands"), otherwise the size of the block (and the first byte returned) will always be four: count, status/error, and 2-byte CRC. The bus timing is shown in Figure 7-3 in Section 7.3.2.

Table 6-3. I²C transmission from Atmel ATSHA204

Atmel ATSHA204	I²C Name	Direction	Description
Device Address	Device Address	To slave	This byte selects a particular chip on the I²C interface, and the Atmel ATSHA204 will be selected if bits 1-7 of this byte match bits 1-7 of the I2C_Address byte in the configuration zone. Bit 0 of this byte is the standard I²C R/W pin, and should be one to indicate that the bytes following the device address travel from the slave to the master (read)
Data	$\text{Data}_{1,N}$	To master	The output block, consisting of the count and status/error byte or the output packet followed by the two-byte CRC per Section 8.1

The status, error, or command outputs can be read repeatedly by the master. Each time a read command is sent to the ATSHA204 along the I²C interface, the chip transmits the next sequential byte in the output buffer. See the following section for details on how the chip handles the address counter.

If the ATSHA204 is busy, idle, or asleep, it will NACK the device address on a read sequence. If a partial command has been sent to the chip, then it will NACK the device address, but float the bus during the data intervals.

6.6 Address Counter

Writes to and/or reads from the ATSHA204 I/O buffer over the I²C interface are treated as if the chip were a FIFO. Either the I²C byte or block write/read protocols can be used. The number of bytes transferred with each block sequence does not affect the operation of the chip.

The first byte transmitted to the chip is treated as the size byte. Any attempt to send more than this number of bytes or any attempts to write beyond the end of the I/O buffer (84 bytes) will cause the ATSHA204 to NACK those bytes.

After the host writes a single command byte to the input buffer, reads from the host are prohibited until after the chip completes command execution. Attempts to read from the chip prior to the last command byte being sent will result in an ACK of the device address but all ones (0xFF) on the bus. If the master attempts to send a read byte to the chip during command execution, the chip will NACK the device address.

Data may be read from the chip under the following three conditions:

- On power up, the single byte, 0x11 (See Section 8.1.3), can be read inside a four byte block
- If a complete block has been received by the chip, but there are any errors in parsing or executing the command, a single byte of error code is available, also inside a four byte block
- Upon completion of command execution, from 1-32 bytes of command result are available to be read inside a block of 4-35 bytes

Any attempt to read beyond the end of the valid output buffer returns 0xFF to the system – the address counter does not wrap around to the beginning of the buffer.

There may be situations where the system may wish to re-read the output buffer; for example, when the CRC check reveals an error. In this case, the master should send a two-byte sequence to the ATSHA204 consisting of the correct device address and a word address of 0x00 (Reset, per Table 6-2), followed by a stop condition. This causes the address counter to be reset to zero, and permits the data to be re-written (re-read) to (from) the chip. This address reset sequence does not prohibit subsequent read operations if data was available for reading in the I/O buffer prior to the sequence execution.

After one or more read operations to retrieve the results of a command execution, the first write operation resets the address counter to the beginning of the I/O buffer.

6.7 I^{2}C Synchronization

It is possible for the system to lose synchronization with the I/O port on the ATSHA204, perhaps due a system reset, I/O noise, or other condition. Under this circumstance, the ATSHA204 may not respond as expected, may be asleep, or may be transmitting data during an interval when the system is expecting to send data. Any command results in the I/O buffer may be lost when the system and chip lose synchronization.

To re-synchronize, the following procedure should be followed:

1. To ensure an I/O channel reset, the system should send the standard I^{2}C software reset sequence, as follows:
   - A start condition
   - Nine cycles of SCL with SDA held high
   - Another start condition
   - A stop condition

   It should then be possible to send a read sequence, and, if synchronization has completed properly, the ATSHA204 will ACK the device address. The chip may return data or may leave the bus floating (which the system will interpret as a data value of 0xFF) during the data periods.

   If the chip does ACK the device address, the system should reset the internal address counter to force the ATSHA204 to ignore any partial input command that may have been sent. This can be accomplished by sending a write sequence to word address 0x00 (Reset), followed by a stop condition.
2. If the chip does *not* respond to the device address with an ACK, then it may be asleep. In this case, the system should send a complete wake token and wait t_{WHI} after the rising edge. The system may then send another read sequence, and, if synchronization has completed, the chip will ACK the device address.
3. If the chip still does not respond to the device address with an ACK, then it may be busy executing a command. The system should wait the longest t_{EXEC} (max) and then send the read sequence, which will be acknowledged by the chip.

7. Electrical Characteristics

7.1 Absolute Maximum Ratings*

Operating temperature	−40°C to +85°C
Storage temperature	−65°C to + 150°C
Maximum operating voltage	6.0V
DC output current	5.0mA
Voltage on any pin	-0.5V to (V_{CC} + 0.5V)

*Notice: Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only, and functional operation of the device at these or any other condition beyond those indicated in the operational sections of this specification is not implied. Exposure to absolute maximum rating conditions for extended periods may affect device reliability.

7.2 Reliability

The ATSHA204 is fabricated with the high reliability of the Atmel CMOS EEPROM manufacturing technology.

Table 7-1. EEPROM reliability

Parameter	Min	Typical	Max	Units
Write Endurance (each byte at 25°C)	100,000			Write cycles
Data Retention (at 55°C)	10			Years
Data Retention (at 35°C)	30	50		Years
Read Endurance	Unlimited			Read cycles

7.3 AC Parameters – All I/O Interfaces

Figure 7-1. AC Timing Diagram – All I/O Interfaces

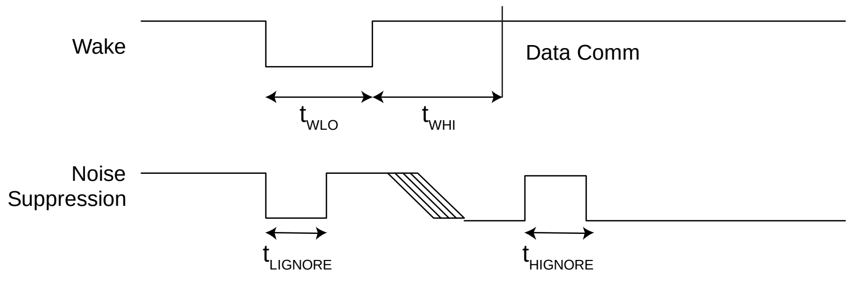


Table 7-2. AC Parameters – All I/O Interfaces

Parameter	Symbol	Direction	Min	Typ	Max	Unit	Notes
Wake low duration	t_{WLO}	To Crypto Authentication	60		-	µs	SDA can be stable in either high or low levels during extended sleep intervals
Wake high delay to data comm.	t_{WHI}	To Crypto Authentication	2.5			ms	SDA should be stable high for this entire duration
High side glitch filter @ active	$t_{HIGNORE\_A}$	To Crypto Authentication	45[1]			ns	Pulses shorter than this in width will be ignored by the chip, regardless of its state when active
Low side glitch filter @ active	$t_{LIGNORE\_A}$	To Crypto Authentication	45[1]			ns	Pulses shorter than this in width will be ignored by the chip, regardless of its state when active
High side glitch filter @ sleep	$t_{HIGNORE\_S}$	To Crypto Authentication	15[1]			µs	Pulses shorter than this in width will be ignored by the chip when in sleep mode
Low side glitch filter @ sleep	$t_{LIGNORE\_S}$	To Crypto Authentication	15[1]			µs	Pulses shorter than this in width will be ignored by the chip when in sleep mode
Watchdog reset	$t_{WATCHDOG}$	To Crypto Authentication	0.7	1.3	1.7	s	Max. time from wake until chip is forced into sleep mode (See Section 8.1.6)

Note: 1. These parameters are guaranteed through characterization, but not tested

7.3.1 AC Parameters – Single-wire Interface

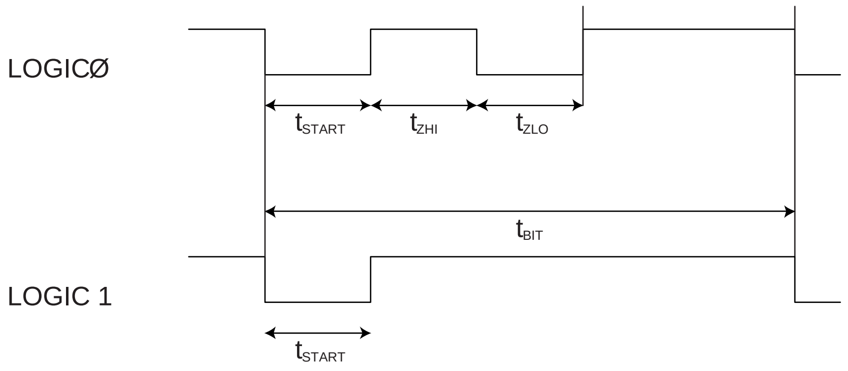


Figure 7-2. AC Timing Diagram – Single-wire Interface

Table 7-3. AC Parameters – Single-wire Interface

Applicable from TA = −40°C to +85°C, V_{CC} = +2.0V to +5.5V, CL =100pF (unless otherwise noted)

Parameter	Symbol	Direction	Min	Typ	Max	Unit	Notes
Start Pulse Duration	t_{START}	To Crypto Authentication	4.1	4.34	4.56	µs	
		From Crypto Authentication	4.6	6.0	8.6	µs	
Zero Transmission High Pulse	t_{ZHI}	To Crypto Authentication	4.1	4.34	4.56	µs	
		From Crypto Authentication	4.6	6.0	8.6	µs	
Zero Transmission Low Pulse	t_{ZLO}	To Crypto Authentication	4.1	4.34	4.56	µs	
		From Crypto Authentication	4.6	6.0	8.6	µs	
Bit Time(1)	t_{BIT}	To Crypto Authentication	37	39	-	µs	If the bit time exceeds $t_{TIMEOUT}$, then the Atmel ATSHA204 may enter the sleep state. See Section 5.3.1 for specific details.
		From Crypto Authentication	41	54	78	µs	
Turnaround Delay	$t_{TURNAROUND}$	From Crypto Authentication	28	60	95	µs	The Atmel ATSHA204 will initiate the first low-going transition after this time interval following the end of the last bit (t_{BIT}) of the Transmit flag
		To Crypto Authentication	15			µs	After the Atmel ATSHA204 transmits the last bit of a block, the system must wait this interval before sending the first bit of a flag
I/O Timeout	$t_{TIMEOUT}$	To Crypto Authentication	45	65	85	ms	The Atmel ATSHA204 may transition to the sleep state if the bus is inactive longer than this duration. See Section 5.3.1 for specific details.

Note: 1. t_{START}, t_{ZLO}, t_{ZHI}, and t_{BIT} are designed to be compatible with a standard UART running at 230.4K baud for both transmit and receive. The UART should be set to seven data bits, no parity, and one stop bit

7.3.2 AC Parameters – I^{2}C Interface

Figure 7-3. I^{2}C Synchronous Data Timing

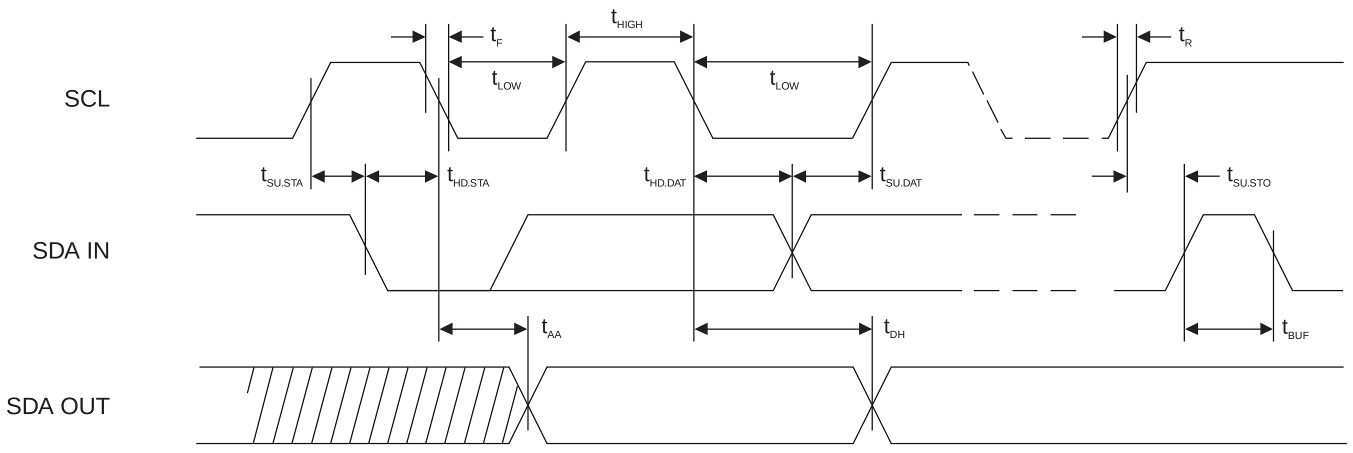


Table 7-4. AC Characteristics of I^{2}C Interface

Applicable over recommended operating range from T_A = −40°C to + 85°C, V_{CC} = +2.0V to +5.5V, CL = 1 TTL gate and 100pF (unless otherwise noted)

Symbol	Parameter	Min	Max	Units
f_{SCK}	SCK clock frequency	0	1	MHz
	SCK clock duty cycle	30	70	percent
t_{HIGH}	SCK high time	400		ns
t_{LOW}	SCK low time	400		ns
$t_{SU.STA}$	Start setup time	250		ns
$t_{HD.STA}$	Start hold time	250		ns
$t_{SU.STO}$	Stop setup time	250		ns
$t_{SU.DAT}$	Data in setup time	100		ns
$t_{HD.DAT}$	Data in hold time	0		ns
t_R	Input rise time (1)		300	ns
t_F	Input fall time (1)		100	ns
t_{AA}	Clock low to data out valid	50	550	ns
t_{DH}	Data out hold time	50		ns
t_{BUF}	Time bus must be free before a new transmission can start. (1)	500		ns

Notes: 1. Values are based on characterization, but are not tested

2. AC measurement conditions:
RL (connects between SDA and V_{CC}): 2.0kΩ (for V_{CC} +2.0V to +5.0V)
Input pulse voltages: 0.3V_{CC} to 0.7V_{CC}
Input rise and fall times: ≤ 50ns
Input and output timing reference voltage: 0.5V_{CC}

7.4 DC Parameters – All I/O Interfaces

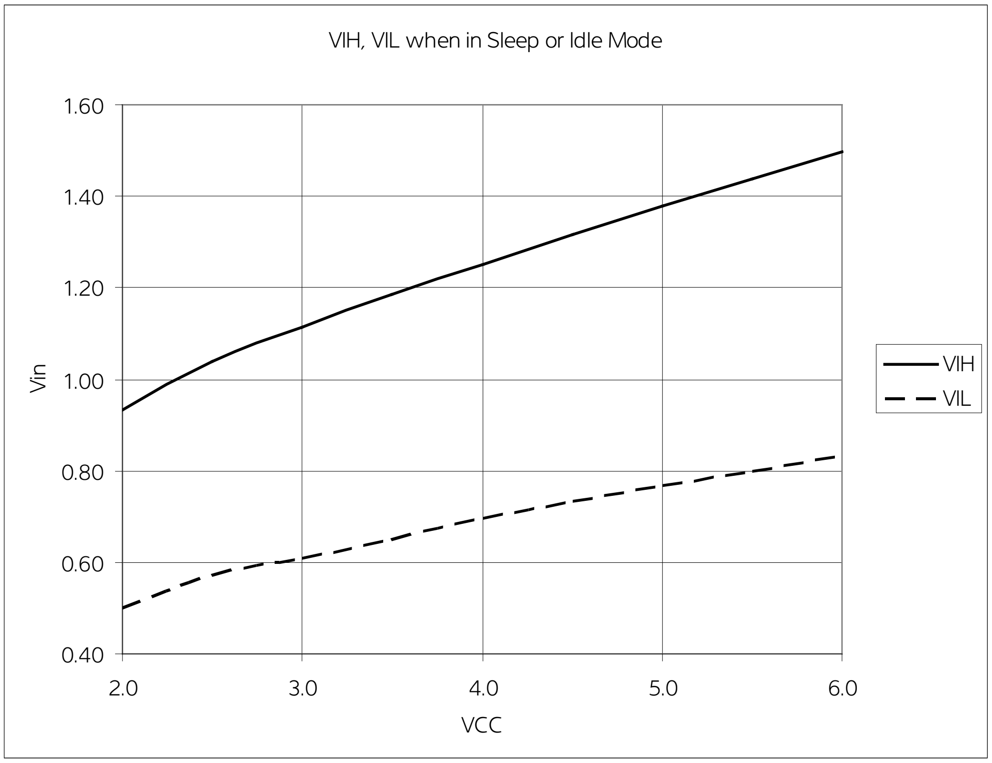
Table 7-5. DC Parameters – All I/O Interfaces

Parameter	Symbol	Min	Typ	Max	Unit	Notes
Ambient Operating Temperature	T_A	-40		85	°C	
Power Supply Voltage	V_{CC}	2.0		5.5	V	
Active Power Supply Current	I_{CC}		1		mA	0°C →+70°C, V_{CC} = 3.3V
			-	3	mA	-40°C →+85°C, V_{CC} = 5.5V
Idle Power Supply Current	I_{IDLE}		700		µA	When chip is in idle mode, V_{CC} = 3.3V, V_{SDA} & V_{SCL} < 0.3V or > > V_{CC}-0.3
Sleep Current	I_{SLEEP}		30	150	nA	When chip is in sleep mode, $V_{CC} \leq 3.6V$, V_{SDA} & V_{SCL} < 0.3V or > V_{CC}-0.3, $T_A \leq 55°C$
				2	µA	When chip is in sleep mode
Output Low Voltage	V_{OL}			0.4	V	When chip is in active mode, V_{CC} = 2.5 – 5.5V
Output Low Current	I_{OL}			4	mA	When chip is in active mode, V_{CC} = 2.5 – 5.5V, V_{OL} = 0.4V

7.4.1 V_{IH} and V_{IL} Specifications

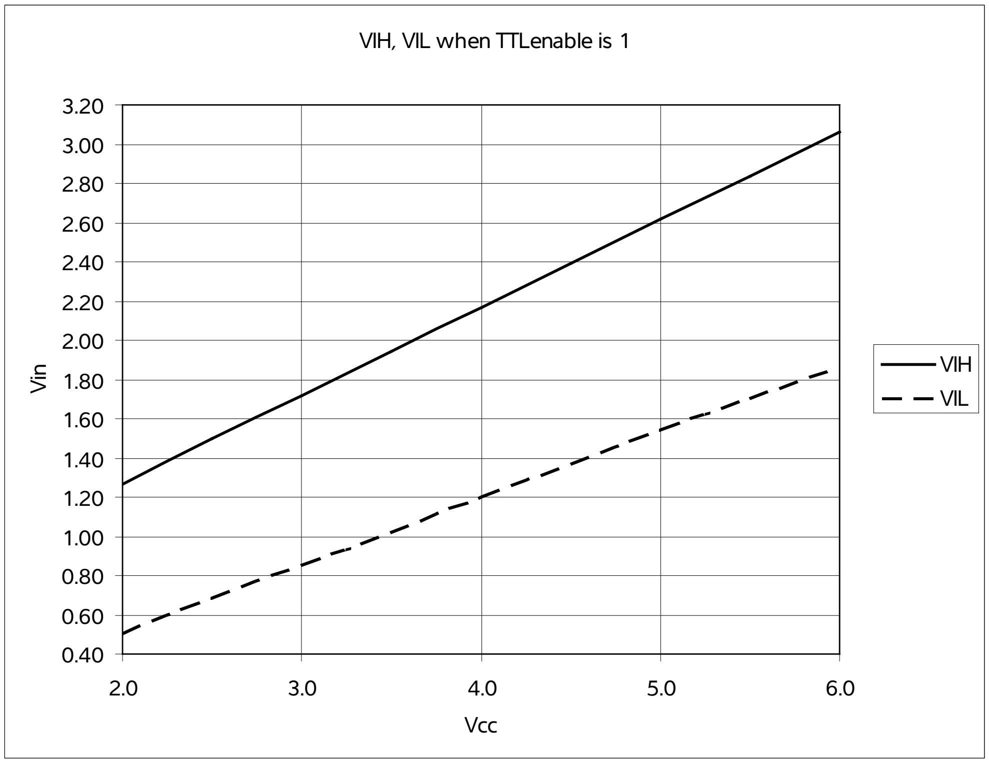
The input voltage thresholds when in sleep or idle mode are dependent on the VCC level as follows:

Figure 7-4. V_{IH} and V_{IL} When in Sleep or Idle Mode



When the chip is active (not in sleep or idle mode), the input voltage thresholds are different, depending on the state of TTLenable (bit three) within the I2C_Address byte stored in the configuration zone of the EEPROM. When a common voltage is used for the ATSHA204 V_{CC} pin and the input pull-up resistor, then this bit should be set to a one, which permits the input thresholds to track the supply as follows:

Figure 7-5. V_{IH} and V_{IL} (Chip Active, TTLenable = 1) – All I/O Interfaces



If the voltage supplied to the V_{CC} pin of the ATSHA204 is different from the system voltage to which the input pull-up resistor is connected, then the system designer may chose to set TTLenable to zero, which enables a fixed input threshold according to the following table.

Table 7-6. V_{IL} and V_{IH} (Chip Active, TTLenable = 0) – All I/O Interfaces

Parameter	Symbol	Min	Typ	Max	Unit	Notes
Input Low Voltage	V_{IL}	GND-0.5		0.5	V	When chip is active and TTLenable bit in configuration memory is zero; otherwise, see above
Input High Voltage	V_{IH}	1.5		V_{CC} + 0.5	V	When chip is active and TTLenable bit in configuration memory is zero; otherwise, see above

8. Commands

8.1 I/O Blocks

Regardless of the I/O protocol being used (single-wire or I^2C), commands are sent to the chip, and responses received from the chip, within a block that is constructed in the following way:

Table 8-1. I/O blocks

Byte #	Name	Meaning
0	Count	Number of bytes to be transferred to (or from) the chip in the block, including count byte, packet bytes, and checksum bytes. The count byte should, therefore, always have a value of (N+1), where N is equal to the number of bytes in the packet plus the two checksum bytes. Thus, for a block with one count byte, 50 packet bytes, and two checksum bytes, the count byte should be set to 53. The maximum size block (and value of count) is 84 bytes, and the minimum size block is four bytes. Values outside this range will cause unpredictable operation
1 to (N-2)	Packet	Command, parameters and data, or response. See below for more details
N-1, N	Checksum	CRC-16 verification of the count and packet bytes. The CRC polynomial is 0x8005. The initial register value should be zero. After the last bit of the count and the packet have been transmitted, the internal CRC register should have a value that matches the checksum bytes in the block. The first CRC byte transmitted (N-1) is the least-significant byte of the CRC value, and so the last byte of the block is the most-significant byte of the CRC

The ATSHA204 is designed in such a way that the count value in the input block should be consistent with the size requirements specified in the command parameters. If the count value is inconsistent with the command opcode and/or parameters within the packet, the ATSHA204 will respond in different ways, depending on the specific command. Either the response may include an error indication or some input bytes may be silently ignored.

8.1.1 Command Packets

The command packet is broken down as shown in Table 8-2.

Table 8-2. Command packets

Byte #	Name	Meaning
0	Opcode	The command code – see Section 8.1.4
1	Param1	The first parameter – always present
2-3	Param2	The second parameter – always present
4+	Data	Optional remaining input data

After the ATSHA204 receives all the bytes in a block, the chip transitions to the busy state and attempts to execute the command. Neither status nor results can be read from the chip when it is busy. During this time, the chip's I/O interface ignores all SDA transitions regardless of the I/O interface selected. The command execution delays are listed in Section 8.1.4

If insufficient bytes are sent to the chip when it is in one-wire mode, the chip automatically transitions to the low-power sleep state after the $t_{TIMEOUT}$ interval. In I^2C mode, the chip continues to wait for the remaining bytes until the watchdog timer limit, $t_{WATCHDOG}$, is reached or a start/stop condition is received by the chip.

In the individual command description tables below in Sections 8.2 through 8.13, the size column describes the number of bytes in the parameter documented in each particular row. If the input block size for a particular command is incorrect, the chip does not attempt to execute the command; instead, the chip returns an error.

8.1.2 Status/Error Codes

The chip does not have a dedicated status register, and so the output FIFO is shared among status, error, and command results. All output from the chip is returned to the system as complete blocks, which are formatted identically to input blocks:

- Count
- Packet
- 2-byte CRC

After the chip receives the first byte of an input command block, the system cannot read anything from the chip until the system has sent all the bytes to the chip.

After wake and after execution of a command, there will be error, status, or result bytes in the chip's output register that can be retrieved by the system. When the length of that block is four bytes, the codes returned are detailed below in Table 8-3. Some commands return more than four bytes when they execute successfully: the resulting packet description is listed in the command section below.

CRC errors are always returned before any other type of error. They indicate that some sort of I/O error occurred and that the command may be resent to the chip. If a command includes both parse and execution errors, there is no particular precedence enforced – an execution error may occur before a parse error and/or the reverse.

Table 8-3. Status/error Codes in 4-byte Blocks

State Description	Error/Status	Description
Successful Command Execution	0x00	Command executed successfully
Checkmac Miscompare	0x01	The CheckMac command was properly sent to the chip, but the input client response did not match the expected value
Parse Error	0x03	Command was properly received, but the length, command opcode, or parameters are illegal, regardless of the state (volatile and/or EEPROM configuration) of the ATSHA204 Changes in the value of the command bits must be made before it is re-attempted
Execution Error	0x0F	Command was properly received, but could not be executed by the chip in its current state Changes in the chip state or the value of the command bits must be made before it is re-attempted
After Wake, but prior to first command	0x11	Indication that the ATSHA204 has received a proper wake token
CRC or other Communications Error	0xFF	Command was *not* properly received by the ATSHA204, and should be re-transmitted by the I/O driver in the system No attempt was made to parse or execute the command

8.1.3 Command Opcodes, Short Descriptions, and Execution Times

During parsing of the parameters and subsequent execution of a properly received command, the chip will be busy and not respond to transitions on the pins. The interval during which the chip will be busy varies depending on the command and its parameter values, the state of the chip, the environmental conditions, and other factors per the table below.

Table 8-4. Command Opcodes, Short Descriptions, and Execution Times

Command	Opcode	Description	Typ. Exec. Time[1], ms	Max. Exec. Time[2], ms
DeriveKey	0x1C	Derive a target key value from the target or parent key	14	62
DevRev	0x30	Return device revision information	0.4	2
GenDig	0x15	Generate a data protection digest from a random or input seed and a key	11	43
HMAC	0x11	Calculate response from key and other internal data using HMAC/SHA-256	27	69
CheckMac	0x28	Verify a MAC calculated on another Atmel CryptoAuthentication device	12	38
Lock	0x17	Prevent further modifications to a zone of the chip	5	24
MAC	0x08	Calculate response from key and other internal data using SHA-256	12	35
Nonce	0x16	Generate a 32-byte random number and an internally stored nonce	22	60
Pause	0x01	Selectively put just one chip on a shared bus into the idle state	0.4	2
Random	0x1B	Generate a random number	11	50
Read	0x02	Read four bytes from the chip, with or without authentication and encryption	0.4	4
UpdateExtra	0x20	Update bytes 84 or 85 within the configuration zone after the configuration zone is locked	8	12
Write	0x12	Write 4 or 32 bytes to the chip, with or without authentication and encryption	4	42

Notes: 1. Typical execution times are representative of the duration to execute the command assuming no error conditions, fastest mode setting, no optional internal actions such as limited use keys, and favorable environmental conditions. For best performance, delay for this interval and then start polling to determine actual command completion

2. Maximum execution times are representative of the longest duration of a successful command execution with all mode and internal actions enabled under extended statistical and environmental conditions. Execution time may extend beyond these values in extreme situations. In most but not all cases, failing commands will return relatively quickly, often well before the typical execution time

8.1.4 Address Encoding

The Read and Write commands include a single address in Param2 that indicates the memory to be accessed. All Reads and Writes are in units of four bytes (one word). The most-significant byte of a legal ATSHA204 address is always zero. All unused address bits should always be set to zero. The least-significant bits in the address describe the offset to the first word to be accessed within the block/slot, while the upper bits specify the block/slot number per the table below:

Table 8-5. Address Encoding (Param2)

Zone	Byte 1	Byte 0		
	Unused	Unused	Block/Slot	Offset
Config	Bits 0 → 7	Bits 5 → 7	Bits 3 → 4	Bits 0 → 2
OTP	Bits 0 → 7	Bits 4 → 7	Bit 3	Bits 0 → 2
Data	Bits 0 → 7	Bit 7	Bits 3 → 6	Bits 0 → 2

Within each zone, there are various access restrictions per the table below:

Table 8-6. Legal Block/slot Values

Zone	Legal Block/Slot (inclusive)	Notes
Config	0-2	Addresses below 16 (block 0, offset 16) and above 87 (block 2, offset 23) can never be written Addresses above 87 can never be read. Both 4- and 32-byte reads/writes are permitted
OTP	0-1	When OTPmode is read-only, all offsets in both blocks are available to use with 4- or 32-byte reads If OTPmode is consumption, then writes are also permitted to all offsets See Section 2.1.3 if OTPmode is Legacy
Data	0-15	All offsets in all slots available for both read and write 4-byte access permitted on a particular slot only if SlotConfig.IsSecret is zero

In the table below, "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-bit) basis, the word address in the table below should be used for the address parameter passed to the Read and Write commands.

Table 8-7. Data Zone Slots

Slot #	Byte Address (Hex)	Word Address (Hex)	Slot #	Byte Address (Hex)	Word Address (Hex)
0	0x0000	0x0000	8	0x0100	0x0040
1	0x0020	0x0008	9	0x0120	0x0048
2	0x0040	0x0010	10	0x0140	0x0050
3	0x0060	0x0018	11	0x0160	0x0058
4	0x0080	0x0020	12	0x0180	0x0060
5	0x00A0	0x0028	13	0x01A0	0x0068
6	0x00C0	0x0030	14	0x01C0	0x0070
7	0x00E0	0x0038	15	0x01E0	0x007F

8.1.5 Zone Encoding

The value in Param1 controls which zone the command accesses. See Section 2.1.2.2, “Configuration Zone Locking,” to obtain more information on what controls the “locked” and “unlocked” states for each zone. All other zone values are reserved, and should not be used.

Table 8-8. Zone Encoding (Param1)

Zone Name	Param1 Value	Size	Read	Write
Config	0	512 bits, 64 bytes, 2 slots	Always available	Partially, when unlocked Never when locked Never encrypted
OTP	1	512 bits, 64 bytes, 2 slots	Never when unlocked. Always when locked, except in legacy mode. See Section 2.1.3	All writeable when unlocked using Write When locked, write permissions depend on OTPmode See Section 2.1.3
Data	2	4096 bits, 512 bytes, 16 slots	Never when unlocked. Otherwise, controlled by IsSecret and EncryptRead.	All writeable when unlocked When locked, writes controlled by WriteConfig

8.1.6 Watchdog Failsafe

A watchdog counter starts within the chip after the ATSHA204 receives a wake token. After $t_{WATCHDOG}$, the chip enters sleep mode, regardless of whether some I/O transmission or command execution is in progress. There is no way to reset the counter other than to put the chip into sleep or idle mode and then wake it up again.

The watchdog timer is implemented as a failsafe mechanism so that no matter what happens on either the system side or inside the chip, including any I/O synchronization issue, power consumption will fall to the ultra-low sleep level automatically.

The chip resets the values stored in the SRAM and internal status registers when it transitions to the sleep state. However, if the chip is explicitly put into the idle mode through the appropriate I/O sequence, the chip retains the contents of the two SRAM registers (TempKey and RNG seed).

Normally, all command sequences must complete within $t_{WATCHDOG}$ if they require state that is stored in the SRAM registers. The system software can use this idle mode mechanism to implement a longer command sequence than can be completed during a single watchdog interval.

8.2 CheckMac Command

The CheckMac command calculates a MAC response that would have been generated on a CryptoAuthentication chip and compares that with an input value. It returns a Boolean to indicate the success or failure of the comparison.

Prior to running this command, the Nonce and/or GenDig commands may have been optionally run to create a key or nonce value in TempKey. The input mode parameter determines the source of the “key” (the first 32 bytes of the SHA message) and “challenge/nonce” (the second 32 bytes of the SHA message).

If the comparison matches, then the target EEPROM slot value may be copied into TempKey. If KeyID is even, then the target slot is KeyID+1, else the target slot is KeyID. For the copy to take place, the mode parameter to CheckMac must have a value of 0x01 and SlotConfig.ReadKey for the target key must be zero. When CheckMac is loaded in this manner, it will not be retained when the chip enters the idle state

Table 8-9. Input parameters

	Name	Size	Notes
Opcode	CHECKMAC	1	0x28
Param1	Mode	1	Bit 0: If zero, the second 32 bytes of the SHA message are taken from the input ClientChal parameter. If one, the second 32 bytes of the message are taken from TempKey Bit 1: If zero, use key[KeyID] in first SHA block. If one, use TempKey. Bit 2: If Mode:0 or Mode:1 are set, then the value of this bit must match the value in TempKey.SourceFlag or the command will return an error. Bit 5: If one, use 64 bits of OTP zone in calculation. If zero, use 64 zeros Bits 3-4 and 6-7: Must be zero
Param2	KeyID	2	Which internal key is to be used to generate the response. All but bits 0:3 are ignored
Data1	ClientChal	32	Challenge sent to client. If Mode:0 is one, then the value of this parameter will be ignored (though these 32 bytes MUST still appear in the input stream)
Data2	ClientResp	32	Response generated by the client
Data3	OtherData	13	Remaining constant data needed for response calculation

Table 8-10. Output parameter

Name	Size	Notes
Result	1	Returns a single byte with a value of zero if ClientResp matches the internally computed digest, one if there is a mismatch

The message that will be hashed with the SHA-256 algorithm consists of the following information:

32 bytes	key[KeyID] or TempKey (depending on mode)
32 bytes	ClientChal or TempKey (depending on mode)
4 bytes	OtherData[0:3]
8 bytes	OTP[0:7] (or 0s depending on mode)
3 bytes	OtherData[4:6]
1 byte	SN[8]
4 bytes	OtherData[7:10]
2 bytes	SN[0:1]
2 bytes	OtherData[11:12]

8.3 DeriveKey Command

The chip combines the current value of a key with the Nonce stored in TempKey using SHA-256, and places the result into the target key slot. SlotConfig[TargetKey]. Bit13 must be set or DeriveKey will return an error.

If SlotConfig[TargetKey].Bit12 is zero, the source key that will be combined with TempKey is the target key specified in the command line (Roll Key operation). If SlotConfig[TargetKey].Bit12 is one, the source key is the parent key of the target key, which is found in SlotConfig[TargetKey].WriteKey (Create Key operation).

Prior to execution of this command, the Nonce command must have been run to create a valid nonce in TempKey. Depending on the state of bit two of the input mode, this nonce must have been created with the internal random number generator, or it must have been fixed.

If SlotConfig[TargetKey].Bit15 is set, an input MAC must be present and have been computed as:

SHA-256(ParentKey, Opcode, Param1, Param2, SN[8], SN[0:1])

where the ParentKey ID is always SlotConfig[TargetKey].WriteKey.

If SlotConfig[TargetKey].Bit12 or SlotConfig[TargetKey].Bit15 is set and SlotConfig[ParentKey].SingleUse is also set, DeriveKey returns an error if UseFlag[ParentKey] is 0x00. DeriveKey ignores SingleUse and UseFlag for the target key if SlotConfig[TargetKey].Bit12 and SlotConfig[TargetKey].Bit15 are both zero.

For slots 0-7 only, if input parsing and the optional MAC check succeed, UseFlag[TargetKey] gets set to 0xFF and UpdateCount[TargetKey] is incremented. If UpdateCount currently has a value of 255, it wraps to zero. If the command fails for any reason, these bytes are not updated. The value of UpdateCount may be corrupted if power is interrupted during the execution of DeriveKey.

Note: If the source and target key are the same, there is a risk of permanent loss of the key value if power is interrupted during the write operation. If the configuration bits permit it, the key slot may be recovered using an authenticated and encrypted write based on the parent key

Table 8-11. Input Parameters

	Name	Size	Notes
Opcode	DERIVEKEY	1	0x1C
Param1	Random	1	Bit 2: The value of this bit must match the value in TempKey.SourceFlag or the command will return an error Bits 0:1, 3:7: Must be zero
Param2	TargetKey	2	Key slot to be written
Data	Mac	0 or 32	Optional MAC used to validate operation

Table 8-12. Output parameter

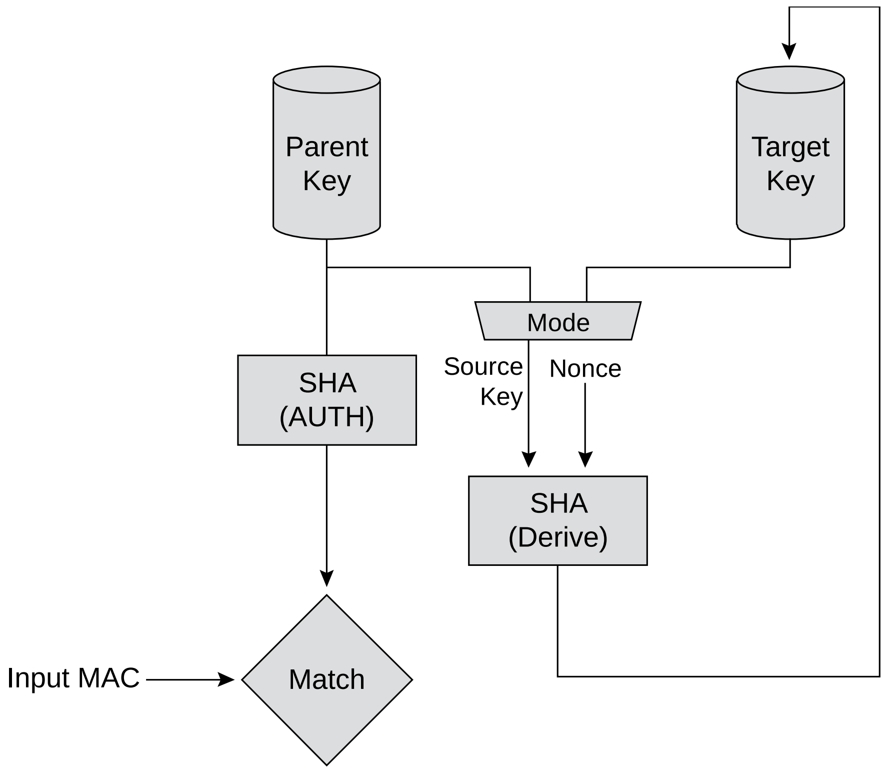
Name	Size	Notes
Success	1	Upon successful completion, the ATSHA204 returns a value of zero

The key written to the target slot is the result of a SHA-256 of the following message:

32 bytes	Target or parent key (depending on SlotConfig Bit12)
1 byte	Opcode
1 byte	Param1
2 bytes	Param2
1 byte	SN[8]
2 bytes	SN[0:1]
25 bytes	Zeros
32 bytes	TempKey.value

The data flow for this command is shown graphically in the figure below:

Figure 8-1. Data Flow for DeriveKey Command



8.4 DevRev Command

DevRev command returns a single four-byte word representing the revision number of the device. Software should not depend on this value as it may change from time to time.

Table 8-13. Input Parameters

	Name	Size	Notes
Opcode	DEVREV	1	0x30
Param1	Mode	1	Must be zero
Param2	-	2	Must be zero
Data	-	0	-

Table 8-14. Output Parameters

Name	Size	Notes
Success	4	The current device revision number

8.5 GenDig Command

Uses SHA-256 to combine a stored value with the contents of TempKey, which must have been valid prior to the execution of this command. The stored value can come from one of the data slots, either of the OTP pages, either of the first two pages of the configuration zone, or retrieved from the hardware transport key array. The resulting digest is retained in TempKey, and can be used in one of three ways:

1. It can be included as part of the message used by the MAC, CheckMac, or HMAC commands. Because the MAC response output incorporates both the data used in the GenDig calculation and the secret key from the MAC command, it serves to authenticate the data stored in the data and/or OTP zones
2. A subsequent Read or Write command can use the digest to provide authentication and/or confidentiality for the data, in which case it is known as a data protection digest
3. You can use this command for secure personalization by using a value from the transport key array. The resulting data protection digest would then be used by the Write Command

If Zone is two (Data) and KeyID is ≤15, the GenDig command sets TempKey.GenData to one and TempKey.KeyID to the input KeyID; otherwise, TempKey.GenData is set to zero.

Regardless of how the resulting digest is computed, it can never be read from the chip.

If TempKey.Valid is invalid, this command returns an error. Upon command completion, the TempKey.Valid bit is set, indicating that a digest has been loaded and is ready for use. The TempKey.Valid bit is cleared when the next command is executed. See Section 2.2 for more details.

For all KeyID values less than 0x8000, the chip uses the least-significant four bits of KeyID to determine the slot number from which to retrieve the key value from the data zone of the EEPROM. KeyID values above 0x8000 reference keys stored in the masks of the design. In any event, all 16 bits of KeyID as input to the chip are used as Param2 in the SHA-256 calculation.

If the Zone parameter points to the configuration zone, then this command returns an error if the configuration zone is unlocked.

When the key specified on input to GenDig has the CheckOnly bit set, GenDig can be used to generate ephemeral keys matching those generated on client CryptoAuthentication chips using the DeriveKey command. Keys that have the CheckOnly bit set represent situations in which the chip is acting as a host. In this case, the opcode and parameter bytes that would normally be included in the SHA calculation are replaced with bytes from the input stream.

Table 8-15. Input parameters

	Name	Size	Notes
Opcode	GENDIG	1	0x15
Param1	Zone	1	If 0x00 (Config): Use KeyID to specify either the first (KeyID=0) or second (KeyID = 1) 256-bit block of the configuration zone If 0x01 (OTP): Use KeyID to specify either the first or second 256-bit block of the OTP zone If 0x02 (Data): KeyID specifies a slot in the data zone or a transport key in the hardware array All other values are reserved and must not be used
Param2	KeyID	2	Identification number of the key to be used, or selection of which OTP block
Data	OtherData	4 or 0	4 bytes of data for SHA calculation when using a CheckOnly key; otherwise ignored

Table 8-16. Output parameter

Name	Size	Notes
Success	1	Upon successful execution, the Atmel ATSHA204 returns a value of zero

If Zone is data and SlotConfig[KeyID].CheckOnly is one, the SHA-256 message body used to create the resulting new TempKey consists of the following bytes:

32 bytes	Data.slot[KeyID]
4 bytes	OtherData
1 byte	SN[8]
2 bytes	SN[0:1]
25 bytes	Zeros
32 bytes	TempKey.value

In all other cases, the message use to create TempKey is as follows:

32 bytes	Config[KeyID] or OTP[KeyID] or Data.slot[KeyID] or TransportKey[KeyID]
1 byte	Opcode
1 byte	Param1
2 bytes	Param2
1 byte	SN[8]
2 bytes	SN[0:1]
25 bytes	Zeros
32 bytes	TempKey.value

8.6 HMAC Command

Computes a HMAC/SHA-256 digest of a key stored in the chip, a challenge, and other information on the chip. The output of this command is the output of the HMAC algorithm computed over this key and message. If the message includes the serial number of the chip, the response is said to be diversified.

The normal command flow to use this command is as follows:

1. Run Nonce command to load input challenge and optionally combine it with a generated random number. The result of this operation is a nonce stored internally on the chip
2. Optionally run GenDig command to combine one or more stored EEPROM locations in the chip with the nonce. The result is stored internally in the chip
3. Run this HMAC command to combine the output of steps one (and step two if desired) with an EEPROM key to generate an output response

Step two addresses multiple use models. If the data in the EEPROM is a key, GenDig has the effect of authenticating the challenge with multiple secret keys. Alternatively, if the contents of the slot are data (which does not have to necessarily even be secret), GenDig has the effect of authenticating the value stored in that location.

Table 8-17. Input parameters

	Name	Size	Notes
Opcode	HMAC	1	0x11
Param1	Mode	1	Controls which fields within the chip are used in the message
Param2	KeyID	2	Which key is to be used to generate the response Bits 0:3 only are used to select a slot but all 16 bits are used in the HMAC message
Data	-	0	-

Table 8-18. Output parameter

Name	Size	Notes
Response	32	HMAC digest

The HMAC digest is computed using the key at KeyID as the HMAC key over a message consisting of the following information:

32 bytes	Zeros
32 bytes	TempKey
1 byte	Opcode (always 0x11)
1 byte	Mode
2 bytes	KeyID
8 bytes	OTP[0:7] (or zeros, see Table 8-19)
3 bytes	OTP[8:10] (or zeros, see Table 8-19)
1 byte	SN[8] bits (never zeroed out)
4 bytes	SN[4:7] bits (or zeros, see Table 8-19)
2 bytes	SN[0:1] (never zeroed out)
2 bytes	SN[2:3] (or zeros, see Table 8-19)

See the NIST HMAC specification for a complete description of how the various digests are calculated using SHA-256, the HMAC key, and appropriate padding. See http://csrc.nist.gov/publications/fips/fips198/fips-198a.pdf. The padding is part of the SHA-256 message (see Section 3.1). HMAC is a construct that sits on top of SHA-256.

Table 8-19. Mode Encoding

Bits	Meaning
7	Must be zero
6	If set, include the 48 bits SN[2:3] and SN[4:7] in the message Otherwise, the corresponding message bits are set to zero
5	Include the first 64 OTP bits (OTP[0] through OTP[7]) in the message Otherwise, the corresponding message bits are set to zero If Mode[4] is set, the value of this mode bit is ignored
4	Include the first 88 OTP bits (OTP[0] through OTP[10]) in the message Otherwise, the corresponding message bits are set to zero
3	Must be zero
2	The value of this bit must match the value in TempKey.SourceFlag or the command will return an error
0-1	Must be zero

8.7 Lock Command

Write either LockConfig or LockValue to 0xFF, thereby changing the permissions in the designated zone.

This command fails if the designated zone is already locked.

Prior to locking the chip, the ATSHA204 uses the CRC-16 algorithm to generate a summary digest of the designated zone(s). The calculation is made identically to the CRC computed over the input and output blocks.

- For the configuration zone, the CRC is calculated over all 88 bytes
- For the data and OTP zones, their contents are concatenated in that order to create the input to the CRC algorithm

If the input summary does not match that computed on the chip, an error is returned and the personalization process should be repeated.

Table 8-20. Input Parameters

	Name	Size	Notes
Opode	LOCK	1	0x17
Param1	Zone	1	Bit 0: Zero for config zone, 1 for data and OTP zones Bits 1-6: Must be zero Bit 7: If one, the check of the zone CRC is ignored and the zone is locked, regardless of the state of the memory. Atmel does not recommend using this mode
Param2	Summary	2	Summary of the designated zones, or should be 0x0000 if Zone[7] is set
Data	-	0	-

Table 8-21. Output Parameter

Name	Size	Notes
Success	1	Upon successful execution, the Atmel ATSHA204 returns a value of zero

8.8 MAC Command

Computes a SHA-256 digest of a key stored in the chip, a challenge, and other information on the chip. The output of this command is the digest of this message. If the message includes the serial number of the chip, the response is said to be diversified.

The normal command flow to use this command is as follows:

1. Run Nonce command to load input challenge and optionally combine it with a generated random number. The result of this operation is a nonce stored internally on the chip
2. Optionally run GenDig command to combine one or more stored EEPROM locations in the chip with the nonce. The result is stored internally in the chip. This capability permits two or more keys to be used as part of the response generation
3. Run this MAC command to combine the output of step one (and step two if desired) with an EEPROM key to generate an output response (or digest)

Alternatively, data in any slot (which does not have to necessarily even be secret) can be accumulated into the response through the same GenDig mechanism. This has the effect of authenticating the value stored in that location.

Table 8-22. Input Parameters

	Name	Size	Notes
Opcode	MAC	1	0x08
Param1	Mode	1	Controls which fields within the chip are used in the message
Param2	KeyID	2	Which internal key is to be used to generate the response Bits 0:3 only are used to select a slot but all 16 bits are used in the SHA-256 message
Data	Challenge	0 or 32	Input portion of message to be digested, ignored if Mode:0 is one

Table 8-23. Output Parameter

Name	Size	Notes
Response	32	SHA-256 digest

The message that will be hashed with the SHA-256 algorithm consists of the following information:

32 bytes	key[KeyID] or TempKey (See Table 8-24)
32 bytes	Challenge or TempKey (See Table 8-24)
1 byte	Opcode (always 0x08)
1 byte	Mode
2 bytes	Param2
8 bytes	OTP[0:7] (or zeros, see Table 8-24)
3 bytes	OTP[8:10] (or zeros, see Table 8-24)
1 byte	SN[8] bits (never zeroed out)
4 bytes	SN[4:7] bits (or zeros, see Table 8-24)
2 bytes	SN[0:1] (never zeroed out)
2 bytes	SN[2:3] (or zeros, see Table 8-24)

Table 8-24. Mode Encoding

Bits	Meaning
7	Must be zero
6	If set, include the 48 bits SN[2:3] and SN[4:7] in the message Otherwise, the corresponding message bits are set to zero
5	Include the first 64 OTP bits (OTP[0] through OTP[7]) in the message Otherwise, the corresponding message bits are set to zero If Mode[4] is set, the value of this mode bit is ignored
4	Include the first 88 OTP bits (OTP[0] through OTP[10]) in the message Otherwise, the corresponding message bits are set to zero
3	Must be zero
2	If either Mode:0 or Mode:1 are set, Mode:2 must match the value in TempKey.SourceFlag or the command will return an error
1	If zero, the first 32 bytes of the SHA message are loaded from one of the data slots If one, the first 32 bytes are filled with TempKey
0	If zero, the second 32 bytes of the SHA message are taken from the input Challenge parameter If one, the second 32 bytes are filled with the value in TempKey. This mode is recommended for all use

8.9 Nonce Command

This command generates a nonce for use by a subsequent GenDig, MAC, HMAC, Read, or Write command by combining an internally generated random number with an input value from the system. The resulting Nonce is stored internally in TempKey and the generated random number is returned to the system.

The input value is designed to prevent replay attacks against the host ─ it must be externally generated by the system and passed into the chip using this command. It may be any value that changes consistently, such as a nonvolatile counter, current real time of day, and so on, or it can be an externally generated random number.

To provide a Nonce value for subsequent crypto commands, the input number and output random number are hashed together per the information listed below. The resulting digest (nonce) is always stored in the TempKey register, TempKey.Valid is set, and TempKey.SourceFlag is set to "Rand." The Nonce can be used by a subsequent GenDig, Read, Write, HMAC, or MAC command – thus, the system must externally compute this digest value and store it externally to complete the execution of those commands.

Alternatively, this command can also be run in a pass-through mode if a fixed nonce is required for subsequent commands. In this case, the input value must be 32 bytes long, and it is passed directly to TempKey without modification. No SHA-256 calculation is performed, and TempKey.SourceFlag is set to "Input." The nonce value in TempKey may not be used with Read or Write commands. If operated in this mode and with a repeated input number value, the chip provides no protection against replay attacks.

Prior to the configuration section being locked, the random number generator produces a value of 0xFF FF 00 00 FF FF 00 00 to facilitate testing. This test value is combined with the input value in the manner described above.

Table 8-25. Input Parameters

	Name	Size	Notes
Opode	Nonce	1	0x16
Param1	Mode	1	Controls the mechanism of the internal random number generator and seed update
Param2	Zero	2	Must be 0x0000
Data	NumIn	20,32	Input value from system

Table 8-26. Output Parameter

Name	Size	Notes
RandOut	1 or 32	The output of the random number generator or a single byte with a value of zero if Mode[0:1] is three

If Mode[0:1] is zero or one, the input NumIn parameter must be 20 bytes long, and the SHA-256 message body used to create the nonce stored internally in TempKey consists of the following:

32 bytes	RandOut
20 bytes	NumIn from input stream
1 byte	Opcode (always 0x16)
1 byte	Mode
1 byte	LSB of Param2 (should always be 0x00)

Upon completion of the command, TempKey.SourceFlag is set to “Rand.”

If Mode[0:1] is three, this command operates in pass-through mode, the input parameter (NumIn) must be 32 bytes long, and TempKey is loaded with NumIn. No SHA-256 calculation is performed, no data is returned to the system, and TempKey.SourceFlag is set to “Input.”

If Mode[0:1] is one, the automatic seed update is suppressed. See Section 3.4.2 for more details.

Table 8-27. Mode Encoding

Bits	Meaning
2-7	Must be zero
0-1	0: Combine new random number with NumIn, store in TempKey. Automatically update EEPROM seed only if necessary prior to random number generation. Recommended for highest security 1: Combine new random number with NumIn, store in TempKey. Generate random number using existing EEPROM seed, do NOT update EEPROM seed 2: Invalid 3: Operate in pass-through mode and write TempKey with NumIn

8.10 Pause Command

All chips on the bus for which the configuration Selector byte does *not* match the input selector parameter will go into the idle state. This command is used to prevent bus conflicts in a system that includes multiple ATSHA204 chips sharing the same bus.

This command differs from the idle flag/sequence in that individual chips on the single pin bus may be selected to go into the idle state, as opposed to the idle flag which causes all the CryptoAuthentication devices on the bus into the idle state.

If the EEPROM Selector byte does *not* match the input selector parameter, the chip will immediately go to the idle state and no result information will be available. If the input selector parameter does match the configuration selector byte, the chip returns a success code of 0x00.

The pause command cannot be used to put the chips into the sleep state.

Table 8-28. Input Parameters

	Name	Size	Notes
Opode	PAUSE	1	0x01
Param1	Selector	1	All chips that do not match this value go to idle state
Param2	Zero	2	Must be 0x0000
Data	-	0	-

Table 8-29. Output Parameter

Name	Size	Notes
Success	1	If the command indicates that some other chip should idle, the Atmel ATSHA204 returns a value of 0x00 If this chip goes to idle, no value is returned

8.11 Random Command

This command generates a random number for use by the system.

Random numbers are generated through a combination of the output of a hardware random number generator and an internal seed value stored in the EEPROM or SRAM. The external system may choose to update the internally stored EEPROM seed value prior to the generation of the random number as part of the execution of the nonce or random command, though the endurance limitations of the EEPROM limit the number of times that this update can be performed. After the endurance limit has been reached, attempts to update the EEPROM seed return an error.

The random command does not provide a mechanism to integrate an input number with the internal stored seed. If this functionality is desired, the system should use the Nonce command and ignore the generated nonce.

Prior to the configuration section being locked, the random number generator produces a value of 0xFF, 0xFF, 0x00, 0x00, 0xFF, 0xFF, 0x00, 0x00 to facilitate testing.

Note: The same internally stored seeds are used for both the Nonce and Random commands. Use of Mode=0 ensures that the EEPROM is updated, if necessary.

Table 8-30. Input Parameters

	Name	Size	Notes
Opode	RANDOM	1	0x1B
Param1	Mode	1	Controls the mechanism of the internal random number generator and seed update
Param2	Zero	2	Must be 0x0000
Data	-	0	-

Table 8-31. Output Parameter

Name	Size	Notes
RandOut	32	The output of the random number generator

Table 8-32. Mode Encoding

Bits	Meaning
1-7	Must be zero
0	0: Automatically update EEPROM seed only if necessary prior to random number generation Recommended for highest security 1: Generate random number using existing EEPROM seed; do *not* update EEPROM seed

8.12 Read Command

Reads words (one 4-byte word or an 8-word block of 32 bytes) from one of the memory zones of the chip. The data may optionally be encrypted before being returned to the system. See also Section 8.1.5, “Zone Encoding,” and Section 8.1.4, “Address Encoding,” for data zone byte and word addressing information.

If reading from a slot in which SlotConfig.EncryptRead is set, the GenDig command must have been run prior to the execution of this command to generate the key that will be used for encryption. The input nonce to GenDig must have been a random number, and the key specified in SlotConfig.ReadKey must have been used in the GenDig calculation.

The chip encrypts data to be read by XORing each byte read from the EEPROM with the corresponding byte from TempKey. Encrypted reads of the configuration and/or OTP zones are not permitted.

The byte addresses to be read should be divided by four (drop the least-significant two bits) before being passed to the chip. If 32 bytes are being read, the least-significant three bits of the input address are ignored. Addresses beyond the end of the specified zone result in an error.

The following restrictions apply to the three zones:

Config The words within this zone are always readable using this command, regardless of the value of LockConfig. See Section 2.1.1, as some bytes are unreadable under any circumstances, and any attempt to read these bytes result in an error.

OTP If the OTP zone is unlocked, this command returns an error. Once locked, if OTPmode is set to a non-zero value and the address points to either word zero or one, then the command also returns an error. Otherwise, the corresponding word within the OTP zone is returned in the clear. If OTPmode is Legacy, then only four byte reads are permitted.

Data If the data zone is unlocked, this command returns an error. Otherwise, the values within the corresponding SlotConfig word control access to the data slot. If SlotConfig.IsSecret is set and a four byte read is attempted, the chip returns an error. If EncryptRead is set, this command encrypts the data as specified above. If IsSecret is set and EncryptRead is clear, this command returns an error. If IsSecret is clear and EncryptRead is clear, this command returns the desired slot in the clear.

Table 8-33. Input Parameters

	Name	Size	Notes
Opcode	READ	1	0x02
Param1	Zone	1	Bits 0 and 1: Select among config, OTP, or data. See Section 8.1.5 Bits 2-6: Must be zero Bit 7: If one, 32 bytes are read; otherwise four bytes are read. Must be zero if reading from OTP zone
Param2	Address	2	Address of first word to be read within the zone. See Section 8.1.4
Data	-	0	-

Table 8-34. Output Parameter

Name	Size	Notes
Contents	4 or 32	The contents of the specified memory location

If reading the data zone and the EncryptRead bit is set in the corresponding SlotConfig word, the following actions are taken to encrypt the data:

All of the TempKey register bits must be properly set as follows, or this command returns an error:

TempKey.Valid == 1

TempKey.GenData == 1

TempKey.KeyID == SlotConfig.ReadKey

TempKey.SourceFlag == “Rand”

XOR the data from the memory zone with TempKey. Return as “Contents.”

8.13 UpdateExtra Command

This command is used to update the values of the two “extra” bytes within the configuration zone (location 84 and 85) after the configuration zone has been locked.

If the mode parameter indicates UserExtra at address 84:

If the current value in UserExtra (byte 84 of configuration zone) is zero, then UpdateExtra writes this byte with the LS byte of NewValue and returns success.

If the current value in UserExtra is non-zero, the command returns an execution error.

If the mode parameter indicates selector at address 85:

If SelectorMode (byte 19 of the configuration zone) is non-zero and Selector (byte 85 of the configuration zone) is zero, this command will write Selector with the LS byte of NewValue and return success. Once written to a non-zero value, it is then locked against further updating.

If SelectorMode has a value of zero, indicating that no check of the current Selector should be made, this command always updates Selector and always succeeds.

Table 8-35. Input Parameters

	Name	Size	Notes
Opode	UPDATEEXTRA	1	0x20
Param1	Mode	1	Bit 0: If zero, update config byte 84. If one, update config byte 85. Bits1-7: Must be zero
Param2	NewValue	2	LSB: Value to optionally be written to location 84 or 85 in configuration zone MSB: Must be 0x00
Data	-	0	-

Table 8-36. Output Parameter

Name	Size	Notes
Success	1	If the memory byte was updated, this command returns a value of 0x00 Otherwise, it returns an Execution error

8.14 Write Command

Writes either a one 4-byte word or an 8-word block of 32 bytes to one of the EEPROM zones on the chip. Depending on the value of the WriteConfig byte for this slot the data may be required to be encrypted by the system prior to being sent to the chip.

The following restrictions apply to writes within zones using this command:

Config	If the Config zone is locked or Zone:6 is set, this command returns an error. Otherwise the bytes are written as requested. Any attempt to write any byte for which Writes are permanently prohibited (per Section 2.1.1) results in a command error with no modifications to the EEPROM.
OTP	If the OTP zone is unlocked, all bytes can be written with this command. If the OTP zone is locked and the OTPmode byte is read-only or legacy, then this command returns an error. Otherwise, OTP mode should be consumption and this command sets to zero those bits in the OTP zone that correspond to the zero bits in the input parameter value. When the OTP zone is locked, encrypted writes to it are never permitted regardless of OTPmode.
Data	If the data zone is unlocked, all bytes in all zones can be written with either plain text or encrypted data. After the data zone is locked, the values within the WriteConfig bytes control access to the data slots. If the WriteConfig bits for this slot are set to “always”, the input data should be passed to the chip in the clear. If Bit:14 of SlotConfig is set to one, the input data should be encrypted and an input MAC calculated.

Four byte writes are only permitted in the data and OTP zones if all four of the following conditions are met:

- SlotConfig.IsSecret must be zero
- SlotConfig.WriteConfig must be “always”
- The input data must not encrypted
- The data/OTP zones must be locked

Four byte writes will return an error under all other circumstances.

The least significant three bits of Param2, Address[0:2], indicate the word within the block, or are ignored if an entire 32 byte block is being written. Address[3:6] contains the slot number for writes to the data zone, or the block number for the Config and OTP zones. Address values beyond the size of the specified zone result in the command returning an error.

Any attempt to write the OTP and/or data zones prior to the configuration section being locked results in the chip returning an error code.

8.14.1 Input Data Encryption

The input data may be encrypted to prevent snooping on the bus during personalization or system operation. The system should encrypt the data by XOR’ing the plain text with the current value in TempKey. Upon receipt the chip will XOR the input data with TempKey to restore the plain text prior to writing to the EEPROM.

Whenever the input data is encrypted an authorizing input MAC is always required when writing the data zone. This MAC is computed as:

SHA-256(TempKey, Opcode, Param1, Param2, SN[8], SN[0:1], <25 bytes of 0’s>, PlainTextData)

Prior to locking of the OTP/Data zones, Zone:6 is used to indicate to the chip whether or not the input data is encrypted. After locking of the OTP/Data zones, Zone:6 is ignored and only bit 14 of the slotConfig corresponding to the slot being written is used to determine whether or not the input data is encrypted.

If data encryption is indicated, TempKey must be valid prior to this command being called, it must be the result of GenDig. Specifically, this means that TempKey.Valid and TempKey.GenDig must both be set to one. Prior to data locking, any key can be used to generate TempKey. After locking, the last slot used by GenDig for TempKey creation and stored in TempKey.KeyID must match that in SlotConfig.WriteKey and the random number generator must have been used to originally generate TempKey prior to GenDig.

Table 8-37. Input Parameters

	Name	Size	Notes
Opcode	Write	1	0x12
Param1	Zone	1	Bits 0 and 1: Select among config, OTP or data. See Section 8.1.5 Bits 2-5: Must be zero Bit 6: If one, the input data must be encrypted. Must be zero if data/OTP zones are locked Bit 7: If one, 32 bytes will be written; otherwise, four bytes are written
Param2	Address	2	Address of first word to be written within the zone. See Section 8.1.4
Data_1	Value	4 or 32	Information to be written to the zone; may be encrypted
Data_2	Mac	0 or 32	Message authentication code to validate address and data Ignored if zone is unlocked

Table 8-38. Output parameter

Name	Size	Notes
Success	1	Upon successful completion, the Atmel ATSHA204 returns a value of zero

9. Compatibility

The ATSHA204 is designed to be upwards compatible with the AT88SA102S for field operation. Most systems designed to use the AT88SA102S in client devices will work perfectly with the ATSHA204 in the client devices without any modification to the host system software or hardware.

Host systems that utilize the AT88SA10HS host device will also interoperate properly with the ATSHA204 client device in place of a previously used AT88SA102S client. However, the AT88SA10HS itself cannot be replaced with the ATSHA204 without software modifications. With the appropriate software updates, the ATSHA204 can implement all the functions of an AT8810HS host chip and continue to properly communicate with client AT88SA102S chips.

For compatibility with the AT88SA102S, the following values should be written to the memory of the ATSHA204:

1. During configuration, OTPmode should be set to Legacy to hide the values of the first 64 bits of the OTP section, which contain a secret in the Atmel AT88SA102S
2. The same secret and status information that would have been written to the first 88 fuse bits of the Atmel AT88SA102S should be written to the first 88 bits of the OTP section on the Atmel ATSHA204
3. OTP bits 88 through 127 should be written with copies of the values stored in SN[4:8] within the configuration section of the Atmel ATSHA204 chip. The read command on legacy systems will always use the values in the OTP section while the Atmel ATSHA204 always uses the values in the configuration zone during the computation of cryptographic results
4. The key slot identified by the least significant four bits of the Atmel AT88SA102S KeyID assigned to a particular customer should be loaded with the Atmel-provided value for that key
5. The SlotConfig bits for the key slot identified in step four should be set to: CheckOnly=0, SingleUse=0, EncryptRead=0, IsSecret=1, WriteConfig=1000

The following compatibility exceptions apply:

- Those Atmel AT88SA102S systems using the BurnFuse command on the client device cannot be replaced with the Atmel ATSHA204, as the corresponding command is not available on the Atmel ATSHA204. The same capability is implemented with the Write command, but system software modifications are necessary
- Those Atmel AT88SA102S systems in which the system software reads and depends on a fixed value for the chip revision number (RevNum at ROM address one) will find a different value in the Atmel ATSHA204

  Note: This value is not guaranteed to be identical for all Atmel AT88SA102S chips
- Systems including multiple Atmel AT88SA102S and/or Atmel AT88SA10HS chips on a shared single-wire bus cannot be replaced with the Atmel ATSHA204, as the Pause command operates differently
- The key diversification strategy implemented by the Atmel ATSHA204 (when operating as a host) is different from the similar strategy used by the Atmel AT88SA10HS. The Atmel ATSHA204 can be used as a host authentication device for Atmel ATSHA204 clients that include diversified keys, but those clients will not work interchangeably with Atmel AT88SA102S clients
- Because of the difference in the nonvolatile memory technology and size, the secure personalization mechanism is different on the Atmel ATSHA204 as compared to the Atmel AT88SA10HS and Atmel AT88SA102S. Users will need to modify their manufacturing processes and procedures accordingly
- The Atmel ATSHA204 cannot replace a client Atmel AT88SA100S chip used for batteries and other self-powered systems

10. Mechanical

10.1 Pin-out

The chip is offered in multiple packages: 3-lead SOT23, 8-lead SOIC, 8-lead TSSOP and 8-pad UDFN. The pin-outs are as follows:

Table 10-1. Package Pinouts

Name	SOT23-3 LD	SOIC-8LD, TSSOP-8LD, UDFN-8 pad
SDA	1	5
SCL	–	6
V_{CC}	2	8
GND	3	4
NC	–	1, 2, 3, 7

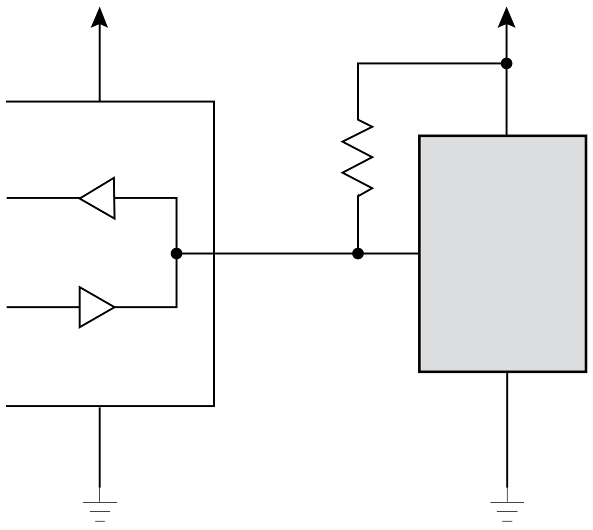
10.2 Wiring Configuration for Single-wire Interface

Using the single-wire interface allows the connection of the ATSHA204 to a host using only a single pin (SDA) 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 chip to the system.

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.

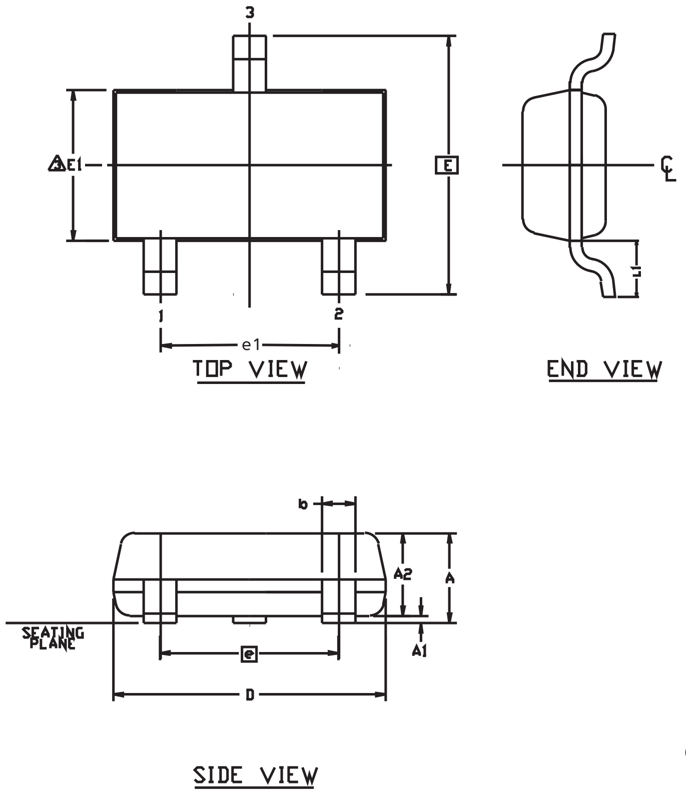
If the signal levels for SDA are different from the V_{CC} voltage, consult the parametric specifications section of this document to ensure that the signal levels are such that excessive leakage current will be minimized when in sleep modes. This situation might occur if the ATSHA204 chip is physically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

Figure 10-1. Three-wire Configuration for Single-Wire Interface



11. Package Drawings

3TS1 – SOT23



1. Dimension D does not include mold flash, protrusions or gate burrs. Mold flash, protrusions or gate burrs shall not exceed 0.25mm per end. Dimension E1 does not include interlead flash or protrusion. Interlead flash or protrusion shall not exceed 0.25mm per side.
2. The package top may be smaller than the package bottom. Dimensions D and E1 are determined at the outermost extremes of the plastic body exclusive of mold flash, tie bar burrs, gate burrs and interlead flash, but including any mismatch between the top and bottom of the plastic body.
3. These dimensions apply to the flat section of the lead between 0.08mm and 0.15mm from the lead tip.

COMMON DIMENSIONS
(Unit of Measure = mm)

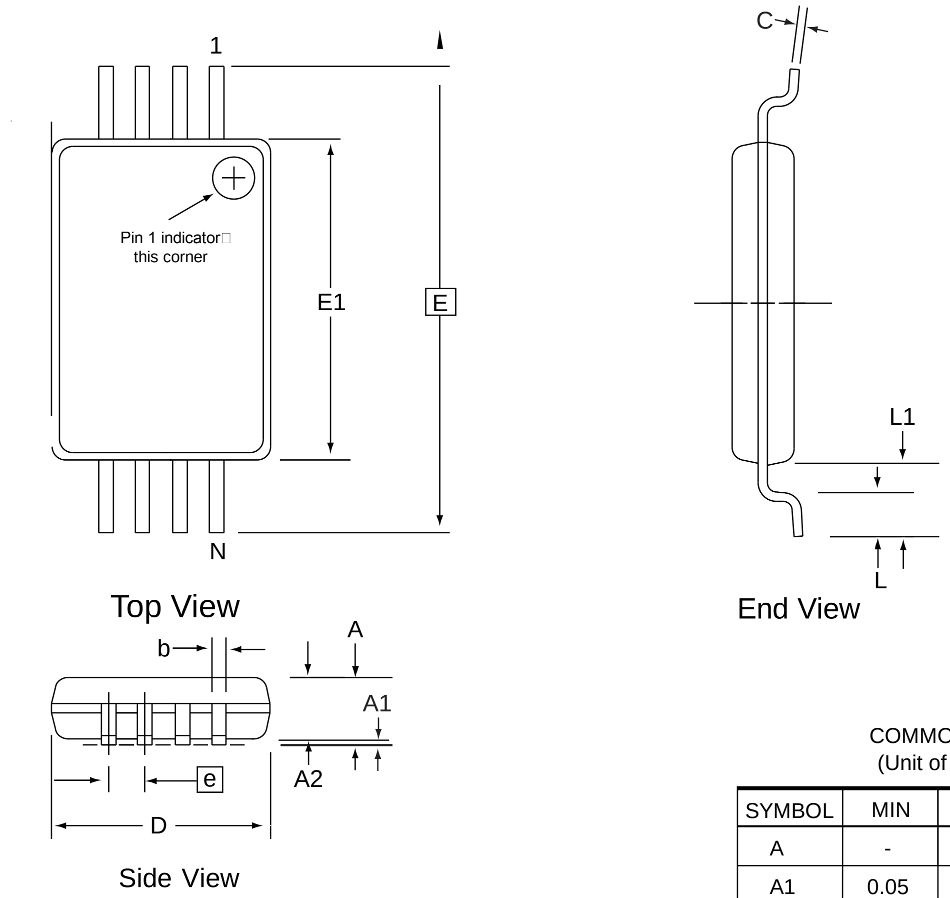
SYMBOL	MIN	NOM	MAX	NOTE
A	0.89	-	1.12	
A_1	0.01	-	0.10	
A_2	0.88	-	1.02	
D	2.80	2.90	3.04	1,2
E	2.10	-	2.64	
E_1	1.20	1.30	1.40	1,2
L_1	0.54 REF			
e1	1.90 BSC			
b	0.30	-	0.50	3

This drawing is for general information only. Refer to JEDEC Drawing TO-236, Variation AB for additional information.

11/5/08

Package Drawing Contact: packagedrawings@atmel.com	TITLE 3TS1, 3-lead, 1.30mm Body, Plastic Thin Shrink Small Outline Package (Shrink SOT)	GPC TBG	DRAWING NO. 3TS1	REV. A

8X – TSSOP



Notes: 1. This drawing is for general information only. Refer to JEDEC Drawing MO-153, Variation AA, for proper dimensions, tolerances, datums, etc.
2. Dimension D does not include mold Flash, protrusions or gate burrs. Mold Flash, protrusions and gate burrs shall not exceed 0.15 mm (0.006 in) per side.
3. Dimension E1 does not include inter-lead Flash or protrusions. Inter-lead Flash and protrusions shall not exceed 0.25 mm (0.010 in) per side.
4. Dimension b does not include Dambar protrusion. Allowable Dambar protrusion shall be 0.08 mm total in excess of the b dimension at maximum material condition. Dambar cannot be located on the lower radius of the foot. Minimum space between protrusion and adjacent lead is 0.07 mm.
5. Dimension D and E1 to be determined at Datum Plane H.

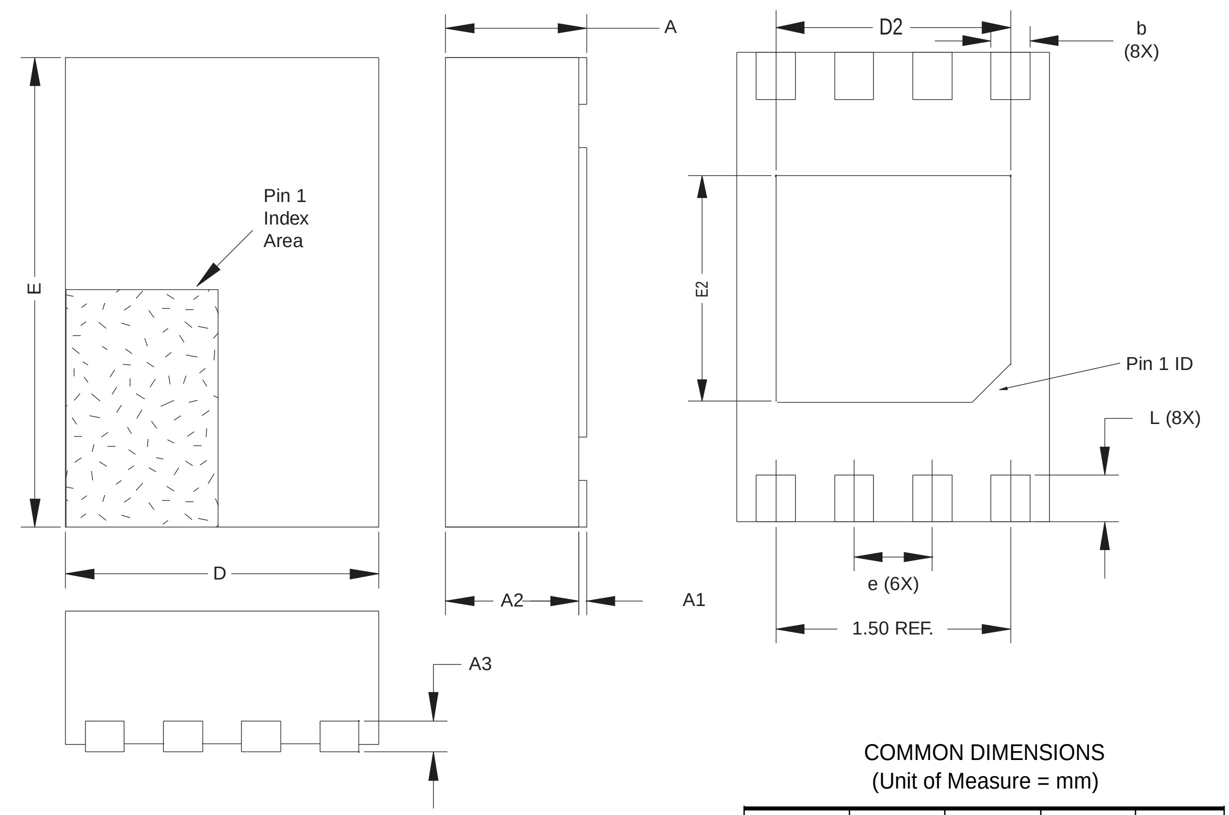
COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	-	-	1.20	
A1	0.05	-	0.15	
A2	0.80	1.00	1.05	
D	2.90	3.00	3.10	2, 5
E			6.40 BSC	
E1	4.30	4.40	4.50	3, 5
b	0.19	–	0.30	4
e		0.65 BSC		
L	0.45	0.60	0.75	
L1		1.00 REF		
C		0.09	-	0.20

6/22/11

	TITLE	GPC	DRAWING NO.	REV.
Package Drawing Contact: packagedrawings@atmel.com	8X, 8-lead 4.4mm Body, Plastic Thin Shrink Small Outline Package (TSSOP)	TNR	8X	D

8Y6 — UDFN



COMMON DIMENSIONS
(Unit of Measure = mm)

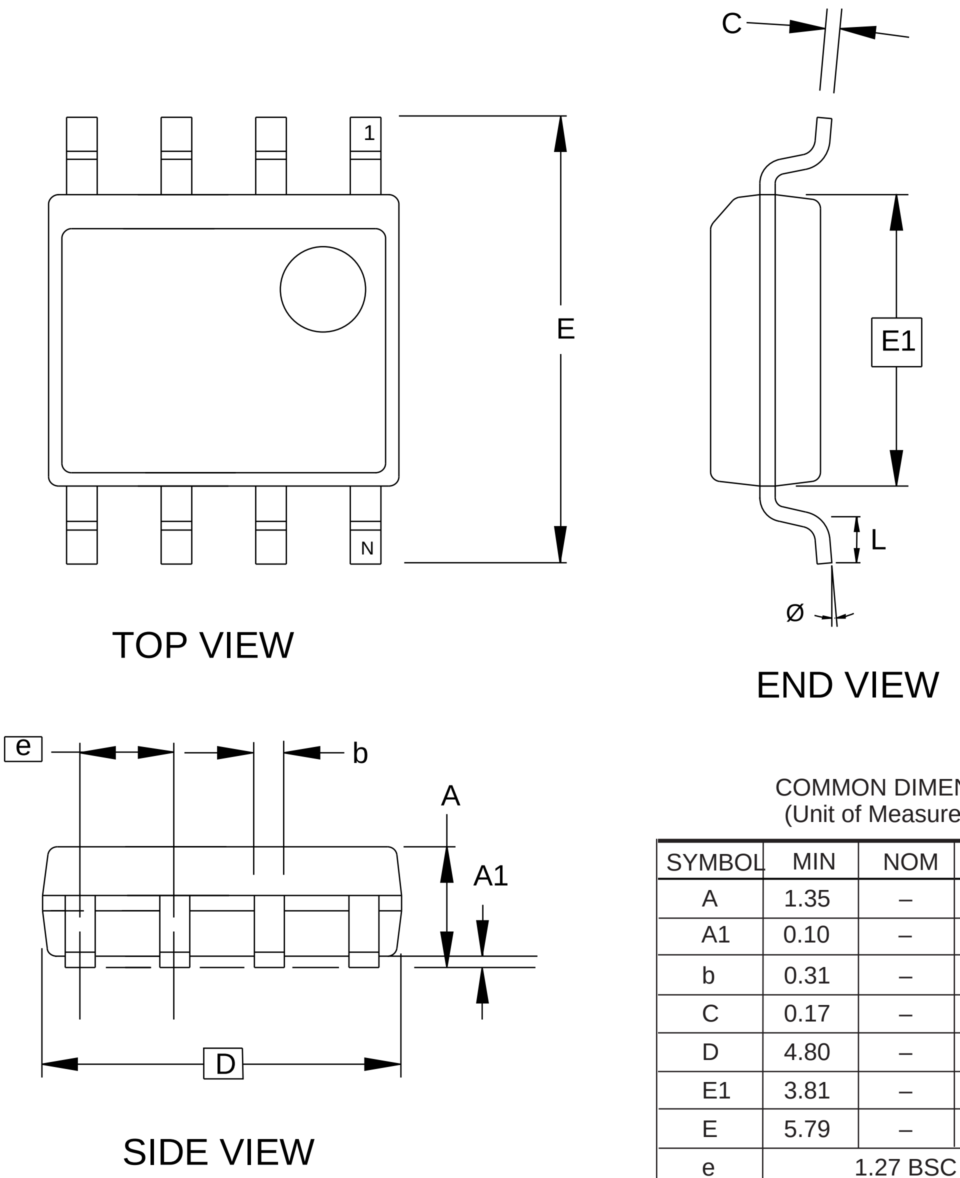
SYMBOL	MIN	NOM	MAX	NOTE
D		2.00 BSC		
E		3.00 BSC		
D2	1.40	1.50	1.60	
E2	–	–	1.40	
A	–	–	0.60	
A1	0.00	0.02	0.05	
A2	–	–	0.55	
A3		0.20 REF		
L	0.20	0.30	0.40	
e		0.50 BSC		
b	0.20	0.25	0.30	2

Notes:
1. This drawing is for general information only. Refer to JEDEC Drawing MO-229, for proper dimensions, tolerances, datums, etc.
2. Dimension b applies to metallized terminal and is measured between 0.15mm and 0.30mm from the terminal tip. If the terminal has the optional radius on the other end of the terminal, the dimension should not be measured in that radius area.
3. Soldering the large thermal pad is optional, but not recommended. No electrical connection is accomplished to the device through this pad, so if soldered it should be tied to ground

11/21/08

	TITLE	GPC	DRAWING NO.	REV.
Package Drawing Contact: packagedrawings@atmel.com	8Y6, 8-lead, 2.0x3.0mm Body, 0.50mm Pitch, UltraThin Mini-MAP, Dual No Lead Package (Sawn)(UDFN)	YNZ	8Y6	E

8S1 — SOIC



Notes: This drawing is for general information only. Refer to JEDEC Drawing MS-012, Variation AA for proper dimensions, tolerances, datums, etc.

COMMON DIMENSIONS
(Unit of Measure = mm)

SYMBOL	MIN	NOM	MAX	NOTE
A	1.35	–	1.75	
A1	0.10	–	0.25	
b	0.31	–	0.51	
C	0.17	–	0.25	
D	4.80	–	5.05	
E1	3.81	–	3.99	
E	5.79	–	6.20	
e	1.27 BSC			
L	0.40	–	1.27	
Ø	0°	–	8°	

6/22/11

	TITLE	GPC	DRAWING NO.	REV.
Package Drawing Contact: packagedrawings@atmel.com	8S1, 8-lead (0.150" Wide Body), Plastic Gull Wing Small Outline (JEDEC SOIC)	SWB	8S1	G

12. Ordering Information

Atmel ordering code	Package type	Interface configuration
ATSHA204-SH-CZ-T	SOIC, tape and reel	Single-wire
ATSHA204-SH-DA-T	SOIC, tape and reel	I^2C
ATSHA204-SH-DA-B	SOIC, bulk in tubes	I^2C
ATSHA204-TH-CZ-T	TSSOP, tape and reel	Single-wire
ATSHA204-TH-DA-T	TSSOP, tape and reel	I^2C
ATSHA204-TSU-T	SOT3LD, tape and reel	Single-wire
ATSHA204-MAH-CZ-T(1)	UDFN 8LD, tape and reel	Single-wire
ATSHA204-MAH-DA-T(1)	UDFN 8LD, tape and reel	I^2C

Note: 1. Contact Atmel for availability

13. Revision History

Doc. Rev.	Date	Comments
8740C	07/2011	Table 8-4, Command Opcodes, Short Descriptions, and Execution Times - Change UpdateExtra command for Typ from 4 to 8 and Max from 6 to 12 Change Mode:6 to Zone:6 Edit/update Write Command section Update template
8740B	04/2011	Document update
8740A	03/2011	Initial document release

14. Errata

The design should ensure that all IO pin levels are within the datasheet limits of V_{SS}-0.5V and V_{CC}+0.5V. The same power supply signal net should be used for both the IO driver, the V_{CC} pin on the SHA204 and any pull-up resistor on the SDA pin. Failure to adhere to these requirements may result in increased susceptibility to latchup.

Atmel Corporation
2325 Orchard Parkway
San Jose, CA 95131
USA
Tel: (+1)(408) 441-0311
Fax: (+1)(408) 487-2600
www.atmel.com

Atmel Asia Limited
Unit 01-5 & 16, 19F
BEA Tower, Millennium City 5
418 Kwun Tong Road
Kwun Tong, Kowloon
HONG KONG
Tel: (+852) 2245-6100
Fax: (+852) 2722-1369

Atmel Munich GmbH
Business Campus
Parkring 4
D-85748 Garching b. Munich
GERMANY
Tel: (+49) 89-31970-0
Fax: (+49) 89-3194621

Atmel Japan
9F, Tonetsu Shinkawa Bldg.
1-24-8 Shinkawa
Chuo-ku, Tokyo 104-0033
JAPAN
Tel: (+81)(3) 3523-3551
Fax: (+81)(3) 3523-7581