The Atmel ATAES132 is a high-security, Serial Electrically-Erasable and Programmable Read-Only Memory (EEPROM) providing both authentication and confidential nonvolatile data storage capabilities. Access restrictions for the 16 user zones are independently configured and any key can be used with any zone. Keys can also be used for standalone authentication. This flexibility permits the ATAES132 to be used in a wide range of applications.

The AES-128 cryptographic engine operates in AES-CCM mode to provide authentication, stored data encryption/decryption, and Message Authentication Codes. Data encryption/decryption can be performed for internally stored data or for small external data packets, depending upon the configuration. Data encrypted by one ATAES132 device can be decrypted by another, and vice versa.
The ATAES132 pinout is compatible with standard SPI and I²C Serial EEPROMs to allow placement on existing PC boards. The SPI and I²C instruction sets are identical to Atmel Serial EEPROMs. The extended security functions are accessed by sending command packets to the ATAES132 using standard Write instructions and reading responses using standard Read instructions. The ATAES132 Secure Serial EEPROM architecture allows it to be inserted into existing applications.

The ATAES132 device incorporates multiple physical security mechanisms to prevent release of the internally stored secrets. Secure personalization features are provided to facilitate third-party product manufacturing.
<table>
<thead>
<tr>
<th>1. Introduction .................................................................</th>
<th>7</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1 Scope .................................................................</td>
<td>7</td>
</tr>
<tr>
<td>1.2 Conventions ..........................................................</td>
<td>7</td>
</tr>
<tr>
<td>1.3 Abbreviations ..........................................................</td>
<td>8</td>
</tr>
<tr>
<td>1.4 Communication ..........................................................</td>
<td>9</td>
</tr>
<tr>
<td>2. Memory .................................................................</td>
<td>11</td>
</tr>
<tr>
<td>2.1 User Memory ............................................................</td>
<td>11</td>
</tr>
<tr>
<td>2.2 Key Memory .............................................................</td>
<td>11</td>
</tr>
<tr>
<td>2.3 Configuration Memory ................................................</td>
<td>12</td>
</tr>
<tr>
<td>2.4 SRAM Memory ............................................................</td>
<td>12</td>
</tr>
<tr>
<td>3. Security Features .....................................................</td>
<td>15</td>
</tr>
<tr>
<td>3.1 Architecture ............................................................</td>
<td>15</td>
</tr>
<tr>
<td>3.2 Authentication ..........................................................</td>
<td>15</td>
</tr>
<tr>
<td>3.3 Encrypted Memory Read/Write .....................................</td>
<td>16</td>
</tr>
<tr>
<td>3.4 Data Encryption/Decryption .......................................</td>
<td>16</td>
</tr>
<tr>
<td>3.5 Keys .................................................................</td>
<td>16</td>
</tr>
<tr>
<td>3.6 Random Numbers ........................................................</td>
<td>17</td>
</tr>
<tr>
<td>4. Security Configuration Registers ...................................</td>
<td>19</td>
</tr>
<tr>
<td>4.1 User Zone Configuration ............................................</td>
<td>19</td>
</tr>
<tr>
<td>4.2 Key Configuration ....................................................</td>
<td>20</td>
</tr>
<tr>
<td>4.3 VolatileKey Configuration .........................................</td>
<td>22</td>
</tr>
<tr>
<td>4.4 Counter Configuration ...............................................</td>
<td>23</td>
</tr>
<tr>
<td>5. Standard Serial EEPROM Read and Write Commands ............</td>
<td>24</td>
</tr>
<tr>
<td>5.1 Read .................................................................</td>
<td>24</td>
</tr>
<tr>
<td>5.2 Write .................................................................</td>
<td>25</td>
</tr>
<tr>
<td>6. Commands .................................................................</td>
<td>27</td>
</tr>
<tr>
<td>6.1 Command Block and Packet ........................................</td>
<td>27</td>
</tr>
<tr>
<td>6.2 Command Summary ....................................................</td>
<td>28</td>
</tr>
<tr>
<td>6.3 ReturnCode .............................................................</td>
<td>30</td>
</tr>
<tr>
<td>7. Command Definitions ..................................................</td>
<td>31</td>
</tr>
<tr>
<td>7.1 Auth Command ..........................................................</td>
<td>31</td>
</tr>
<tr>
<td>7.2 AuthCheck Command ...................................................</td>
<td>34</td>
</tr>
<tr>
<td>7.3 AuthCompute Command ...............................................</td>
<td>35</td>
</tr>
<tr>
<td>7.4 BlockRead Command ...................................................</td>
<td>36</td>
</tr>
<tr>
<td>7.5 Counter Command .......................................................</td>
<td>37</td>
</tr>
<tr>
<td>7.6 Crunch Command ........................................................</td>
<td>39</td>
</tr>
<tr>
<td>7.7 DecRead Command .......................................................</td>
<td>40</td>
</tr>
<tr>
<td>7.8 Decrypt Command .......................................................</td>
<td>41</td>
</tr>
<tr>
<td>7.9 EncRead Command .......................................................</td>
<td>43</td>
</tr>
<tr>
<td>7.10 Encrypt Command ......................................................</td>
<td>43</td>
</tr>
<tr>
<td>7.11 EncWrite Command ....................................................</td>
<td>46</td>
</tr>
<tr>
<td>7.12 INFO Command ........................................................</td>
<td>48</td>
</tr>
<tr>
<td>7.13 KeyCreate Command ..................................................</td>
<td>49</td>
</tr>
<tr>
<td>7.14 KeyImport Command ..................................................</td>
<td>51</td>
</tr>
<tr>
<td>7.15 KeyLoad Command .....................................................</td>
<td>53</td>
</tr>
<tr>
<td>7.16 KeyTransfer Command ...............................................</td>
<td>54</td>
</tr>
<tr>
<td>7.17 Legacy Command .......................................................</td>
<td>56</td>
</tr>
<tr>
<td>Appendix</td>
<td>Section</td>
</tr>
<tr>
<td>----------</td>
<td>--------------------------------------------------------------</td>
</tr>
<tr>
<td>J</td>
<td>I²C Interface</td>
</tr>
<tr>
<td>J.1</td>
<td>I²C Serial Interface Description</td>
</tr>
<tr>
<td>J.2</td>
<td>Pin Descriptions</td>
</tr>
<tr>
<td>J.3</td>
<td>I²C Instruction Set</td>
</tr>
<tr>
<td>J.4</td>
<td>I²C Interface Synchronization Procedure</td>
</tr>
<tr>
<td>J.5</td>
<td>I²C Auth Signaling</td>
</tr>
<tr>
<td>J.6</td>
<td>I²C Compatibility</td>
</tr>
<tr>
<td>J.7</td>
<td>Timing Diagram</td>
</tr>
<tr>
<td>K</td>
<td>SPI Interface</td>
</tr>
<tr>
<td>K.1</td>
<td>SPI Serial Interface Description</td>
</tr>
<tr>
<td>K.2</td>
<td>SPI Communication Mode Pin Descriptions</td>
</tr>
<tr>
<td>K.3</td>
<td>SPI Instruction Set</td>
</tr>
<tr>
<td>K.4</td>
<td>Timing Diagram</td>
</tr>
<tr>
<td>L</td>
<td>Power Management</td>
</tr>
<tr>
<td>L.1</td>
<td>Power State Descriptions</td>
</tr>
<tr>
<td>L.2</td>
<td>Power State Transitions</td>
</tr>
<tr>
<td>L.3</td>
<td>Understanding the ChipState Register</td>
</tr>
<tr>
<td>M</td>
<td>Block Checksum</td>
</tr>
<tr>
<td>M.1</td>
<td>Checksum Function</td>
</tr>
<tr>
<td>M.2</td>
<td>Checksum Examples</td>
</tr>
<tr>
<td>N</td>
<td>ATAES132 Command Response Time</td>
</tr>
<tr>
<td>O</td>
<td>Default Configuration</td>
</tr>
<tr>
<td>O.1</td>
<td>Configuration Memory Contents</td>
</tr>
<tr>
<td>O.2</td>
<td>Key Memory Contents</td>
</tr>
</tbody>
</table>

The page contains information related to various command sets and interfaces, checksum examples, power management, and more detailed technical specifications.
Appendix P. Serial Memory Backward Compatibility ......................................................................................................................... 161
  P.1  i²C Serial EEPROM Compatibility ..................................................................................................................................................... 161
  P.2  SPI Serial EEPROM Compatibility .................................................................................................................................................... 162

Appendix Q. Ordering Information .......................................................................................................................................................... 166
  Q.1  Atmel Ordering Codes ........................................................................................................................................................................ 166
  Q.2  Mechanical Information ....................................................................................................................................................................... 167

Appendix R. Errata ..................................................................................................................................................................................... 169
  R.1  KeyCreate Command Executed with Usage Counter ......................................................................................................................... 169

Appendix S. Revision History ................................................................................................................................................................. 170
1. Introduction
ATAES132 is the first device in a family of high-security Serial EEPROMs using the Advanced Encryption
Standard (AES) cryptographic algorithm. ATAES132 provides 32Kb of EEPROM user data memory, sixteen
128-bit Key Registers, sixteen high-endurance monotonic EEPROM Counters, factory unique Die Identification
Numbers, and a Configuration Memory. The Configuration Memory registers control access to the User Memory,
as well as the restrictions on Key and Counter functionality.

The User Memory can be accessed directly with standard SPI or i²C commands if a user zone is configured for
open or read-only access. If the user zone security is activated, then the extended ATAES132 command set is
used to access the contents of a user zone. The extended ATAES132 commands are executed by writing the
command packet to the virtual memory using standard SPI or i²C Write commands. The response packet is
retrieved by reading it from the virtual memory using standard SPI or i²C Read commands.

The ATAES132 packages are compatible with standard SPI and i²C EEPROM footprints. This allows the
ATAES132 to be inserted into many existing Serial EEPROM applications.

1.1 Scope
This Specification provides all specifications for configuration and operation of the ATAES132.

1.2 Conventions

<table>
<thead>
<tr>
<th>Nomenclature</th>
<th>Definition</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Host</td>
<td>The SPI or i²C Master Device</td>
<td>The Host initiates all communications with slave devices on the serial interface bus.</td>
</tr>
<tr>
<td>Client</td>
<td>The ATAES132 Secure Serial EEPROM Defined by this Specification</td>
<td>Operates as a SPI or i²C slave.</td>
</tr>
<tr>
<td>nnb</td>
<td>Binary Number</td>
<td>Denotes a binary number “nn” (most-significant bit on the left).</td>
</tr>
<tr>
<td>0xZZZZ</td>
<td>Hexadecimal Number</td>
<td>Denotes hex number ZZZZ (most-significant bit on the left).</td>
</tr>
<tr>
<td>ZZZZ_h</td>
<td>Hexadecimal Number</td>
<td>Denotes hex number ZZZZ (most-significant bit on the left).</td>
</tr>
<tr>
<td>RegName.FieldName</td>
<td>Field Name</td>
<td>Reference to bit field FieldName in register RegName.</td>
</tr>
<tr>
<td>RegArray[xx].FieldName</td>
<td>Field Name</td>
<td>Reference to bit field FieldName in register RegArray[xx], where xx is the array index.</td>
</tr>
<tr>
<td>UZ</td>
<td>User Zone</td>
<td>Reference to a User Zone number.</td>
</tr>
<tr>
<td>CntID</td>
<td>Counter ID</td>
<td>Reference to a Counter number.</td>
</tr>
<tr>
<td>KeyID</td>
<td>Key ID</td>
<td>Reference to a Key Register number.</td>
</tr>
</tbody>
</table>
1.2.1 Byte Order

The ATAES132 device uses a big-endian coding scheme and utilizes the same bit and byte orders as a standard Serial EEPROM. The byte order is identical to the NIST AES specifications (see Appendix A, Standards and Reference Documents):

- The most significant bit of each byte is transmitted first on the bus.
- The most significant byte of multi-byte integers is transmitted prior to the least significant byte. This applies to the CRC, address, and other 16-bit command parameters.
- All arrays are transmitted in index order, with byte index 0 first.
- Configuration fields that are more than eight bits appear on the bus during a Read or Write in the index order in which they appear in this specification. The top byte in the input parameters table is byte[0] and appears first on the bus. These fields are arrays of bytes, not multi-byte integers.

1.3 Abbreviations

Table 1-2. Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Phrase</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>AES</td>
<td>Advanced Encryption Standard</td>
<td>Block cipher algorithm standardized by NIST with 128-bit block size.</td>
</tr>
<tr>
<td>AES-CCM</td>
<td>AES Cipher Chaining Message</td>
<td>AES mode using the Counter with Cipher Block Chaining-Message Authentication Code Algorithm.</td>
</tr>
<tr>
<td>Ciphertext</td>
<td></td>
<td>Data communicated after it has been encrypted.</td>
</tr>
<tr>
<td>Cleartext</td>
<td></td>
<td>Data communicated in a nonencrypted state.</td>
</tr>
<tr>
<td>MAC</td>
<td>Message Authentication Code</td>
<td>A 128-bit value used to validate the authenticity of ciphertext.</td>
</tr>
<tr>
<td>Nonce</td>
<td>Number Used Once</td>
<td>A value used in cryptographic operations.</td>
</tr>
<tr>
<td>Plaintext</td>
<td></td>
<td>Data which is either the input to an encryption operation or the output of a decryption operation.</td>
</tr>
<tr>
<td>RFU</td>
<td>Reserved For Future Use</td>
<td>Any feature, memory location, or bit that is held as reserved for future use by Atmel.</td>
</tr>
<tr>
<td>RNG</td>
<td>Random Number Generator</td>
<td>Produces high-quality pseudo-random numbers.</td>
</tr>
</tbody>
</table>
1.4 Communication

The ATAES132 is designed to interface directly with SPI and I^2C microcontrollers. The Read and Write commands are similar to the standard Atmel Serial EEPROM commands for ease of use. Since the ATAES132 pinout is also similar to standard Serial EEPROMs, it is possible to use the ATAES132 on existing PC boards in some cases.

When Read and/or Write access to a user zone is unrestricted, the memory is accessed using the standard I^2C or SPI Read and Write commands. Similarly, if Authentication Only is required and the authentication requirement has been satisfied, then the memory is accessed directly by the Host using standard I^2C or SPI Read and Write commands.

If the Host begins a Read operation in an open user zone but continues reading until a prohibited section of memory is reached, the ATAES132 will continue to increment the address and will return 0xFF for each byte in the restricted user zone. If the Host begins a Read operation in an open user zone but continues reading beyond the end of the User Memory, the ATAES132 will return 0xFF for each byte requested, but will stop incrementing the address.

All other operations, including execution of the extended commands, are performed by using the standard I^2C or SPI Read and Write commands to exchange data packets via the command and response memory buffers. The Device Status Register reports the state of the device and is used for handshaking between the Host and the ATAES132.

1.4.1 Sending ATAES132 Commands

The ATAES132 commands described in Section 7, Command Definitions, are executed by writing the command block to virtual memory (Appendix D, Command Memory Map) using standard SPI or I^2C Write commands. The response block is retrieved by reading it from virtual memory using standard SPI or I^2C Read commands.

1.4.1.1 Command Memory Buffer

The Command Memory Buffer is a write-only memory buffer that is used by writing a command block to the buffer at the base address of 0xFE00. After the Host completes its Write operation to the buffer, the ATAES132 verifies the integrity of the block by checking the 16-bit checksum, and then executes the requested operation. See Section 6.1, Command Block and Packet for a description of the command packet. See Appendix D for additional Command Memory Buffer information.

<table>
<thead>
<tr>
<th>Base Address</th>
<th>Base + 1</th>
<th>Base + 2</th>
<th>Base + 3</th>
<th>......</th>
<th>......</th>
<th>......</th>
<th>......</th>
<th>Base + N-2</th>
<th>Base + N-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Opcode</td>
<td>Mode</td>
<td>Param1</td>
<td>Param1</td>
<td>Param2</td>
<td>......</td>
<td>DataX</td>
<td>CRC1</td>
<td>CRC2</td>
</tr>
</tbody>
</table>

1.4.1.2 Response Memory Buffer

The Response Memory Buffer is a read-only memory buffer that is used by reading a response from the buffer at the base address of 0xFE00. The base address of the Response Memory Buffer contains the first byte of the response packet after an ATAES132 command is processed. See Section 6.1 for a description of the response packet. See Appendix D for additional Response Memory Buffer information.

<table>
<thead>
<tr>
<th>Base Address</th>
<th>Base + 1</th>
<th>Base + 2</th>
<th>Base + 3</th>
<th>......</th>
<th>......</th>
<th>......</th>
<th>......</th>
<th>Base + N-2</th>
<th>Base + N-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>ReturnCode</td>
<td>Data1</td>
<td>Data2</td>
<td>Data3</td>
<td>......</td>
<td>......</td>
<td>DataX</td>
<td>CRC1</td>
<td>CRC2</td>
</tr>
</tbody>
</table>
The Response Memory Buffer is also used to report errors which occur during execution of standard I²C or SPI Write commands. When the I²C or SPI command execution is complete (as indicated by the STATUS Register), the Response Memory Buffer contains a block containing an error code (ReturnCode) if an error occurred; otherwise, it contains a block with ReturnCode = 0x00. See Section 6.3, ReturnCode, for the error descriptions.

1.4.2 Device Status Register (STATUS)

The Device Status Register is used for handshaking between the Host microcontroller and the ATAES132. The Host microcontroller is expected to read the STATUS Register before sending a command or reading a response. The read-only Device Status Register at address 0xFFF0 reports the current status of the ATAES132 device. This register can be read with the standard I²C or SPI Read Memory commands. The SPI Read Status Register command can also be used to read the STATUS Register, as described in Appendix K.3.5, Read Status Register Command (RDSR).

Reading the STATUS Register does not increment the Memory Read Address, and so a Host microcontroller can easily monitor the ATAES132 device status by repeatedly reading the STATUS Register. See Appendix G, Understanding the STATUS Register for a detailed description of the STATUS Register bits and Status Bit behavior.

Table 1-5. Device Status Register Definition

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EERR</td>
<td>RRDY</td>
<td>Reserved</td>
<td>CRCE</td>
<td>Reserved</td>
<td>WAKEb</td>
<td>WEN</td>
<td>WIP</td>
</tr>
</tbody>
</table>

The Device Status Register can always be read when the ATAES132 is configured for SPI interface mode, even if the ATAES132 is processing a command or writing the EEPROM. When the ATAES132 is configured for I²C interface mode, the Host can read the STATUS Register only when the I²C Device Address is ACKed.

If the ATAES132 is in the Sleep or Standby power state, reading the STATUS Register forces the ATAES132 to wake-up; the STATUS Register is 0xFF until the wake-up process is complete.
2. Memory

The ATAES132 EEPROM is a nonvolatile memory which is divided into several sections with each section having a different function. The User Memory section contains 32Kb for data storage. The Configuration Memory section contains the configuration information, security control registers, and counters. The Key Memory stores the 16 secret keys used to perform cryptographic functions. The EEPROM page length is 32 bytes. The ATAES132 SRAM buffers and registers are located near the top of the memory address space and are accessed using standard EEPROM Read/Write commands.

The complete memory map is shown in Appendix B, Memory Map. Each portion of the memory is described briefly in the following sections.

2.1 User Memory

The 32Kb User Memory is organized as 16 user zones of 2Kb each. Each user zone has an associated user zone configuration register in the Configuration Memory. A user zone can only be accessed when the security requirements specified in the associated user zone configuration register have been satisfied. All bytes within a user zone have the same access restrictions. Since the user zone access restrictions are independently configured, the security requirements for each user zone can be unique. Any key can be used with any user zone.

Each user zone can be configured to require authentication, Read Encryption, Write Encryption, a combination of these, or no security. The User Memory can be accessed directly with standard SPI or I2C commands if a user zone is configured for open or read-only access. If the user zone security is activated, then the extended ATAES132 command set is used to access the contents of a user zone.

2.1.1 Automatic Post Write Data Verification

The Write and EncWrite commands include an automatic data verification function. After the EEPROM Write is complete, the Data Verification Logic reads the new EEPROM contents and compares it to the data received from the Host. If the data does not match, the ATAES132 sets the EERR bit in the STATUS Register and returns a DataMatch error code. If the data is correct, then the ReturnCode indicates success.

2.2 Key Memory

The Key Memory securely stores 16 keys which are each 128 bits long. Each key has an associated Key Configuration Register in the Configuration Memory. Keys can only be used for the cryptographic functions enabled in the Key Configuration Register. Individual keys can be configured to require a successful authentication prior to use. Key values can never be read from the ATAES132 under any circumstances. See Appendix F, Key Memory Map.

Key Memory can be written prior to locking with either encrypted or cleartext data. Encrypted writes are performed using the EncWrite command (see Section 7.11, EncWrite Command). Cleartext writes are performed using the standard SPI or I2C Write commands (see Section 5.2, Write). After locking, the Key Registers are managed with the KeyCreate, KeyImport, KeyLoad, and KeyTransfer commands. The KeyTransfer command allows the User Memory to be used as the Extended Key Memory; eight keys can be stored in each user zone (see Section 7.16, KeyTransfer Command).
2.3 Configuration Memory

The Configuration Memory contains all of the registers which control user zone access requirements, the Key usage restrictions, and the Counter usage restrictions. Device-level Configuration Option Registers are also located in Configuration Memory.

The ATAES132 Configuration Memory includes a register programmed with unique, read-only die identification data at the factory. The Configuration Memory also contains several registers for customer information. The Configuration Memory registers can always be read using the BlockRead command (see Section 7.4, BlockRead Command). The Lock command is used to permanently lock the contents of the Configuration Memory after personalization (see Section 7.18, Lock Command).

See Table 2-1 for a summary of the Configuration Memory registers sorted by register name. See Appendix E for the Configuration Memory Map.

2.3.1 Counters

The ATAES132 includes 16 high-endurance EEPROM Counters. Each Counter has or can:

- An associated Counter Configuration Register in the Configuration Memory,
- Only be incremented,
- Never be decremented or reset,
- Be used to track system usage or to store small values.

A key can be configured to prevent exhaustive attacks by limiting key usage with a Counter.

Each counter can increment up to a value of 2,097,134 using the Count Command; after which they can no longer changed. Counters attached to keys are incremented each time the key is used; when the Usage Counter reaches its limit, the key is disabled. The Counters include a power interruption protection feature to prevent corruption of the Count value if power is removed during the increment operation.

On shipment from Atmel, the EEPROM locations are initialized to their lowest value. The initial value of each Counter may be written to a different value prior to personalization and prior to locking the configuration. See Appendix H, Understanding Counters.

2.4 SRAM Memory

The ATAES132 SRAM is used to store volatile data and status information. The ATAES132 SRAM buffers and registers are mapped into the top of the memory address space and are accessed using the standard EEPROM Read/Write commands.

- The Command Memory Buffer is used to send extended commands to the device.
- The Response Memory Buffer is used to read responses to the extended commands from the device.
- An IO Address Reset Register is used to reset the buffer address pointers.
- The STATUS Register reports the state of the device.
- The VolatileKey register and the Authentication Status Register are stored in the SRAM and are managed by the internal logic. These registers cannot be directly written or read by the user.

2.4.1 Nonce

The SRAM is used to store the Nonce and Random Number Generator (RNG) Seed. The RNG Seed is generated automatically by ATAES132, as described in Section 3.6, Random Numbers. The Nonce is generated using the Nonce command or the NonceCompute command. The Nonce and RNG Seed Register are erased when the device loses power, enters the Sleep state, or is reset.
2.4.2 VolatileKey

The SRAM contains a session key register named VolatileKey. This key location can be written with the KeyCreate, KeyImport, KeyLoad, or KeyTransfer commands. The VolatileKey register is erased when the device loses power, enters the Sleep state, or is reset. Restrictions on VolatileKey are established when the register is created/loaded and persist until the power is lost or the key is reloaded.

VolatileKey can never be used to read or write the User Memory or to increment the Counters. VolatileKey can only be used to perform authentication operations and to encrypt or decrypt external data. See Section 4.3, VolatileKey Configuration for the VolatileKey usage restrictions.

2.4.3 Command Memory Buffer

The Host executes extended ATAES132 commands by writing a command block to the Command Memory Buffer using a standard SPI or I²C Write command. After the Host completes its Write operation to the SRAM buffer, ATAES132 verifies the integrity of the block by checking the 16-bit Checksum and then executes the requested operation.

2.4.4 Response Memory Buffer

The Host receives responses to the extended ATAES132 commands by reading a response block from the Response Memory Buffer using a standard SPI or I²C Read command. The base address of the Response Memory Buffer contains the first byte of the response packet after an ATAES132 command is processed.

2.4.5 IO Address Reset Register

Writing the IO Address Reset Register causes the address pointers in the Command Memory Buffer and the Response Memory Buffer to be reset to the base address of the buffers. Writing the IO Address Reset Register does not alter the contents of the Response Memory Buffer or the value of the STATUS Register.

2.4.6 Device Status Register (STATUS)

The Device Status Register is used for handshaking between the Host microcontroller and ATAES132. The Host is expected to read the STATUS Register before sending a command or reading a response. Reading the STATUS Register does not alter the contents of the Command Memory Buffer, the Response Memory Buffer, or the value of the STATUS Register. See Appendix G, Understanding the STATUS Register.

2.4.7 Authentication Status Register

The ATAES132 Authentication Status Register stores the result of most recent authentication attempt. The Authentication Status Register contains the Authentication KeyID, the AuthComplete status flag, and the authentication usage restriction bits. Prior to executing the Auth command, the AuthComplete status flag is set to NoAuth. After successful Inbound Only or Mutual Authentication, the AuthComplete status flag is set to YesAuth.

Table 2-1. Summary of the Configuration Memory Registers Sorted by Register Name

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
<th>Write</th>
<th>Read</th>
<th>Bytes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Algorithm</td>
<td>Algorithm ID code (0x0000).</td>
<td>Never</td>
<td>Always</td>
<td>2</td>
</tr>
<tr>
<td>ChipConfig</td>
<td>Device-level cryptographic and power-up configuration options.</td>
<td>If LockConfig = Unlocked</td>
<td>Always</td>
<td>1</td>
</tr>
<tr>
<td>Counters</td>
<td>16 high-endurance counters, each capable of counting to 2M. See Appendix H, Understanding Counters.</td>
<td>If LockConfig = Unlocked</td>
<td>Always</td>
<td>128</td>
</tr>
<tr>
<td>CounterConfig</td>
<td>Configuration information for each counter. See Section 4.4, Counter Configuration.</td>
<td>If LockConfig = Unlocked</td>
<td>Always</td>
<td>32</td>
</tr>
<tr>
<td>Name</td>
<td>Description</td>
<td>Write</td>
<td>Read</td>
<td>Bytes</td>
</tr>
<tr>
<td>----------------</td>
<td>-----------------------------------------------------------------------------</td>
<td>---------------------</td>
<td>-----------</td>
<td>-------</td>
</tr>
<tr>
<td>DeviceNum</td>
<td>Atmel device number code.</td>
<td>Never</td>
<td>Always</td>
<td>1</td>
</tr>
<tr>
<td>EEPROMSize</td>
<td>Length in bytes of physical EEPROM page (32, 0x20).</td>
<td>Never</td>
<td>Always</td>
<td>1</td>
</tr>
<tr>
<td>EncReadSize</td>
<td>Maximum data length in bytes for EncRead (32, 0x20).</td>
<td>Never</td>
<td>Always</td>
<td>1</td>
</tr>
<tr>
<td>EncWriteSize</td>
<td>Maximum data length in bytes for EncWrite (32, 0x20).</td>
<td>Never</td>
<td>Always</td>
<td>1</td>
</tr>
<tr>
<td>FreeSpace</td>
<td>Free memory for customer data storage.</td>
<td>If LockConfig = Unlocked</td>
<td>Always</td>
<td>96</td>
</tr>
<tr>
<td>JEDEC</td>
<td>Atmel JEDEC manufacturer code (0x001F).</td>
<td>Never</td>
<td>Always</td>
<td>2</td>
</tr>
<tr>
<td>KeyConfig</td>
<td>Configuration information for each key. See Section 4.2, Key Configuration.</td>
<td>If LockConfig = Unlocked</td>
<td>Always</td>
<td>64</td>
</tr>
<tr>
<td>LockConfig</td>
<td>Controls Configuration Memory Write access, except SmallZone. Default is the Unlocked state. (2)</td>
<td>Via Lock Command Only</td>
<td>Always</td>
<td>1</td>
</tr>
<tr>
<td>LockKeys</td>
<td>Controls Key Memory Write access. Default is the Unlocked state. (2)</td>
<td>Via Lock Command Only</td>
<td>Always</td>
<td>1</td>
</tr>
<tr>
<td>LockSmall</td>
<td>Controls SmallZone Register Write access. Default is the Unlocked state. (2)</td>
<td>Via Lock Command Only</td>
<td>Always</td>
<td>1</td>
</tr>
<tr>
<td>LotHistory</td>
<td>Atmel proprietary manufacturing information.</td>
<td>Never</td>
<td>Always</td>
<td>8</td>
</tr>
<tr>
<td>ManufacturingID</td>
<td>Two byte manufacturing ID code.</td>
<td>Never</td>
<td>Always</td>
<td>2</td>
</tr>
<tr>
<td>PermConfig</td>
<td>Atmel factory device configuration options.</td>
<td>Never</td>
<td>Always</td>
<td>1</td>
</tr>
<tr>
<td>SerialNum</td>
<td>Guaranteed unique die serial number. SerialNum is optionally included in cryptographic calculations. See Appendix E.2.1, SerialNum Register.</td>
<td>Never</td>
<td>Always</td>
<td>8</td>
</tr>
<tr>
<td>SmallZone</td>
<td>32 byte value. The first four bytes are optionally included in cryptographic calculations. See Appendix E.2.23, SmallZone Register.</td>
<td>If LockSmall = Unlocked</td>
<td>Always</td>
<td>32</td>
</tr>
<tr>
<td>I2C Addr</td>
<td>Selects the serial interface mode and stores the I2C Device Address.</td>
<td>If LockConfig = Unlocked</td>
<td>Always</td>
<td>1</td>
</tr>
<tr>
<td>ZoneConfig</td>
<td>Access and usage permissions for each user zone. See Section 4.1, User Zone Configuration.</td>
<td>If LockConfig = Unlocked</td>
<td>Always</td>
<td>64</td>
</tr>
</tbody>
</table>

Notes:
1. Changes to most of the configuration registers take effect immediately which allows the functionality to be tested during the personalization process. Changes to the I2C Addr register take effect at the next Reset, Power Up, or Wake-Up from the Sleep state.
2. The LockConfig, LockKeys, and LockSmall bytes can only be changed with the Lock command (see Section 7.18.1, User Zone ReadOnly Activation). **Warning:** ATAES132 must always be locked by the customer prior to shipment to the end user to protect the customer secrets.
3. Security Features

All ATAES132 security features are optional. Each feature is enabled or disabled by programming configuration bits in the EEPROM Configuration Memory. Each user zone, Key, and Counter is separately and independently configured.

This section describes the ATAES132 security features and cryptographic capabilities. The functionality associated with each portion of the memory is described in Section 2, Memory.

3.1 Architecture

ATAES132 contains all circuitry for performing authentication, encryption, and decryption using keys stored securely in the internal EEPROM. Since the secrets are stored securely in the ATAES132, they do not have to be exchanged prior to executing cryptographic operations.

ATAES132 has fixed cryptographic functionality; it is not a microcontroller and cannot accept customer firmware. ATAES132 contains a hardware AES cryptographic engine and has a fixed command set. Although the functionality is fixed, it is also flexible because each feature is enabled or disabled by the customer by programming registers in the EEPROM Configuration Memory. After personalization is complete, fuses lock the configuration so that it cannot be changed.

3.1.1 AES

The ATAES132 cryptographic functions are implemented with a hardware cryptographic engine using AES in CCM mode with a 128-bit key. AES-CCM mode provides both confidentiality and integrity checking with a single key. The integrity MAC includes both the encrypted data and additional authenticate-only data bytes, as described in each command definition. Each MAC is unique due to inclusion of a Nonce and an incrementing MacCount Register in the MAC calculation.

See Appendix I, Cryptographic Computations for information about how the AES computations are performed. Hyperlinks to the AES standard are provided in Appendix A, Standards and Reference Documents.

3.1.2 Hardware Security Features

The ATAES132 device contains physical security features to prevent an attacker from determining the internal secrets. ATAES132 includes tamper detectors for voltage, temperature, frequency, and light, as well as an active metal shield over the circuitry, internal memory encryption, and other various features. The ATAES132 physical design and cryptographic protocol are designed to prevent or significantly complicate most algorithmic, timing, and side-channel attacks.

3.2 Authentication

The authentication commands utilize AES-CCM to generate or validate a MAC value computed using an internally stored key. The command set supports both one-way and mutual authentication. One ATAES132 device can generate packets for authentication of a second ATAES132 device containing the same key. The internal authentication status register remembers only the most recent authentication attempt. A user zone can be configured to require prior authentication of a designated key before access to the user zone is permitted.

3.2.1 Key Authentication

Individual keys can be configured to require a successful authentication prior to use. This requirement can be used to prevent some kinds of exhaustive attacks on the keys. The authentication requirement can be chained to require authentication of several keys prior to allowing a particular operation. The internal Authentication Status Registers remember only the most recent authentication attempt.
3.3 Encrypted Memory Read/Write
A user zone can be configured to require AES-CCM encryption for EEPROM Read or Write operations. If encryption is required for Write access, then the MAC is validated before the received (encrypted) data are written to the EEPROM. If encryption is required for Read access, then ATAES132 encrypts data when they are read from the internal EEPROM, and generates an associated integrity MAC.

3.4 Data Encryption/Decryption
A key can be configured to allow encryption/decryption of small packets of data using AES-CCM with an internally stored key. The Encrypt command encrypts 16 or 32 bytes of plaintext data provided by the Host; the encrypted data and MAC are returned to the Host. The Decrypt command decrypts 16 or 32 bytes of encrypted data after verifying the MAC; the data is returned to the Host only if the MAC is valid. When these commands are used, none of the data is stored in the internal EEPROM.

3.4.1 AES-ECB Encryption/Decryption
A key can be configured to allow AES-ECB mode operations using the Legacy command. A single AES-ECB operation is performed using an internally stored key and the 16-byte input packet received with the AES-ECB command. The 16-byte result is returned to the Host. No input or output formatting is performed by this command, and no data is stored in the internal EEPROM.

3.5 Keys
ATAES132 securely stores sixteen 128-bit keys in the EEPROM. Keys can only be used for the cryptographic functions enabled in the ZoneConfig, CounterConfig, or KeyConfig Register bits in the Configuration Memory. Key values can never be read from ATAES132 under any circumstances. Any key can be used with any user zone. A seventeenth key register in the internal SRAM can be used for session keys.

See Section 7.11, Encrypted Key Writes, for the EncWrite command. See Section 7.18, User Zone ReadOnly Activation, for the Lock command.

3.5.1 Key Management
The key registers can be written with plaintext data or with encrypted data before the Key Memory is locked. After the Key Memory is locked, a key register can only be updated only if the corresponding KeyConfig Register allows updates.

Several key management commands are available for updating or generating the keys:
1. An encrypted key provided by the Host can be written to an internal key register after validating the MAC. The KeyImport command and KeyLoad command perform this function.
2. The internal random number generator can be used to create a key for use as a session key or for storage in an internal Key Register. The new key can also be encrypted and returned to the Host for use as the encrypted key input to another ATAES132 device. The KeyCreate command performs this function.
3. Keys stored in User Memory can be transferred to an internal key register or used as a session key. A user zone configured as extended Key Memory can be used to store eight keys. The KeyTransfer command performs this function.
3.5.2 Limited Use Keys

To prevent exhaustive attacks on the keys, ATAES132 can be configured to limit key usage with a Counter. If a key is configured with a Usage Counter, then the following steps are performed for any command using that key:

1. Read the Counter from memory to check if the count has reached the maximum count value.
2. If the maximum count has been reached, then the command is not executed and an error code is returned.
3. If the maximum count has not been reached, then the Counter is incremented and the command is executed.

By default, the Counters are configured to allow two million counts, allowing two million operations using a key with the usage limits enabled. Atmel recommends that the customer configure Key Usage Counters to a smaller number at personalization; the appropriate key usage limit is dependent on the application. See Appendix H, Understanding Counters for additional information.

3.5.3 Secure Personalization

The ATAES132 is designed to allow personalization of keys using encryption so that the secret key values cannot be determined by a third party. AES encryption of the keys prevents them from being determined by observation of data communicated to or from ATAES132.

A Transport Key is programmed into the KeyID 00 Register by Atmel during the device manufacturing process. This Transport Key is securely exchanged between the customer and Atmel. During personalization, the secret keys are encrypted using the Transport Key before being written to ATAES132.

Atmel also offers a secure personalization service at additional cost which uses a Hardware Security Module (HSM) to store the customer secrets.

3.5.3.1 Key Diversification

Atmel recommends that each unit should contain one or more unique keys to minimize the potential impact of cloning. The keys stored in the ATAES132 should be a cryptographic combination of a root secret not stored in the device along with the unique ATAES132 SerialNum Register value. The Host must have a secure place to store the root secret to protect the integrity of the diversified keys.

It may also be beneficial for the ATAES132 devices to contain secrets for validating the authenticity of the Host. These secrets may need to be the same on all ATAES132 devices for a particular application to permit any Client to validate any Host. See Section 7.13, KeyCreate Command, Mode bit 2.

3.6 Random Numbers

ATAES132 includes a high-quality Random Number Generator (RNG) for Nonce generation, child key creation, and general random number generation. The ATAES132 commands can generate random numbers for internal or external use. Sixteen byte random numbers for external use are generated using the internal RNG and the AES engine, as described in NIST SP800-90.

The RNG can be used to generate the Nonce for cryptographic operations. A mechanism is also provided to synchronize the Nonces in two ATAES132 devices using random numbers generated by both devices. A key can be configured to require that cryptographic operations using the key use a Nonce generated with the internal RNG.
3.6.1 Random Number Generation

The RNG architecture includes both a hardware random number generator and a stored random seed. On power-up, the stored seed is read from the EEPROM, cryptographically combined with the hardware random number generator output, and then stored in SRAM. Whenever a random number is required, this SRAM Seed is cryptographically combined with the hardware random number generator output and the optional input seed to create both a new SRAM Seed and the random number.

For the highest security, the EEPROM Seed should be updated every power cycle in which the RNG is used; however, the EEPROM Seed Register has a maximum life expectancy of 100,000 writes per unit. The Host system is expected to manage the EEPROM Seed by using the command mode option to suppress automatic EEPROM Seed updates.
4. **Security Configuration Registers**

4.1 **User Zone Configuration**

Access permissions to each user zone are controlled by the ZoneConfig Registers in the Configuration Memory. There is one ZoneConfig Register for each User Memory zone.

<table>
<thead>
<tr>
<th>ZoneConfig Field</th>
<th>Byte</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthRead</td>
<td>0</td>
<td>0</td>
<td>1b = Authentication is required to read data. 0b = Authentication is not required to read data.</td>
</tr>
<tr>
<td>AuthWrite</td>
<td>0</td>
<td>1</td>
<td>1b = Authentication is required to write data. 0b = Authentication is not required to write data.</td>
</tr>
<tr>
<td>EncRead</td>
<td>0</td>
<td>2</td>
<td>1b = Encryption is required to read data. 0b = Encryption is not required to read data.</td>
</tr>
<tr>
<td>EncWrite</td>
<td>0</td>
<td>3</td>
<td>1b = Encryption is required to write data. 0b = Encryption is not required to write data.</td>
</tr>
<tr>
<td>WriteMode</td>
<td>0</td>
<td>4 to 5</td>
<td>00b = Zone is permanently read/write. 01b = Zone is permanently read-only. 10b = The ReadOnly byte determines if writes are permitted. 11b = The ReadOnly byte determines if writes are permitted, and the Lock command must include an authenticating MAC calculated using the KeyID stored in ZoneConfig[UZ].WriteID.</td>
</tr>
<tr>
<td>UseSerial</td>
<td>0</td>
<td>6</td>
<td>UseSerial = 1b and EncWrite = 1b, then SerialNum must be included in EncWrite operations. EncWrite = 0b, then this bit is ignored.</td>
</tr>
<tr>
<td>UseSmall</td>
<td>0</td>
<td>7</td>
<td>UseSmall = 1b and EncWrite = 1b, the first four bytes of SmallZone must be included in EncWrite operations. EncWrite = 0b, this bit is ignored.</td>
</tr>
<tr>
<td>ReadID</td>
<td>1</td>
<td>0 to 3</td>
<td>KeyID which is used to encrypt data read from this zone. The same key is used to generate the MAC.</td>
</tr>
<tr>
<td>AuthID</td>
<td>1</td>
<td>4 to 7</td>
<td>KeyID which is used for inbound authentication before access is permitted.</td>
</tr>
<tr>
<td>Reserved</td>
<td>2</td>
<td>0 to 3</td>
<td>Reserved for future use. All bits must be 0b.</td>
</tr>
<tr>
<td>WriteID</td>
<td>2</td>
<td>4 to 7</td>
<td>KeyID that is used to decrypt data written to this zone. The same key is used to verify the MAC.</td>
</tr>
<tr>
<td>ReadOnly</td>
<td>3</td>
<td>0 to 7</td>
<td>The contents of this byte are ignored unless WriteMode contains 10b or 11b. If 0x55, then the user zone is Read/Write. If any other value, then the user zone is read-only. This byte can be updated after the Configuration Memory is locked using the Lock command (See Section 7.18, Lock Command).</td>
</tr>
</tbody>
</table>

**Note:** 1. Most changes to the ZoneConfig Registers take effect immediately. Changes to the AuthRead and EncRead bits do not affect the SPI or I²C Read command until the next reset or power-up.  
**Warning:** ATAES132 must always be locked by the customer prior to shipment to the end user to protect the customer secrets. See Section 7.18.
### 4.2 Key Configuration

Restrictions on key usage are controlled by the KeyConfig Registers in the Configuration Memory. There is one KeyConfig Register for each key.

<table>
<thead>
<tr>
<th>KeyConfig Field</th>
<th>Byte</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>ExternalCrypto</td>
<td>0</td>
<td>0</td>
<td>1b = The key can be used with the Encrypt and Decrypt commands. (3) 0b = The Encrypt and Decrypt commands are prohibited.</td>
</tr>
<tr>
<td>InboundAuth</td>
<td>0</td>
<td>1</td>
<td>1b = The key can only be used by the Auth command for Inbound Only or Mutual Authentication. The key cannot be used by any other command, but KeyID can be the target of a key management command. 0b = The key can be used for any purpose not prohibited by another KeyConfig bit, including Outbound Only authentication.</td>
</tr>
<tr>
<td>RandomNonce</td>
<td>0</td>
<td>2</td>
<td>1b = Operations using this key requires a random Nonce (see Section 7.19). 0b = The Nonce is not required to be random.</td>
</tr>
<tr>
<td>LegacyOK</td>
<td>0</td>
<td>3</td>
<td>1b = The key can be used with the Legacy command. 0b = The key cannot be used with the Legacy command.</td>
</tr>
<tr>
<td>AuthKey</td>
<td>0</td>
<td>4</td>
<td>1b = The key requires prior authentication using the KeyID stored in LinkPointer. 0b = Prior authentication is not required.</td>
</tr>
<tr>
<td>Child</td>
<td>0</td>
<td>5</td>
<td>1b = The key is permitted to be the target of a KeyCreate or KeyLoad command. 0b = This use is prohibited.</td>
</tr>
<tr>
<td>Parent</td>
<td>0</td>
<td>6</td>
<td>1b = This key can be used as the parent when writing VolatileKey via KeyCreate, KeyImport, or KeyLoad (see Section 4.3). 0b = This use is prohibited.</td>
</tr>
<tr>
<td>ChangeKeys</td>
<td>0</td>
<td>7</td>
<td>1b = Key updates are permitted after locking. The new key is written using the EncWrite command with a MAC generated with the current value of key. (see Section 7.11). 0b = Key updates with EncWrite command are prohibited.</td>
</tr>
<tr>
<td>CounterLimit</td>
<td>1</td>
<td>0</td>
<td>1b = Usage count limits are enabled for this key (see CounterNum). 0b = No usage limits.</td>
</tr>
<tr>
<td>ChildMac</td>
<td>1</td>
<td>1</td>
<td>1b = An input MAC is required to modify this key using the KeyCreate command. 0b = The KeyCreate command does not require an input MAC (it will be ignored, if provided).</td>
</tr>
<tr>
<td>AuthOut</td>
<td>1</td>
<td>2</td>
<td>1b = I2C Auth signaling is enabled for this key (see Appendix J.5). 0b = I2C Auth signaling is disabled for this key.</td>
</tr>
<tr>
<td>AuthOutHold</td>
<td>1</td>
<td>3</td>
<td>1b = The I2C AuthO output state is unchanged when an authentication reset is executed using this key. 0b = Then the I2C AuthO output is reset when an authentication reset is executed using this key (see Appendix J.5).</td>
</tr>
<tr>
<td>ImportOK</td>
<td>1</td>
<td>4</td>
<td>1b = The key is permitted to be the target of a KeyImport command. 0b = KeyImport command is prohibited.</td>
</tr>
<tr>
<td>ChildAuth</td>
<td>1</td>
<td>5</td>
<td>1b = The KeyCreate command requires prior authentication using the KeyID stored in LinkPointer. 0b = Prior authentication is not required.</td>
</tr>
<tr>
<td>KeyConfig Field</td>
<td>Byte</td>
<td>Bit</td>
<td>Description</td>
</tr>
<tr>
<td>-----------------</td>
<td>------</td>
<td>-----</td>
<td>-------------</td>
</tr>
</tbody>
</table>
| TransferOK      | 1    | 6   | 1b = The key is permitted to be the target of a KeyTransfer command (see Section 7.16).  
0b = KeyTransfer command is prohibited. |
| AuthCompute     | 1    | 7   | 1b = The key can be used with the AuthCompute command.  
0b = The key cannot be used with the AuthCompute command. |
| LinkPointer     | 2    | 0 to 3 | For child keys; stores the ParentKeyID.  
For all other keys; the KeyID of the authorizing key (see AuthKey). |
| CounterNum      | 2    | 4 to 7 | Stores the CntID of the Monotonic Counter attached to this key for usage limits or for MAC calculation. MAC calculations will include the Counter if Command Mode bit 5 is 1b even if key usage limits are disabled. |
| Reserved        | 3    | 0 to 7 | Reserved for future use. All bits must be 0b. |

Notes: 1. Changes to the KeyConfig Registers take effect immediately which allows the functionality to be verified during the personalization process.

2. **Warning:** ATAES132 must always be locked by the customer prior to shipment to the end user to protect the customer secrets. See Section 7.18, Lock Command.

3. **Warning:** Since the Encrypt command does not include an input MAC, the Encrypt command can exhaustively be run with selected input data to attack the key. Requiring authentication prior to allowing encryption makes these attacks more difficult. To require prior authentication, the AuthKey and RandomNonce bits must be set to 1b.

4. A key can be disabled by setting KeyConfig[KeyN].AuthKey to 1b, and KeyConfig[KeyN].LinkPointer to contain “KeyN”, where KeyN = KeyID of the key being configured.
4.3 VolatileKey Configuration

There is a seventeenth key register, named VolatileKey, which has a KeyID of 0xFF and is stored in the internal SRAM. This key location can be written with the KeyCreate (see Section 7.13, KeyCreate Command), KeyImport (see Section 7.14, KeyImport Command), KeyLoad (see Section 7.15, KeyLoad Command), or KeyTransfer (see Section 7.16, KeyTransfer Command) commands. The contents of the VolatileKey Register are erased when the device is powered down, enters the Sleep state, or is reset.

When the VolatileKey Register is loaded, restrictions are placed on its usage which persists until the power is lost or the key is reloaded. The definition of the VolUsage field is shown in Table 4-3.

Table 4-3. VolUsage Field Bit Definitions in the KeyCreate or KeyLoad Command at VolatileKey Creation

<table>
<thead>
<tr>
<th>VolUsage Field Name</th>
<th>Byte</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthOK</td>
<td>0</td>
<td>0</td>
<td>1b = Auth command can be run using this key. 0b = Auth command is prohibited.</td>
</tr>
<tr>
<td>EncryptOK</td>
<td>0</td>
<td>1 to 2</td>
<td>00b = Encrypt command is prohibited. 01b = Encrypt command can be run using this key without a prior authentication. 10b or 11b = Encrypt command can be run using this key only with a prior authentication.</td>
</tr>
<tr>
<td>DecryptOK</td>
<td>0</td>
<td>3</td>
<td>1b = Decrypt command can be run using this key. 0b = Decrypt command is prohibited.</td>
</tr>
<tr>
<td>RandomNonce</td>
<td>0</td>
<td>4</td>
<td>1b = Operations using this key require a random Nonce (see Section 7.19, Nonce Command). 0b = A fixed (input-only) Nonce is permitted.</td>
</tr>
<tr>
<td>AuthCompute</td>
<td>0</td>
<td>5</td>
<td>1b = AuthCompute command can be run using this key. 0b = AuthCompute command is prohibited.</td>
</tr>
<tr>
<td>LegacyOK</td>
<td>0</td>
<td>6</td>
<td>1b = Legacy command can be run using this key. 0b = Legacy command is prohibited.</td>
</tr>
<tr>
<td>Reserved</td>
<td>0</td>
<td>7</td>
<td>Reserved for future use. All bits must be 0b.</td>
</tr>
<tr>
<td>WriteCompute</td>
<td>1</td>
<td>0</td>
<td>1b = WriteCompute command can be run using this key. 0b = WriteCompute command is prohibited.</td>
</tr>
<tr>
<td>DecRead</td>
<td>1</td>
<td>1</td>
<td>1b = DecRead command can be run using this key. 0b = DecRead command is prohibited.</td>
</tr>
<tr>
<td>Reserved</td>
<td>1</td>
<td>2 to 7</td>
<td>Reserved for future use. All bits must be 0b.</td>
</tr>
</tbody>
</table>

Note: 1. **Warning:** Since the Encrypt command does not include an input MAC, the Encrypt command can be exhaustively run with selected input data to attack VolatileKey. Requiring authentication prior to allowing encryption makes these attacks more difficult. To implement this, the Auth and RandomNonce bits must be set to 1b, and the Encrypt bits must be set to 10b or 11b when the VolatileKey is created.
4.4 Counter Configuration

The CounterConfig Registers impose restrictions on the usage of the Counter command with a Counter (see Section 7.5, Counter Command). There is one CounterConfig Register for each Counter. Each Counter can increment up to a value of 2,097,134 using the Count command, after which they can no longer be changed. See Appendix H, Understanding Counters for additional Counter information.

The CounterConfig bits have no impact on the functionality of a Key Usage Counter. If a Counter is identified in a KeyConfig Register (see Section 4.2, Key Configuration) as a Key Usage Counter, then the Counter will increment each time the key is used. The CounterConfig[CntID].IncrementOK bit is typically set to 0b to prohibit the Counter command from incrementing a Key Usage Counter.

Table 4-4. CounterConfig Register Bit Definitions (1)(2)

<table>
<thead>
<tr>
<th>CounterConfig Field</th>
<th>Byte</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IncrementOK</td>
<td>0</td>
<td>0</td>
<td>1b = Increments using the Counter command are permitted. 0b = Increments using the Counter command are prohibited.</td>
</tr>
<tr>
<td>RequireMAC</td>
<td>0</td>
<td>1</td>
<td>1b = The increment operation requires an input MAC. 0b = An input MAC is prohibited.</td>
</tr>
<tr>
<td>Reserved</td>
<td>0</td>
<td>2 to 7</td>
<td>Reserved for future use. All bits must be 0b.</td>
</tr>
<tr>
<td>IncrID</td>
<td>1</td>
<td>0 to 3</td>
<td>KeyID of the key used to generate the Counter command input MAC for increment operations.</td>
</tr>
<tr>
<td>MacID</td>
<td>1</td>
<td>4 to 7</td>
<td>KeyID of the key used to generate the Counter command output MAC for Counter Read operations.</td>
</tr>
</tbody>
</table>

Notes: 1. Changes to the CounterConfig Registers take effect immediately, allowing the functionality to be verified during the personalization process.

2. Warning: ATAES132 must always be locked by the customer prior to shipment to the end user to protect the customer secrets. See Section 7.18, Lock Command.
5. **Standard Serial EEPROM Read and Write Commands**

This section provides a summary of the operations that can be performed using the standard Serial EEPROM Read and Write commands. For detailed information, see the specification sections that are referenced below.

### Table 5-1. Standard Serial EEPROM Read and Write Commands

<table>
<thead>
<tr>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Read</td>
<td>The Read command is used to read cleartext from the user zones, to retrieve a response by reading the Response Memory Buffer, or to read the STATUS Register.</td>
</tr>
<tr>
<td>Write</td>
<td>The Write command is used to write cleartext to unrestricted memory or to send a command by writing the command packet to the Command Memory Buffer. The Write command is also used to write the IO Address Reset Register.</td>
</tr>
</tbody>
</table>

5.1 **Read**

ATAES132 supports the standard Serial EEPROM commands to read from the User Memory. All bytes in the User Memory address space may be read; however, only bytes in the user zones in which neither authentication nor encryption is required will return the actual data from the memory. All other locations will return the value 0xFF. See Appendix J, I²C Interface for I²C Read command information and Appendix K, SPI Interface for SPI Read command information.

When a Read command is received, the device looks at the AuthRead and EncRead bits in the ZoneConfig Register for the user zone to determine whether to return 0xFF or the EEPROM data. If the EncRead bit is 1b or the AuthRead bit is 1b, then 0xFF will always be returned.

If the ZoneConfig AuthRead bit is 1b and the EncRead is 0b, then the BlockRead command must be used to read the user zone (see Section 7.4, BlockRead Command). If the EncRead bit is 1b, then the EncRead command must be used to read the user zone (see Section 7.9, EncRead Command).

The standard SPI and I²C Read commands can be used to read any number of bytes in a single operation. Read operations can cross EEPROM page boundaries.

5.1.1 **Read the Response Memory Buffer**

The Host sends ATAES132 commands to the device by writing the command packet to the Command Memory Buffer using a standard SPI or I²C Write command. ATAES132 processes the command packet and places the response in the Response Memory Buffer. The Host retrieves the response by reading the response packet using a standard SPI or I²C Read command. See Appendix D, Command Memory Map for additional information. See Appendix G, Understanding the STATUS Register for examples.

When any error occurs, the EERR bit of the STATUS Register is set to 1b to indicate an error. See Appendix G.1, Device Status Register (STATUS) Definition for more information.
5.1.2 Read the Key Memory or Configuration Memory

Reading the Key Memory is never allowed.

The Read command can never be used to read data from the Configuration Memory. The BlockRead command is used to access the Configuration Memory (see Section 7.4, BlockRead Command).

If a standard SPI or I²C Read command is used within the Configuration Memory or Key Memory address space, then 0xFF will be returned for each byte. 0xFF is also returned for address locations that do not physically exist. The EERR bit of the STATUS Register is set to 1b if 0xFF was substituted for any byte returned by a Read command. See Appendix G.1, Device Status Register (STATUS) Definition for more information.

5.1.3 Read the STATUS Register

The Host reads the STATUS Register by reading address 0xFFF0. In SPI interface mode, the Host can also read STATUS using the RDSR command. See Appendix G, Understanding the STATUS Register, for detailed information and examples.

5.2 Write

ATAES132 supports the standard Serial EEPROM commands to write to unrestricted User Memory (AuthWrite and EncWrite are both 0b). See Appendix J, I²C Interface for I²C Write command information and Appendix K, SPI Interface for SPI Write command information. The ATAES132 is capable of writing 1 to 32 bytes on a single physical page with each Write operation.

The Write command can only write data to a single user zone; the data cannot span multiple user zones. The Write command can write data only to a single EEPROM page; the data cannot cross page boundaries. The EERR bit of the STATUS Register is set to 1b to indicate an error if a prohibited Write is attempted. See Appendix G.1, Device Status Register (STATUS) Definition for more information.

5.2.1 Write the Command Memory Buffer

The Host sends ATAES132 commands to the device by writing the command packet to the Command Memory Buffer using a standard SPI or I²C Write command. The ATAES132 processes the command packet and places the response in the Response Memory Buffer. The Host retrieves the response by reading the response packet using a standard SPI or I²C read command. See Appendix D, Command Memory Map for additional information. See Appendix G, for examples.

When any error occurs, either the EERR or CRCE bit of the STATUS Register is set to 1b to indicate an error. See Appendix G.1 for more information.

5.2.2 Write the IO Address Reset Register

The Host can reset the pointer in the Command Memory Buffer and the Response Memory Buffer by writing to address 0xFFFE. See Appendix D.3, IO Address Reset Register for additional information.
5.2.3 Write the Key Memory or Configuration Memory

ATAES132 supports standard Serial EEPROM commands to write the Configuration Memory or the Key Memory prior to locking. ATAES132 is capable of writing 1 to 32 bytes on a single physical page with each Write operation.

Note: Partial writes to key registers are prohibited.

If LockKeys has a value of 0x55 (unlocked) and the address points to Key Memory, then the starting address must be the first byte of a key register, and 16 bytes of cleartext data must be sent. If these conditions are not satisfied, then an error response will be generated and the EEPROM will remain unchanged.

If LockConfig has a value of 0x00 (locked) and the address points to the Configuration Memory, then a Write command will generate an error and the EEPROM will be unchanged.

If LockConfig has a value of 0x55 (unlocked), then the User Zone write restrictions imposed by ZoneConfig are enforced, but can be changed.

WARNING Atmel does not recommend writing secret data into the User Zones prior to locking the Configuration Memory due to the fact an attacker can change the ZoneConfig bits to allow a read of the User Zone if the Configuration Memory is unlocked.

When any error occurs, either the EERR bit or the CRCE bit of the STATUS Register is set to 1b to indicate an error. See Appendix G.1 for more information. See the Lock command (Section 7.18.1, User Zone ReadOnly Activation) for additional information.
6. Commands

6.1 Command Block and Packet

The Host sends ATAES132 extended commands to the device in a block of at least nine bytes. The ATAES132 responses are returned to the Host in a block of at least four bytes. The command and response blocks are constructed in the following manner:

**Table 6-1. Command and Response Blocks Descriptions**

<table>
<thead>
<tr>
<th>Byte</th>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Count</td>
<td>Number of bytes to be transferred to the device in the block, including Count, Packet, and Checksum. This byte will always have a value of N.</td>
</tr>
<tr>
<td>1 to (N-3)</td>
<td>Packet</td>
<td>Command, parameters, and data or response. Data is transmitted in the byte order shown in the command definitions.</td>
</tr>
<tr>
<td>N-2, N-1</td>
<td>Checksum</td>
<td>Atmel CRC-16 verification of the Count and Packet bytes. See Appendix M, Block Checksum for additional information and examples.</td>
</tr>
</tbody>
</table>

**Table 6-2. Input Command Packet Descriptions within the Command Block**

<table>
<thead>
<tr>
<th>Byte</th>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Opcode</td>
<td>Command Code</td>
</tr>
<tr>
<td>2</td>
<td>Mode</td>
<td>Command Modifier</td>
</tr>
<tr>
<td>3, 4</td>
<td>Param1</td>
<td>First Command Parameter</td>
</tr>
<tr>
<td>5, 6</td>
<td>Param2</td>
<td>Second Command Parameter</td>
</tr>
<tr>
<td>7+</td>
<td>Data</td>
<td>Optional Input Data</td>
</tr>
</tbody>
</table>

**Table 6-3. Response Packet Descriptions within the Response Block**

<table>
<thead>
<tr>
<th>Byte</th>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ReturnCode</td>
<td>Command Return Code (See Section 6.3, ReturnCode)</td>
</tr>
<tr>
<td>2+</td>
<td>Data</td>
<td>Optional Output Data</td>
</tr>
</tbody>
</table>

**Table 6-4. Response Packet Descriptions Contains when an Error Occurs**

<table>
<thead>
<tr>
<th>Byte</th>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>ReturnCode</td>
<td>Error Code (See Section 6.3, ReturnCode)</td>
</tr>
</tbody>
</table>

The Host sends ATAES132 commands to the device by writing the command block to the Command Memory Buffer using a standard SPI or \(^{2}\)C Write command. ATAES132 processes the Command Packet and places the response block in the Response Memory Buffer. The Host retrieves the response by reading the response block using a standard SPI or \(^{2}\)C Read command. If the Host reads beyond the end of the block, then 0xFF is returned.
### 6.2 Command Summary

Table 6-5 shows the command set sorted by the opcode value. Table 6-6 shows the command set in alphabetical order by command name. See Section 7, Command Definitions for the ATAES132 command definitions.

Table 6-5. Extended ATAES132 Command Set Sorted by Opcode Value

<table>
<thead>
<tr>
<th>Opcode</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>Reset</td>
<td>Resets the device, clearing the cryptographic status.</td>
</tr>
<tr>
<td>0x01</td>
<td>Nonce</td>
<td>Generates a 128-bit Nonce from the internal random number generator for use by the cryptographic commands. This command can also be used to write a Host Nonce directly into the Nonce Register.</td>
</tr>
<tr>
<td>0x02</td>
<td>Random</td>
<td>Returns a 128-bit random number from the internal random number generator.</td>
</tr>
<tr>
<td>0x03</td>
<td>Auth</td>
<td>Performs one-way or mutual authentication using the specified key.</td>
</tr>
<tr>
<td>0x04</td>
<td>EncRead</td>
<td>Encrypts 1 to 32 bytes of data from User Memory and returns the encrypted data and integrity MAC.</td>
</tr>
<tr>
<td>0x05</td>
<td>EncWrite</td>
<td>Writes 1 to 32 bytes of encrypted data into the User Memory or Key Memory after verifying the integrity MAC.</td>
</tr>
<tr>
<td>0x06</td>
<td>Encrypt</td>
<td>Encrypts 16 or 32 bytes of plaintext data provided by the Host.</td>
</tr>
<tr>
<td>0x07</td>
<td>Decrypt</td>
<td>Decrypts 16 or 32 bytes of data provided by the Host after verifying the integrity MAC.</td>
</tr>
<tr>
<td>0x08</td>
<td>KeyCreate</td>
<td>Generates a random number, stores it in Key Memory, and returns the encrypted key to the Host.</td>
</tr>
<tr>
<td>0x09</td>
<td>KeyLoad</td>
<td>Writes an encrypted key to Key Memory after verifying the integrity MAC.</td>
</tr>
<tr>
<td>0x0A</td>
<td>Counter</td>
<td>Increments a High Endurance Counter and/or returns the Counter value.</td>
</tr>
<tr>
<td>0x0B</td>
<td>Crunch</td>
<td>Processes a seed value through the internal crunch engine. This function is used to detect clones.</td>
</tr>
<tr>
<td>0x0C</td>
<td>Info</td>
<td>Returns device information: MacCount, Authentication status, or hardware revision code.</td>
</tr>
<tr>
<td>0x0D</td>
<td>Lock</td>
<td>Permanently locks the Configuration Memory or Key Memory. Locked memory can never be unlocked.</td>
</tr>
<tr>
<td>0x0F</td>
<td>Legacy</td>
<td>Performs a single AES-ECB mode operation on 16 bytes of data provided by the Host.</td>
</tr>
<tr>
<td>0x10</td>
<td>BlockRead</td>
<td>Reads 1 to 32 bytes of data from User Memory or the Configuration Memory. Returns cleartext data.</td>
</tr>
<tr>
<td>0x11</td>
<td>Sleep</td>
<td>Places the device in the Sleep state or Standby state to reduce power consumption.</td>
</tr>
<tr>
<td>0x13</td>
<td>NonceCompute</td>
<td>Generates a Nonce in a manner that allows two ATAES132 devices to have identical Nonce values.</td>
</tr>
<tr>
<td>0x14</td>
<td>AuthCompute</td>
<td>Computes the input MAC required to execute the Auth command or to increment a counter using the Counter command on a second ATAES132 device.</td>
</tr>
<tr>
<td>0x15</td>
<td>AuthCheck</td>
<td>Checks the output MAC generated by the Auth command or by reading a counter using the Counter command on a second ATAES132 device.</td>
</tr>
<tr>
<td>0x16</td>
<td>WriteCompute</td>
<td>Encrypts data and generates the input MAC required to execute the EncWrite command.</td>
</tr>
<tr>
<td>0x17</td>
<td>DecRead</td>
<td>Checks the output MAC and decrypts data that was encrypted by the EncRead command.</td>
</tr>
<tr>
<td>0x19</td>
<td>KeyImport</td>
<td>Decrypts and writes a key that was output by the KeyCreate command.</td>
</tr>
<tr>
<td>0x1A</td>
<td>KeyTransfer</td>
<td>Transfers a key from User Memory into the Key Memory or into the VolatileKey Register.</td>
</tr>
</tbody>
</table>

Note: 1. The most-significant three bits of the command opcode may contain any value; they are ignored by the ATAES132 command decoder.
### Table 6-6. Extended ATAES132 Command Set Sorted by Command Name

<table>
<thead>
<tr>
<th>Opcode(1)</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x03</td>
<td>Auth</td>
<td>Performs one-way or mutual authentication using the specified key.</td>
</tr>
<tr>
<td>0x15</td>
<td>AuthCheck</td>
<td>Checks the output MAC generated by the Auth command or by reading a counter using the Counter command on a second ATAES132 device.</td>
</tr>
<tr>
<td>0x14</td>
<td>AuthCompute</td>
<td>Computes the input MAC required to execute the Auth command or to increment a counter using the Counter command on a second ATAES132 device.</td>
</tr>
<tr>
<td>0x10</td>
<td>BlockRead</td>
<td>Reads 1 to 32 bytes of data from User Memory or the Configuration Memory. Returns cleartext data.</td>
</tr>
<tr>
<td>0x0A</td>
<td>Counter</td>
<td>Increments a high endurance Counter and/or returns the counter value.</td>
</tr>
<tr>
<td>0x0B</td>
<td>Crunch</td>
<td>Processes a seed value through the internal crunch engine. This function is used to detect clones.</td>
</tr>
<tr>
<td>0x17</td>
<td>DecRead</td>
<td>Checks the output MAC and decrypts data that was encrypted by the EncRead command.</td>
</tr>
<tr>
<td>0x07</td>
<td>Decrypt</td>
<td>Decrypts 16 or 32 bytes of data provided by the Host after verifying the integrity MAC.</td>
</tr>
<tr>
<td>0x04</td>
<td>EncRead</td>
<td>Encrypts 1 to 32 bytes of data from User Memory and returns the encrypted data and integrity MAC.</td>
</tr>
<tr>
<td>0x06</td>
<td>Encrypt</td>
<td>Encrypts 16 or 32 bytes of plaintext data provided by the Host.</td>
</tr>
<tr>
<td>0x05</td>
<td>EncWrite</td>
<td>Writes 1 to 32 bytes of encrypted data into the User Memory or Key Memory after verifying the integrity MAC.</td>
</tr>
<tr>
<td>0x0C</td>
<td>Info</td>
<td>Returns device information: MacCount, Authentication status, or hardware revision code.</td>
</tr>
<tr>
<td>0x08</td>
<td>KeyCreate</td>
<td>Generates a random number, stores it in Key Memory, and returns the encrypted key to the Host.</td>
</tr>
<tr>
<td>0x19</td>
<td>KeyImport</td>
<td>Decrypts and writes a key that was output by the KeyCreate command.</td>
</tr>
<tr>
<td>0x09</td>
<td>KeyLoad</td>
<td>Writes an encrypted key to Key Memory after verifying the integrity MAC.</td>
</tr>
<tr>
<td>0x1A</td>
<td>KeyTransfer</td>
<td>Transfers a key from User Memory into the Key Memory or into the VolatileKey Register.</td>
</tr>
<tr>
<td>0x0F</td>
<td>Legacy</td>
<td>Performs a single AES-ECB mode operation on 16 bytes of data provided by the Host.</td>
</tr>
<tr>
<td>0x0D</td>
<td>Lock</td>
<td>Permanently locks the Configuration Memory or Key Memory. Locked memory can never be unlocked.</td>
</tr>
<tr>
<td>0x01</td>
<td>Nonce</td>
<td>Generates a 128-bit Nonce from the internal random number generator for use by the cryptographic commands. This command can also be used to write a Host Nonce directly into the Nonce Register.</td>
</tr>
<tr>
<td>0x13</td>
<td>NonceCompute</td>
<td>Generates a Nonce in a manner that allows two ATAES132 devices to have identical Nonce values.</td>
</tr>
<tr>
<td>0x02</td>
<td>Random</td>
<td>Returns a 128-bit random number from the internal random number generator.</td>
</tr>
<tr>
<td>0x00</td>
<td>Reset</td>
<td>Resets the device, clearing the cryptographic status.</td>
</tr>
<tr>
<td>0x11</td>
<td>Sleep</td>
<td>Places the device in the Sleep state or Standby state to reduce power consumption.</td>
</tr>
<tr>
<td>0x16</td>
<td>WriteCompute</td>
<td>Encrypts data and generates the input MAC required to execute the EncWrite command.</td>
</tr>
</tbody>
</table>

**Note:** 1. The most-significant three bits of the command opcode may contain any value; they are ignored by the ATAES132 command decoder.
6.3 **ReturnCode**

The response packet for each ATAES132 command includes a ReturnCode to report success or failure to the Host.

The Reset command and the Sleep command do not generate a ReturnCode because they do not generate a response packet. All other ATAES132 commands generate a ReturnCode.

<table>
<thead>
<tr>
<th>Value</th>
<th>Name</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x00</td>
<td>Success</td>
<td>No errors.</td>
</tr>
<tr>
<td>0x02</td>
<td>BoundaryError</td>
<td>Crossed a page boundary for a Write, BlockRead, or EncRead; crossed a Key Register boundary for a Write or EncWrite.</td>
</tr>
<tr>
<td>0x04</td>
<td>RWConfig</td>
<td>Access to the specified user zone is not permitted due to the configuration or internal state.</td>
</tr>
<tr>
<td>0x08</td>
<td>BadAddr</td>
<td>Attempted to Write Locked Memory, address is not implemented, or address is illegal for this command.</td>
</tr>
<tr>
<td>0x10</td>
<td>CountErr</td>
<td>Counter limit reached, count usage error, or restricted key error.</td>
</tr>
<tr>
<td>0x20</td>
<td>NonceError</td>
<td>Nonce invalid or not available, Nonce not generated with internal RNG. MacCount limit has been reached.</td>
</tr>
<tr>
<td>0x40</td>
<td>MacError</td>
<td>Missing input MAC, or MAC compare failed.</td>
</tr>
<tr>
<td>0x50</td>
<td>ParseError</td>
<td>Bad opcode, bad mode, bad param, invalid length, or other encoding failure.</td>
</tr>
<tr>
<td>0x60</td>
<td>DataMatch</td>
<td>EEPROM post-write automatic data verification failed due to data mismatch.</td>
</tr>
<tr>
<td>0x70</td>
<td>LockError</td>
<td>Lock command contained bad Checksum or bad MAC.</td>
</tr>
<tr>
<td>0x80</td>
<td>KeyErr</td>
<td>Key not permitted to be used for this operation or wrong key was used for operation. Prior authentication has not been performed. Other authentication error or other key error.</td>
</tr>
</tbody>
</table>

If ReturnCode has any value other than 0x00, no additional data will be returned by the ATAES132. If the ReturnCode is greater than zero for any command that performs cryptographic operations, then the Nonce will be invalidated. A non-zero ReturnCode only reports the first error encountered; although, multiple errors might exist.
7. Command Definitions

The ATAES132 extended command definitions are described in this section. The commands are presented in alphabetical order by command name. The standard Serial EEPROM Read and Write commands are in Section 5, Standard Serial EEPROM Read and Write Commands and are not included in this section. The cryptographic operations performed by the ATAES132 extended commands are described in Appendix I, Cryptographic Computations.

7.1 Auth Command

The Auth command performs a one-way or mutual authentication using AES-CCM. The Auth command options are shown in Table 7-1. The Nonce Register value is used as the CCM Nonce for all Auth command MAC calculations.

- **Mutual Authentication**
  The InMAC is verified, and upon success, an OutMAC is calculated and returned to the Host. The AuthComplete status flag is set to YesAuth if the InMAC is verified.

- **Outbound Only Authentication**
  The OutMAC is calculated and output to the Host. The AuthComplete status flag is set to NoAuth. Outbound-only Authentication is also known as Challenge-Response Authentication.

- **Inbound Only Authentication**
  The InMAC value is verified, and the success or failure is reported to the Host. The AuthComplete status flag is set to YesAuth if the InMAC is verified.

- **Authentication Reset**
  The AuthComplete status flag is set to NoAuth.

Table 7-1. Auth Command Options

<table>
<thead>
<tr>
<th>Mode Bit 1</th>
<th>Mode Bit 0</th>
<th>Description</th>
<th>InMAC</th>
<th>OutMAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1b</td>
<td>1b</td>
<td>Mutual Authentication</td>
<td>Required</td>
<td>Generated</td>
</tr>
<tr>
<td>1b</td>
<td>0b</td>
<td>Outbound Only Authentication</td>
<td>Prohibited</td>
<td>Generated</td>
</tr>
<tr>
<td>0b</td>
<td>1b</td>
<td>Inbound Only Authentication</td>
<td>Required</td>
<td>No</td>
</tr>
<tr>
<td>0b</td>
<td>0b</td>
<td>Authentication Reset</td>
<td>Prohibited</td>
<td>No</td>
</tr>
</tbody>
</table>

If a MAC is required or will be generated by the Auth command, then a valid Nonce is required. If the KeyConfig[AKeyID].RandomNonce bit is 1b, then the Nonce must be random.

The AuthCompute command can be used to generate the InMac required for Inbound Only Authentication, or Mutual Authentication (see Section 7.3, AuthCompute Command). The AuthCheck command can be used to validate the OutMac (see Section 7.2, AuthCheck Command).

In the I²C interface mode, the Auth command can also be used for Auth signaling. See Appendix J.5, I²C Auth Signaling.

7.1.1 Authentication Status Register

The Authentication Status Register contains the AKeyID, the AuthComplete status flag, and the usage bits. Prior to executing the Auth command, the AuthComplete status flag is set to NoAuth. If the InMAC is successfully verified in the Inbound Only or Mutual Authentication mode, then the AuthComplete status flag is set to YesAuth.

The ATAES132 Authentication Status Register only stores the result of the most recent authentication attempt. If there is a parsing or execution error, then the prior Authentication status will be lost.
7.1.2 Authentication Usage

The usage field (Param2) controls which operations are permitted with a successful Inbound-only or Mutual Authentication (see Table 7-2). If Param2 is 0x0000, the AuthComplete flag is set to NoAuth, but the authentication outputs are generated. Param2 is ignored if outbound-only authentication is performed.

Table 7-2. Auth Command Usage Field Definition (Param2)

<table>
<thead>
<tr>
<th>Byte</th>
<th>Bit</th>
<th>Name</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>0</td>
<td>ReadOK</td>
<td>1b = Read and EncRead commands are enabled for user zone reads after successful authentication.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0b = Read and EncRead commands are prohibited for user zone reads if authentication is required in ZoneConfig[UZ] (see Section 4.1, User Zone Configuration).</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>WriteOK</td>
<td>1b = Write and EncWrite commands are enabled for user zone writes after successful authentication.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0b = Write and EncWrite commands are prohibited for user zone writes if authentication is required in ZoneConfig[UZ] (see Section 4.1).</td>
</tr>
<tr>
<td>0</td>
<td>2</td>
<td>KeyUse</td>
<td>1b = If a key requires authentication (KeyConfig[AKeyID].AuthKey is 1b), the Encrypt, Decrypt, Legacy, KeyCreate, and KeyLoad commands are enabled after successful authentication.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0b = EncRead, EncWrite, Encrypt, Decrypt, Legacy, KeyCreate, and KeyLoad commands using the authenticated key are prohibited after authentication (see Section 4.2, Key Configuration).</td>
</tr>
<tr>
<td>0</td>
<td>3–7</td>
<td>Zero</td>
<td>Reserved. Must be 0b.</td>
</tr>
<tr>
<td>1</td>
<td>0:7</td>
<td>Zero</td>
<td>Reserved. Must be 0x00.</td>
</tr>
</tbody>
</table>

If the AKeyID is VolatileKey, then VolUsage.AuthOK must be 1b when the key is loaded or authentication will fail.

Table 7-3. Input Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>Auth</td>
<td>1</td>
</tr>
<tr>
<td>Mode</td>
<td>Mode</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 0 and Bit 1 if:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>11b = Perform mutual authentication.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>10b = Perform Outbound Only authentication.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>01b = Perform Inbound Only authentication.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>00b = Perform authentication reset.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bits 2, 3, and 4: Reserved. Must be 0b.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 5: 1b = Include the associated Usage Counter in the authentication.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 6: 1b = Include SerialNum in the authentication.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 7: 1b = Include the first four bytes of SmallZone in the authentication.</td>
</tr>
<tr>
<td>Param1</td>
<td>AKeyID</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Upper byte is always 0x00.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lower byte is the pointer to the key.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Legal values: 0x00 to 0x0F, 0xFF.</td>
</tr>
<tr>
<td>Param2</td>
<td>Usage</td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Authentication usage restrictions. Ignored if Mode bits 0 and 1 are 00b or 10b.</td>
</tr>
<tr>
<td>Data</td>
<td>InMac</td>
<td>0 or 16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Input MAC to be verified (see Appendix I.3, MAC Generation).</td>
</tr>
</tbody>
</table>
Table 7-4. Output parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReturnCode</td>
<td>1</td>
<td>Upon success, 0x00 will be returned. Any command execution or validation failure generates a nonzero error code, per Section 6.3, ReturnCode.</td>
</tr>
<tr>
<td>OutMac</td>
<td>0 or 16</td>
<td>If an output MAC generation was required (and any optional input MAC verification succeeded), then a 16-byte MAC will be returned.</td>
</tr>
</tbody>
</table>

The command and response packet is transmitted as a block, beginning with the Count and ending with a packet Checksum. This block format is described in Section 6.1, Command Block and Packet.
7.2 AuthCheck Command

The AuthCheck command is used to check the OutMAC generated by the Auth command or the Counter command on a second ATAES132 device. This command cannot check MACs created by other commands.

To use this command, the Nonce must be identical on both devices (see Section 7.20.1, Nonce Synchronization) and the MacCount must have the same value. Both devices must also contain identical key values, but it is not necessary for the KeyID on the origin device to match the KeyID on the destination device. In this section, the device that generates the MAC is referred to as the origin device, and the device that checks the MAC is referred to as the destination device.

If Mode bit 5, 6, or 7 is 1b, then the associated Usage Counter, SerialNum Register value, or the first four bytes of the SmallZone Register in the SecondBlock field must match the values on the origin device. The ManufacturingID Register must be identical on both devices, since it is always included in the MAC calculation.

A valid Nonce is required to run the AuthCheck command. If the KeyConfig[MacKeyID].RandomNonce bit is 1b, then the Nonce must be random.

The AuthCheck command always sets the AuthComplete status flag to NoAuth.

<table>
<thead>
<tr>
<th>Table 7-5. Input Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>Opcode</td>
</tr>
<tr>
<td>Mode</td>
</tr>
<tr>
<td>Param1</td>
</tr>
<tr>
<td>Param2</td>
</tr>
<tr>
<td>Data1</td>
</tr>
<tr>
<td>Data2</td>
</tr>
<tr>
<td>Data3</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 7-6. Output Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
</tr>
<tr>
<td>ReturnCode</td>
</tr>
</tbody>
</table>

The command and response packet is transmitted as a block, beginning with the Count and ending with a packet Checksum. This block format is described in Section 6.1, Command Block and Packet.
7.3 **AuthCompute Command**

The AuthCompute command is used to compute a MAC that will be used to execute the Auth command or the Counter command on a second ATAES132 device.

To use this command, the Nonce must be identical on both devices (see Section 7.20.1, Nonce Synchronization) and the MacCount must have the same value. Both devices must also contain identical key values, but it is not necessary for the KeyID on the origin device to match the KeyID on the destination device. In this section, the device that generates the MAC is referred to as the origin device, and the device that checks the MAC is referred to as the destination device.

If Mode bit 5, 6, or 7 is 1b, then the associated Usage Counter, SerialNum Register value, or the first four bytes of the SmallZone Register in the SecondBlock field must match the values on the destination device. The ManufacturingID Register must be identical on both devices, since it is always included in the MAC calculation.

A valid Nonce is required to run the AuthCompute command. If the KeyConfig[MacKeyID].RandomNonce bit is 1b, then the Nonce must be random.

The AuthCompute command always sets the AuthComplete status flag to NoAuth.

<table>
<thead>
<tr>
<th>Table 7-7. Input Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td>Opcode</td>
</tr>
<tr>
<td>Mode</td>
</tr>
<tr>
<td>Param1</td>
</tr>
<tr>
<td>Param2</td>
</tr>
<tr>
<td>Data1</td>
</tr>
<tr>
<td>Data2</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Table 7-8. Output Parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Name</strong></td>
</tr>
<tr>
<td>ReturnCode</td>
</tr>
<tr>
<td>OutMac</td>
</tr>
</tbody>
</table>

The command and response packet is transmitted as a block, beginning with the Count and ending with a packet Checksum. This block format is described in Section 6.1, Command Block and Packet.
7.4 BlockRead Command

The BlockRead command reads 1 to 32 bytes of plaintext data from a User Zone or the Configuration Memory. This command differs from the standard Serial EEPROM Read commands, since it can read the Configuration Memory. In addition, this command returns an error code if the Read is unsuccessful. No encryption is performed by the BlockRead command; the EncRead command must be used for encrypted reads (see Section 7.9, EncRead Command).

The BlockRead command can only read data from a single EEPROM page; the requested data cannot cross page boundaries (see Appendix B.2, EEPROM Page Boundary). All bytes within the Configuration Memory can be read with the BlockRead command. If any part of the requested data lies in unimplemented or illegal memory, the command will generate an error code. The Key Memory can never be read under any circumstances; any attempt to read the Key Memory will generate an error code.

User Zone access is dependent upon the value of the EncRead and AuthRead bits of the ZoneConfig[UZ] register. If ZoneConfig[UZ].AuthRead is 0b, then BlockRead can read the user zone. If ZoneConfig[UZ].AuthRead is 1b, then BlockRead can only be used to access the user zone if the authentication requirement has been satisfied. If ZoneConfig[UZ].EncRead is 1b, then BlockRead can never be used to access the user zone. A single BlockRead command can read data from only a single User Zone; the requested data cannot span multiple user zones or multiple EEPROM pages.

Table 7-9. Input Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>BlockRead</td>
<td>1</td>
</tr>
<tr>
<td>Mode</td>
<td>Mode</td>
<td>1</td>
</tr>
<tr>
<td>Param1</td>
<td>Address</td>
<td>2</td>
</tr>
<tr>
<td>Param2</td>
<td>Count</td>
<td>2</td>
</tr>
</tbody>
</table>

Table 7-10. Output Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReturnCode</td>
<td>1</td>
<td>Upon success, 0x00 will be returned. Any command execution or validation failure generates a nonzero error code, per Section 6.3, ReturnCode.</td>
</tr>
<tr>
<td>OutData</td>
<td>0 – 32</td>
<td>Output data (cleartext).</td>
</tr>
</tbody>
</table>

The command and response packet is transmitted as a block, beginning with the Count and ending with a packet Checksum. This block format is described in Section 6.1, Command Block and Packet.
7.5 Counter Command

The Counter command reads or increments the internal, high endurance counters. Each counter can increment up to a value of 2,097,151 using the Count command, after which they can no longer be changed. See Appendix H, Understanding Counters for additional counter information.

Table 7-11. Counter command options

<table>
<thead>
<tr>
<th>Mode bit 1</th>
<th>Mode bit 0</th>
<th>Description</th>
<th>InMAC</th>
<th>OutMAC</th>
</tr>
</thead>
<tbody>
<tr>
<td>1b</td>
<td>1b</td>
<td>Read Counter with MAC</td>
<td>Prohibited</td>
<td>Generated</td>
</tr>
<tr>
<td>0b</td>
<td>1b</td>
<td>Read Counter, No MAC</td>
<td>Prohibited</td>
<td>No</td>
</tr>
<tr>
<td>1b</td>
<td>0b</td>
<td>Increment Counter with MAC</td>
<td>Required</td>
<td>No</td>
</tr>
<tr>
<td>0b</td>
<td>0b</td>
<td>Increment Counter, No MAC</td>
<td>Prohibited</td>
<td>No</td>
</tr>
</tbody>
</table>

The CounterConfig[CntID].RequireMAC Register bit determines if InMAC is required when incrementing the counter (see Section 4.4, Counter Configuration). If CounterConfig[CntID].RequireMAC = 1b, then InMAC is required, and so Mode bit 1 must be set to 1b when incrementing the counter. If CounterConfig[CntID].RequireMAC is 0b, then InMAC is prohibited, and Mode bit 1 must be set to 0b.

If a MAC is required or generated, then a valid Nonce is required to run the Counter command. If the KeyConfig[KeyID].RandomNonce bit is set for the authorizing key, then the Nonce must be random.

The AuthCompute command can be used to generate InMac (see Section 7.3, AuthCompute Command). The AuthCheck command can be used to validate OutMac (see Section 7.2, AuthCheck Command).

Table 7-12. Input Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>Counter</td>
<td>1&lt;br&gt;Bit 0: 1b = Read the Counter.&lt;br&gt;0b = Increment the Counter.</td>
</tr>
<tr>
<td>Mode</td>
<td>Mode</td>
<td>1&lt;br&gt;Bit 0: 1b = Read the Counter.&lt;br&gt;0b = Increment the Counter.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 1: 1b = InMAC is included in the input packet if bit 0 is 0b, or OutMAC is generated if bit 0 is 1b.&lt;br&gt;0b = Neither the input nor output packets include a MAC.&lt;br&gt;Bits 2 to 4: Reserved. Must be 0b.&lt;br&gt;Bit 5: 1b = Include the Usage Counter associated with the key(1) used to generate the MAC.&lt;br&gt;Bit 6: 1b = Include SerialNum in the MAC.&lt;br&gt;Bit 7: 1b = Include the first four bytes of SmallZone in the MAC.</td>
</tr>
<tr>
<td>Param1</td>
<td>CountID</td>
<td>2&lt;br&gt;Upper byte is always 0x00. Upper nibble of lower byte is always 0x0.&lt;br&gt;Lower nibble of lower byte is the counter to be queried.</td>
</tr>
<tr>
<td>Param2</td>
<td>Zero</td>
<td>2&lt;br&gt;Always 0x0000.</td>
</tr>
<tr>
<td>Data</td>
<td>InMac</td>
<td>0 or 16&lt;br&gt;Integrity MAC for the counter increment operation.</td>
</tr>
</tbody>
</table>

Note: 1. The MAC is generated using the key identified by the KeyID in CounterConfig[CountID].IncrID for increment operations, or the KeyID in CounterConfig[CountID].MacID for Counter Read operations. The Usage Counter included in the MAC when Mode bit 5 is 1b is identified by the CntId stored in KeyConfig[KeyID].CounterNum for the key used to generate the MAC.
### Table 7-13. Output Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReturnCode</td>
<td>1</td>
<td>Upon success, 0x00 will be returned. Any command execution or validation failure generates a nonzero error code, per Section 6.3, ReturnCode.</td>
</tr>
<tr>
<td>CountValue</td>
<td>4</td>
<td>The current value of the Counter.</td>
</tr>
<tr>
<td>OutMac</td>
<td>0 or 16</td>
<td>Integrity MAC for the Counter Read operation.</td>
</tr>
</tbody>
</table>

The command and response packet is transmitted as a block, beginning with the Count and ending with a packet Checksum. This block format is described in Section 6.1, Command Block and Packet.

The equivalent decimal value of the Counter can be determined using the following equation:

\[
\text{CountValue} = (\text{BinCount} \times 32) + (\text{CountFlag}/2) \times 8 + \text{Lin2Bin}(\text{LinCount})
\]

Here, Lin2Bin defines a function that converts a linear counter value to corresponding binary value. 0xFFFF converts to 0, 0xFFFFE converts to one, and so on, up to 0x8000 which converts to 15.
7.6 Crunch Command

The Crunch command processes a seed value and returns the result within a specified time. The command provides a 16-byte input seed, which is combined with the ManufacturingID Register and processed with the internal hardware crunch calculator. The calculation is performed within a specified time period.

The Host system should read the response within a few milliseconds after the response is specified to be available and compare the returned value to the expected result to determine if authentic Atmel hardware is present. The crunch algorithm is proprietary, and is available only in authentic Atmel hardware.

The Crunch command does not use the AES engine or the Nonce. Executing the Crunch command does not change the authentication status or cryptographic state of the device.

7.6.1 Crunch Response Time

The response to the Crunch command is available after a period of time that is dependent on the Count field value. A large Count value requires more time to process than a small Count value. The expected response time for the Crunch command is computed using the following equation:

\[ \frac{(\text{count} \times 256) + 600}{1.25 \text{ microseconds}} \]

Table 7-14. Input Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>Crunch</td>
<td>1, 0x0B</td>
</tr>
<tr>
<td>Mode</td>
<td>Mode</td>
<td>1, Must be 0x00</td>
</tr>
<tr>
<td>Param1</td>
<td>Count</td>
<td>2, Upper byte is always 0x00.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Lower byte is the iteration count for the crunch engine.</td>
</tr>
<tr>
<td>Param2</td>
<td>Zero</td>
<td>2, Always 0x0000.</td>
</tr>
<tr>
<td>Data</td>
<td>Seed</td>
<td>16, Input seed.</td>
</tr>
</tbody>
</table>

Table 7-15. Output Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReturnCode</td>
<td>1</td>
<td>Upon success, 0x00 will be returned. Any command execution or validation failure generates a nonzero error code, per Section 6.3, ReturnCode.</td>
</tr>
<tr>
<td>Result</td>
<td>16</td>
<td>Result out.</td>
</tr>
</tbody>
</table>

The command and response packet is transmitted as a block, beginning with the Count and ending with a packet Checksum. This block format is described in Section 6.1, Command Block and Packet.
7.7 DecRead Command

The DecRead command is used to check the OutMAC generated by an EncRead command on a second ATAES132 device. If the MAC matches, then the 1 to 16 bytes of data is returned to the Host in the DecRead response.

To use this command, the Nonce must be identical on both devices (see Section 7.20.1, Nonce Synchronization) and the MacCount must have the same value. Both devices must also contain identical key values, but it is not necessary for the KeyID on the origin device to match the KeyID on the destination device. In this section, the device that encrypts the data and generates the MAC is referred to as the origin device, and the device that checks the MAC is referred to as the destination device.

If Mode bit 5, 6, or 7 is 1b, then the associated Usage Counter, SerialNum Register value, or the first four bytes of the SmallZone Register in the SecondBlock field, must match the values on the origin device. The ManufacturingID Register must be identical on both devices, since it is always included in the MAC calculation.

A valid Nonce is required to run the DecRead command. If the KeyConfig[DKeyID].RandomNonce bit is 1b, then the Nonce must be random.

Table 7-16. Input Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>DecRead</td>
<td>1</td>
</tr>
<tr>
<td>Mode</td>
<td>Mode</td>
<td>1 Always 0x0000.</td>
</tr>
<tr>
<td>Param1</td>
<td>DKeyID</td>
<td>2 Upper byte is always 0x00. Lower byte is the pointer to the decrypt key. Legal values: 0x00 to 0xF, 0xFF.</td>
</tr>
<tr>
<td>Param2</td>
<td>Count</td>
<td>2 Upper byte is always 0x00. Lower byte is the number of data bytes to be decrypted.</td>
</tr>
<tr>
<td>Data1</td>
<td>FirstBlock</td>
<td>6 The value of this field must match the first authenticate-only block used when executing the EncRead command on the origin device.</td>
</tr>
<tr>
<td>Data2</td>
<td>SecondBlock</td>
<td>16 The value of this field must match the second authenticate-only block used when executing the EncRead command on the origin device. If Mode bits 5, 6, and 7 are 0b, then this field must be present, but is ignored.</td>
</tr>
<tr>
<td>Data3</td>
<td>InMac</td>
<td>16 Integrity MAC for the input data.</td>
</tr>
<tr>
<td>Data4</td>
<td>InData</td>
<td>16 Input data (ciphertext) to be decrypted.</td>
</tr>
</tbody>
</table>

Table 7-17. Output Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReturnCode</td>
<td>1</td>
<td>Upon success, 0x00 will be returned. Any command execution or validation failure generates a nonzero error code, per Section 6.3, ReturnCode.</td>
</tr>
<tr>
<td>OutData</td>
<td>1 to 16</td>
<td>Decrypted (plaintext) output data.</td>
</tr>
</tbody>
</table>

The command and response packet is transmitted as a block, beginning with the Count and ending with a packet Checksum. This block format is described in Section 6.1, Command Block and Packet.
7.8 Decrypt Command

The Decrypt command accepts 16 or 32 bytes of ciphertext, decrypts the data, verifies the MAC, and returns the decrypted data if the MAC matches. If the MAC does not match, then an error code is returned.

The Decrypt command has two operating modes:

- Normal Decryption Mode
- Client Decryption Mode

The Client Decryption mode decrypts packets encrypted by an ATAES132 device. The Normal Decryption mode decrypts packets generated by a cryptographic Host. It cannot decrypt packets encrypted by an ATAES132 device.

- If the DKeyID is VolatileKey (see Section 4.3, VolatileKey Configuration), the VolUsage.DecryptOK must be 1b when VolatileKey was loaded.
- If the DKeyID is not VolatileKey, the KeyConfig[DKeyID].ExternalCrypto bit must be 1b.
- If the KeyConfig[DKeyID].AuthKey bit is 1b, prior authentication must be performed using the KeyID stored in KeyConfig[DKeyID].LinkPointer.

A valid Nonce is required to run the Decrypt command. If the KeyConfig[DKeyID].RandomNonce bit is 1b, then the Nonce must be random.

7.8.1 Client Decryption Mode

In the Client Decryption mode, the Decrypt command can be used to decrypt packets encrypted by the ATAES132 (either another device, or by the same device at a later time) using the Encrypt command (see Section 7.10, Encrypt Command). All of the following requirements must be satisfied:

1. The device performing the Encrypt operation (the Encrypt Device) and the device performing the Decrypt operation (the Decrypt Device) must contain identical keys.
2. The KeyID of the key used by the Encrypt Device (called EKeyID) must be known. EKeyID is passed to the Decrypt Device in the upper byte of Decrypt Param1 for use in the MAC calculation.
3. The Nonce used by the Encrypt Device must be known. The Nonce is passed to the Decrypt Device using the Nonce command with Mode bit 0 = 0b (see Section 7.19, Nonce Command), or is synchronized with the Encrypt Device using the procedure in Section 7.20.1, Nonce Synchronization.
4. The lower byte of the Count (Encrypt Param2) used by the Encrypt Device must identical to the value used in the lower byte of Decrypt Param2 by the Decrypt Device. (This is used in the MAC calculation.)
5. The MacCount of the Encrypt Device (called EMacCount) must be known. EMacCount is passed to the Decrypt Device in the upper byte of Decrypt Param2 for use in the Data Decryption operation.
6. The Encrypt/Decrypt command mode bits on both devices must be identical. Mode bit 5 must be 0b. Mode bit 6 must be 0b, unless a single device is performing both the Encrypt and the Decrypt operations. Mode bit 7 can be 1b if the first four bytes of SmallZone are identical on both the Encrypt and the Decrypt Devices.
7. The Decrypt Device KeyConfig[DKeyID] must have ExternalCrypto = 1b and RandomNonce = 0b for the KeyID used for decryption if the Nonce is passed using the Nonce command with Mode bit 0 = 0b.
8. The Encrypt Device KeyConfig[EKeyID] must have ExternalCrypto = 1b and RandomNonce = 1b for the KeyID used for encryption (the EKeyID).

If these conditions are satisfied, then packets encrypted on the Encrypt Device can be decrypted on the Decrypt Device. If a single ATAES132 will be used to encrypt packets for later decryption, then the same key value must be stored in two appropriately configured key registers to allow all of the requirements above to be satisfied.
### Table 7-18. Input Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Decrypt</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>0x07</td>
</tr>
<tr>
<td>Mode</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Mode</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bits 0 to 4: Reserved. Must be 0b. Bits 5: 1b = Include the Usage Counter associated with the encryption key in the MAC. Bit 6: 1b = Include SerialNum in the MAC. Bit 7: 1b = Include the first four bytes of SmallZone in the MAC.</td>
</tr>
<tr>
<td>Param1</td>
<td>DKeyID</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Normal Decryption Mode:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Upper byte is always 0x00.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Lower byte is the KeyID of the decrypt key.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Client Decryption Mode:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Upper byte is the EKeyID.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Lower byte is the KeyID of the decrypt key.</td>
</tr>
<tr>
<td>Param2</td>
<td>Count</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>2</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Normal Decryption Mode:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Upper byte is always 0x00.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Lower byte is the number of bytes to be returned after decryption.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Client Decryption Mode:</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Upper byte is the EMacCount.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>• Lower byte is the number of bytes to be returned after decryption (see Section 7.8.1, Client Decryption Mode).</td>
</tr>
<tr>
<td>Data1</td>
<td>InMac</td>
<td>16</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Integrity MAC for the input data.</td>
</tr>
<tr>
<td>Data2</td>
<td>InData</td>
<td>16 or 32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Input data (ciphertext) to be decrypted.</td>
</tr>
</tbody>
</table>

### Table 7-19. Output Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReturnCode</td>
<td></td>
<td>Upon success, 0x00 will be returned. Any command execution or validation failure generates a nonzero error code, per Section 6.3, ReturnCode.</td>
</tr>
<tr>
<td>OutData</td>
<td></td>
<td>1 – 32</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Decrypted (plaintext) output data.</td>
</tr>
</tbody>
</table>

The command and response packet is transmitted as a block, beginning with the Count and ending with a packet Checksum. This block format is described in Section 6.1, Command Block and Packet.
7.9 EncRead Command

EncRead reads 1 to 32 bytes of encrypted data from User Memory, along with an integrity MAC. The EncRead command only performs encrypted reads; the BlockRead command is used for unencrypted reads (see Section 7.4, BlockRead Command).

The ZoneConfig[UZ].EncRead bit determines if a user zone can be accessed with the EncRead command. If the ZoneConfig[UZ].EncRead bit is 1b, then the EncRead command can read the user zone if the access requirements have been satisfied. A single EncRead command reads data from a single user zone; the requested data cannot span multiple user zones. A single EncRead command reads data from a single EEPROM page; the requested data cannot cross page boundaries (see Appendix B.2, EEPROM Page Boundary).

If ZoneConfig[UZ].Auth is 1b, then prior authentication is required with the following restrictions:

- The Auth command Usage.ReadOK bit must be 1b.
- The Authentication Key AKeyID must match ZoneConfig[UZ].AuthID.
- The Auth command must be run in Inbound Only Authentication or Mutual Authentication mode.
- A valid Nonce is required to run the EncRead command. If KeyConfig[KeyID].RandomNonce for the read key is 1b, then the Nonce must be random.

The DecRead command can be used to validate OutMac and decrypt up to 16 bytes of data (see Section 7.7, DecRead Command).

7.9.1 Configuration Memory Signature

The EncRead command cannot be used to read the Configuration Memory. Only the BlockRead command can be used to read the Configuration Memory. Any attempt to read any address in the Configuration Memory with the EncRead command will activate the Configuration Memory Signature Generation mode.

The Configuration Memory Signature is an AES-CCM MAC generated over the entire Configuration Memory, as described in Appendix I.17, EncRead Command Configuration Memory Signature MAC. A valid Nonce is required to run the EncRead command in Configuration Memory Signature Generation mode. If KeyConfig[00].RandomNonce is 1b, then the Nonce must be random. KeyID 00 is always used to generate the Configuration Memory Signature.

The Configuration Memory Signature Generation mode is intended to be used during secure personalization of the ATAES132 device. The signature can be used to validate the contents of the Configuration Memory prior to programming secret data into other portions of the EEPROM.

7.9.2 Key Memory Signature

The EncRead command cannot be used to read the Key Memory. The Key Memory can never be read. Any attempt to read any address in the Key Memory with the EncRead command will activate the Key Memory Signature Generation mode; however, this signature can be generated only once per unit.

The Key Memory Signature is an AES-CCM MAC generated over all 16 key registers, as described in Appendix I.18, EncRead Command Key Memory Signature MAC.

A valid Nonce is required to run the EncRead command in Key Memory Signature Generation mode. If KeyConfig[00].RandomNonce is 1b, then the Nonce must be random. KeyID 00 is always used to generate the Key Memory Signature.

The Key Memory Signature Generation mode is intended to be used during secure personalization of the ATAES132. The signature can be used to validate the contents of the Key Memory before locking the Key Memory.
### Table 7-20. Input Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>EncRead</td>
<td>1 0x04</td>
</tr>
<tr>
<td>Mode</td>
<td>Mode</td>
<td>Bits 0 to 4: Reserved. Must be 0b. Bit 5: 1b = Include the Usage Counter associated with the ZoneConfig[UZ].ReadID key in the MAC. Bit 6: 1b = Include SerialNum in the MAC. Bit 7: 1b = Include the first four bytes of SmallZone in the MAC.</td>
</tr>
<tr>
<td>Param1</td>
<td>Address</td>
<td>2 The address of data to be read.</td>
</tr>
<tr>
<td>Param2</td>
<td>Count</td>
<td>2 Upper byte is always 0x00. Lower byte is the number of bytes to read.</td>
</tr>
<tr>
<td>Data</td>
<td>—</td>
<td>0</td>
</tr>
</tbody>
</table>

### Table 7-21. Output Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReturnCode</td>
<td>1</td>
<td>Upon success, 0x00 will be returned. Any command execution or validation failure generates a nonzero error code, per Section 6.3, ReturnCode.</td>
</tr>
<tr>
<td>OutMac</td>
<td>16</td>
<td>Integrity MAC for the output data.</td>
</tr>
<tr>
<td>OutData</td>
<td>16 or 32</td>
<td>Encrypted output data (ciphertext).</td>
</tr>
</tbody>
</table>

The command and response packet is transmitted as a block, beginning with the Count and ending with a packet Checksum. This block format is described in Section 6.1, Command Block and Packet.
7.10 Encrypt Command

The Encrypt command accepts 1 to 32 bytes of plaintext, encrypts the data, and generates an integrity MAC. The encrypted data and OutMAC are returned to the system.

The Encrypt command can be used to encrypt packets for decryption by the same or another ATAES132, if the requirements described in Section 7.8.1, Client Decryption Mode are satisfied.

- If the EKeyID specifies a key in the Key Memory, the KeyConfig[EKeyID].ExternalCrypto bit must be 1b.
- If the KeyConfig[EKeyID].AuthKey bit is 1b, then prior authentication is required using the KeyID stored in KeyConfig[EKeyID].LinkPointer.
- If the EKeyID specifies the VolatileKey (see Section 4.3, VolatileKey Configuration), the VolUsage.EncryptOK must be set to 01b, 10b, or 11b.
- If the VolUsage.EncryptOK bits are set to 10b or 11b, then prior authentication is required using VolatileKey prior to execution of the Encrypt command.

A valid Nonce command is required to run the Encrypt command. If the KeyConfig[EKeyID].RandomNonce bit is set for the encryption key, then the Nonce must be random.

Table 7-22. Input Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>Encrypt</td>
<td>1</td>
</tr>
<tr>
<td>Mode</td>
<td>Mode</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td>Bits 0 to 4: Reserved. Must be 0b.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bit 5: 1b = Include the Usage Counter associated with the encryption key in the MAC.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bit 6: 1b = Include SerialNum in the MAC.</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Bit 7: 1b = Include the first four bytes of SmallZone in the MAC.</td>
<td></td>
</tr>
<tr>
<td>Param1</td>
<td>EKeyID</td>
<td>2</td>
</tr>
<tr>
<td>Param2</td>
<td>Count</td>
<td>2</td>
</tr>
<tr>
<td>Data</td>
<td>InData</td>
<td>1 – 32</td>
</tr>
</tbody>
</table>

Table 7-23. Output Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReturnCode</td>
<td>1</td>
<td>Upon success, 0x00 will be returned. Any command execution or validation failure generates a nonzero error code, per Section 6.3, ReturnCode.</td>
</tr>
<tr>
<td>OutMac</td>
<td>16</td>
<td>Integrity MAC for the output data.</td>
</tr>
<tr>
<td>OutData</td>
<td>16 or 32</td>
<td>Encrypted data (ciphertext).</td>
</tr>
</tbody>
</table>

The command and response packet is transmitted as a block, beginning with the Count and ending with a packet Checksum. This block format is described in Section 6.1, Command Block and Packet.
### 7.11 EncWrite Command

The EncWrite command decrypts the ciphertext input data, verifies the input MAC, and then writes 1 to 32 bytes to a User Zone or 16 bytes to Key Memory.

The ZoneConfig[UZ].EncWrite bit determines if a User Zone must be accessed with the EncWrite command. If the ZoneConfig[UZ].EncWrite bit is 1b, then the EncWrite command must be used to write the user zone if the access requirements have been satisfied. If the ZoneConfig[UZ].EncWrite bit is 0b, then a Write command or EncWrite command can be used to write the User Zone. A single EncWrite command writes data to a single User Zone; the data cannot span multiple User Zones. A single EncWrite command writes data to a single EEPROM page; the data cannot cross page boundaries (see Appendix B.2, EEPROM Page Boundary).

If ZoneConfig[UZ].Auth is 1b, then prior authentication is required with the following restrictions:

- The Auth command Usage.WriteOK bit must be 1b.
- The Authentication Key (AKeyID) must match ZoneConfig[UZ].AuthID.
- The Auth command must be run in Inbound-Only Authentication or Mutual Authentication mode.
- A valid Nonce is required to run the EncWrite command. If KeyConfig[KeyID].RandomNonce for the write key is 1b, then the Nonce must be random.

#### 7.11.1 Encrypted Key Writes

When EncWrite is used to write the Key Memory prior to locking, the key data must be encrypted using KeyID 00. The input MAC is also calculated using KeyID 00. Writes to Key Memory must be 16 bytes in length and begin at the starting address of the key.

If LockKeys has a value of 0x55 and the EncWrite address points to Key Memory, then Key Personalization mode is selected. In key Personalization mode, the following requirements are in effect:

- The Count field value must be 16.
- The address must match the starting address of the Key Register.
- The input data must be encrypted with the current value in KeyID 00. If KeyConfig[WriteID].RandomNonce is 1b, then the Nonce must be random (See Section 7.19, Nonce Command).
- The input MAC must be generated with the current value in KeyID 00. The input MAC will be verified.

If the Key Memory is locked, then the new key data is encrypted with the current value of the key being written. The key can be updated only if all of the following requirements are satisfied:

- The corresponding KeyConfig[KeyID].ChangeKeys bit is set to 1b (see Section 4.2, Key Configuration).
- The Count field value must be 16.
- The address must match the starting address of the Key Register.
- The input data must be encrypted with the current value of the Key. If KeyConfig[WriteID].RandomNonce is 1b then Nonce be random (See Section 7.19, Nonce Command).
- The input MAC must be generated with the current value of the Key. The input MAC will be verified (See Section 7.18, Lock Command).
Table 7-24. Input Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>EncWrite</td>
<td>1</td>
</tr>
<tr>
<td>Mode</td>
<td>Mode</td>
<td>1</td>
</tr>
<tr>
<td>Param1</td>
<td>Address</td>
<td>2</td>
</tr>
<tr>
<td>Param2</td>
<td>Count</td>
<td>2</td>
</tr>
<tr>
<td>Data1</td>
<td>InMac</td>
<td>16</td>
</tr>
<tr>
<td>Data2</td>
<td>InData</td>
<td>16 or 32</td>
</tr>
</tbody>
</table>

Table 7-25. Output Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReturnCode</td>
<td>1</td>
<td>Upon success, 0x00 will be returned. Any command execution or validation failure generates a nonzero error code, per Section 6.3, ReturnCode.</td>
</tr>
</tbody>
</table>

The command and response packet is transmitted as a block, beginning with the Count and ending with a packet Checksum. This block format is described in Section 6.1, Command Block and Packet.
7.12 INFO Command

The INFO command reads various information about the device from the internal registers. Param1 selects the information to read. Operation of this command does not require knowledge of any secrets.

Table 7-26. Selector Field Coding (Param1)

<table>
<thead>
<tr>
<th>Selector</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>MacCount</td>
<td>Read the MacCount Register. The first byte is always 0x00; the second byte is the MacCount value.</td>
</tr>
<tr>
<td>0x0005</td>
<td>AuthStatus</td>
<td>Read the Authentication Status Register. Returns 0xFFFF to indicate that the AuthComplete status flag = NoAuth. If the AuthComplete status flag = YesAuth, then the info returns the AKeyID as 0x00KK, where KK is the Authentication Key ID.</td>
</tr>
<tr>
<td>0x0006</td>
<td>DeviceNum</td>
<td>Read the DeviceNum Register. The first byte is the Atmel device code, which is unique to this Atmel catalog number. The second byte provides the device revision number See Table 7-29 for DeviceNum codes.</td>
</tr>
</tbody>
</table>
| 0x000C   | ChipState | Read the ChipState Device State Register:  
|          |         | • 0x0000 indicates ChipState = Active  
|          |         | • 0xFFFF indicates ChipState = Power-Up  
|          |         | • 0x5555 indicates ChipState = Wake-up from Sleep  
|          |         | See Appendix L.3, Understanding the ChipState Register. |
| All Other| Reserved | Reserved for future use. |

Table 7-27. Input Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>Info</td>
<td>1</td>
</tr>
<tr>
<td>Mode</td>
<td>Mode</td>
<td>1</td>
</tr>
<tr>
<td>Param1</td>
<td>Selector</td>
<td>2</td>
</tr>
<tr>
<td>Param2</td>
<td>Zero</td>
<td>2</td>
</tr>
<tr>
<td>Data</td>
<td>—</td>
<td>0</td>
</tr>
</tbody>
</table>

Table 7-28. Output Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReturnCode</td>
<td>1</td>
<td>Upon success, 0x00 will be returned. Any command execution or validation failure generates a nonzero error code, per Section 6.3, ReturnCode.</td>
</tr>
<tr>
<td>Result</td>
<td>2</td>
<td>Current value of the register.</td>
</tr>
</tbody>
</table>

The command and response packet is transmitted as a block, beginning with the count and ending with a packet checksum. This block format is described in Section 6.1, Command Block and Packet.

Table 7-29. DeviceNum Coding for INFO Response and DeviceNum in the Configuration Memory Register

<table>
<thead>
<tr>
<th>Description</th>
<th>INFO DeviceNum</th>
<th>DeviceNum Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>Production Parts</td>
<td>0x0A05</td>
<td>0x0A</td>
</tr>
</tbody>
</table>
### 7.13 KeyCreate Command

The KeyCreate command generates a 16-byte random number, and stores it in either the Key Memory or in the VolatileKey Register. The newly generated key is then encrypted with the parent key and returned to the Host along with a MAC.

If Mode bit 0 is 1b, then the target key is in the Key Memory:

- KeyConfig[childKeyID].Child must be 1b.
- The KeyCreate command KeyID field contains the ChildKeyID.
- KeyConfig[childKeyID].LinkPointer contains the ParentKeyID.

If Mode bit 0 is 0b, then the target key is VolatileKey:

- KeyConfig[parentKeyID].Parent must be 1b.
- The KeyCreate command KeyID field contains the ParentKeyID.
- The VolUsage field specifies VolatileKey usage restrictions, as defined in Section 4.3, VolatileKey Configuration.

If KeyConfig[parentKeyID].AuthKey bit is 1b or the KeyConfig[EKeyID].ChildAuth bit is 1b, then prior authentication is required using the KeyID stored in KeyConfig[parentKeyID].LinkPointer.

InMAC and OutMAC are both calculated using the parent key (ParentKeyID). If KeyConfig[childKeyID].ChildMac is 1b, then an InMAC must be provided; otherwise, InMAC will be ignored.

A valid Nonce is required to run the KeyCreate command. If the KeyConfig[parentKeyID].RandomNonce bit is 1b, then the Nonce must be random.

If the LockConfig Register is unlocked (0x55), then the random number generator is latched in Test mode, and the KeyCreate command will generate nonrandom key values. If the LockConfig Register is locked (0x00), then the RNG generates random numbers and the KeyCreate command functions normally.

The KeyImport command can be used to load a key generated by the KeyCreate command (see Section 7.14, KeyImport Command).

---

**WARNING**

There is one Random Number Generator (RNG) Seed Register in the EEPROM memory, which is used by the KeyCreate, Nonce, and Random commands. The RNG Seed Register is subject to the same Write endurance limitations as the other bytes in the EEPROM (see Section 9.2, Reliability for the EEPROM specifications). The application developer must not exceed the Write endurance limit.
### Table 7-30. Input Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td></td>
<td>Bit 0: 1b = Key load target is Key Memory. 0b = Target is VolatileKey (see Section 4.3, VolatileKey Configuration).</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 1: 0b = Update the EEPROM RNG Seed Register prior to key generation. (1)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1b = Generate the key using the existing RNG Seed.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 2: 1b = A key equivalent to what the KeyCreate InMac would be is generated. Including an InMac with the KeyCreate command is not required.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bits 3-4: Reserved. Must be zero.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 5: 1b = Include the Usage Counter associated with the ParentKeyID in the MAC.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 6: 1b = Include SerialNum in the MAC.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Bit 7: 1b = Include the first four bytes of SmallZone in the MAC.</td>
</tr>
<tr>
<td>Mode</td>
<td>Mode</td>
<td>1</td>
</tr>
<tr>
<td>Param1</td>
<td>KeyID</td>
<td>2 Upper byte is always 0x00. Lower byte is the ChildKeyID for Key Memory loads or the ParentKeyID for VolatileKey loads.</td>
</tr>
<tr>
<td>Param2</td>
<td>VolUsage</td>
<td>2 Usage restrictions for VolatileKey if Mode bit 0 is 0b (see Section 4.3).</td>
</tr>
<tr>
<td>Data</td>
<td>InMac</td>
<td>0 or 16 Optional input MAC (see above).</td>
</tr>
</tbody>
</table>

Note: 1. The RNG Seed Register in the EEPROM will be updated automatically if Mode bit 1 = 0b, unless the Seed Register was previously updated after the most recent Power-On Reset, Wake from the Sleep state, Reset command, or Tamper Event. Updating the RNG Seed Register increases the randomness of the keys generated by the KeyCreate command; however, the EEPROM Write Endurance specification must be respected.

### Table 7-31. Output Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReturnCode</td>
<td>1</td>
<td>Upon success, 0x00 will be returned. Any command execution or validation failure generates a nonzero error code (see Section 6.3, ReturnCode).</td>
</tr>
<tr>
<td>OutMac</td>
<td>16</td>
<td>Output MAC for the encrypted key.</td>
</tr>
<tr>
<td>OutData</td>
<td>16</td>
<td>Encrypted key value (ciphertext).</td>
</tr>
</tbody>
</table>

The command and response packet is transmitted as a block, beginning with the Count and ending with a packet Checksum. This block format is described in Section 6.1, Command Block and Packet.
7.14 KeyImport Command

The KeyImport command accepts 16 bytes of ciphertext, decrypts the key, verifies the MAC, and stores the key in the Key Memory or in the VolatileKey Register. The source of the encrypted key is the KeyCreate command.

- If TargetKeyId specifies that the target key is stored in the Key Memory:
  - The KeyConfig[TargetKeyId].ImportOK bit must be 1b.
  - KeyConfig[TargetKeyId].LinkPointer contains the decrypt KeyID.
  - The KeyImport command DKeyId field value is ignored.

- If the KeyConfig[decrypt KeyID].AuthKey bit is 1b, then prior authentication is required using the KeyID stored in KeyConfig[decrypt KeyID].LinkPointer.

- If TargetKeyId specifies that the target key is VolatileKey (see Section 4.3, VolatileKey Configuration):
  - The KeyConfig[DKeyId].Parent bit must be 1b.
  - The KeyImport command DKeyId field contains the decrypt KeyID.

- If the KeyConfig[DKeyId].AuthKey bit is 1b, then prior authentication is required using the KeyID stored in KeyConfig[DKeyId].LinkPointer.

To use this command, the Nonce must be identical on both devices (see Section 7.20.1, Nonce Synchronization) and the MacCount must have the same value. Both devices must also contain identical key values, but it is not necessary for the encrypt KeyID on the origin device to match the decrypt KeyID on the destination device. In this section, the device that encrypts the key and generates the MAC is referred to as the origin device, and the device that checks the MAC is referred to as the destination device.

If Mode bit 5, 6, or 7 is 1b, then the associated Usage Counter, SerialNum Register value, or the first four bytes of the SmallZone Register in the SecondBlock field must match the values on the origin device. The ManufacturingID Register must be identical on both devices since it is always included in the MAC calculation.

A valid Nonce is required to run the KeyImport command. If the KeyConfig[KeyId].RandomNonce bit is 1b for the Decrypt Key, then the Nonce must be random.
Table 7-32.  Input Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>KeyImport</td>
<td>1x19</td>
</tr>
<tr>
<td>Mode</td>
<td>Mode</td>
<td>1 Bit 0: Reserved. Must be 1b. Bits 1 to 4: Reserved. Must be 0b. Bits 5 to 7: This value must match Mode bits 5, 6, and 7 value used when executing the KeyCreate command on the origin device.</td>
</tr>
<tr>
<td>Param1</td>
<td>TargetKeyID</td>
<td>2 Upper byte is always 0x00. Lower byte is the location where the decrypted key will be stored. Legal values: 0x00 to 0x0F (standard keys), 0xFF (volatile key).</td>
</tr>
<tr>
<td>Param2</td>
<td>DKeyID</td>
<td>2 Upper byte is always 0x00. If TargetKeyID = 0xFF, then lower byte is the pointer to the decrypt key. Legal values: 0x00 to 0x0F. If TargetKeyID = 0x00 to 0x0F, then this field must be present, but is ignored (see above).</td>
</tr>
<tr>
<td>Data1</td>
<td>FirstBlock</td>
<td>6 The value of this field must match the first authenticate-only block used when executing the KeyCreate command on the origin device.</td>
</tr>
<tr>
<td>Data2</td>
<td>SecondBlock</td>
<td>16 The value of this field must match the second authenticate-only block used when executing the KeyCreate command on the origin device. If Mode bits 5, 6, and 7 are 0b, then this field must be present, but is ignored.</td>
</tr>
<tr>
<td>Data3</td>
<td>InMac</td>
<td>16 MAC for the encrypted key.</td>
</tr>
<tr>
<td>Data4</td>
<td>InData</td>
<td>16 Input key (ciphertext) to be decrypted.</td>
</tr>
</tbody>
</table>

Table 7-33.  Output Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReturnCode</td>
<td>1</td>
<td>Upon success, 0x00 will be returned. Any command execution or validation failure generates a nonzero error code, per Section 6.3, ReturnCode.</td>
</tr>
</tbody>
</table>

The command and response packet is transmitted as a block, beginning with the Count and ending with a packet Checksum. This block format is described in Section 6.1, Command Block and Packet.
### 7.15 KeyLoad Command

The KeyLoad command decrypts 16 bytes of ciphertext data, verifies the MAC, and then writes the Key Memory or the VolatileKey Register.

- If Mode bit 0 specifies that the target key is stored in the Key Memory:
  - KeyConfig[ChildKeyID].Child bit must be 1b.
  - KeyLoad command KeyID field contains the ChildKeyID.
  - KeyConfig[ChildKeyID].LinkPointer contains the ParentKeyID.
- If the KeyConfig[ParentKeyID].AuthKey bit is 1b, then prior authentication is required using the KeyID stored in KeyConfig[ParentKeyID].LinkPointer.
- If Mode bit 0 specifies that the target key is VolatileKey (see Section 4.3, VolatileKey Configuration):
  - KeyConfig[ParentKeyID].Parent bit must be 1b.
  - KeyLoad command KeyID field contains the ParentKeyID.
  - VolUsage field specifies VolatileKey usage restrictions, as defined in Section 4.3.
- If the KeyConfig[ParentKeyID].AuthKey bit is 1b, then prior authentication is required using the KeyID stored in KeyConfig[ParentKeyID].LinkPointer.

A valid Nonce is required to run the KeyLoad command. If the appropriate KeyConfig[KeyID].RandomNonce bit is 1b, then the Nonce must be random.

#### Table 7-34. Input Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>KeyLoad</td>
<td>1</td>
</tr>
<tr>
<td>Mode</td>
<td>Mode</td>
<td>1</td>
</tr>
<tr>
<td>Param1</td>
<td>KeyID</td>
<td>2</td>
</tr>
<tr>
<td>Param2</td>
<td>VolUsage</td>
<td>2</td>
</tr>
<tr>
<td>Data1</td>
<td>InMac</td>
<td>16</td>
</tr>
<tr>
<td>Data2</td>
<td>InData</td>
<td>16</td>
</tr>
</tbody>
</table>

#### Table 7-35. Output Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReturnCode</td>
<td>1</td>
<td>Upon success, 0x00 will be returned. Any command execution or validation failure generates a nonzero error code, per Section 6.3, ReturnCode.</td>
</tr>
</tbody>
</table>

The command and response packet is transmitted as a block, beginning with Count and ending with a packet Checksum. This block format is described in Section 6.1, Command Block and Packet.
### 7.16 KeyTransfer Command

The KeyTransfer command copies key data from the User Memory into the VolatileKey Register or into a Key Register in the Key Memory. The KeyTransfer command allows a user zone to be utilized as an Extended Key Memory.

Keys stored in the User Memory cannot be utilized directly by the cryptographic commands; the keys must be transferred into either the VolatileKey Register or into a Key Register in the Key Memory EEPROM prior to use. The usage restrictions for keys transferred into the VolatileKey Register are transferred from the Key Data Structure when the KeyTransfer command is executed. Usage restrictions for keys transferred into the Key Memory are stored in the KeyConfig[TargetKeyID] Register; the KeyTransfer command does not alter the KeyConfig[TargetKeyID] Register.

- If KeyConfig[TargetKeyID].TransferOK is 0b, then the Key Register cannot be updated with the KeyTransfer command.
- If KeyConfig[TargetKeyID].TransferOK is 1b, then the KeyTransfer command can be used to update the Key register; the KeyConfig[TargetKeyID].LinkPointer contains the user zone number of the extended Key Memory.
- If ZoneConfig[UZ].AuthRead is 1b for the user zone number containing the Key Data Structure, then prior authentication is required using the KeyID stored in ZoneConfig[UZ].AuthID before a key can be transferred to either the VolatileKey Register or into a Key Register in the Key Memory EEPROM.

#### 7.16.1 Extended Key Memory Data Structure

When a user zone is utilized as the Extended Key Memory, the keys are stored in the 32-byte Key Data Structure, as shown in Table 7-36. The first 16 bytes contain the key value, two bytes store the VolUsage restrictions, and the remaining bytes should contain all zeros. The starting address of each Key Data Structure is required to be the first byte of a 32-byte physical page (see Appendix B.2, EEPROM Page Boundary).

| Address | 0h | 1h | 2h | 3h | 4h | 5h | 6h | 7h | 8h | 9h | Ah | Bh | Ch | Dh | Eh | Fh |
|---------|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| XX00h – XX0Fh | Key |
| XX10h – XX1Fh | VolUsage | Reserved (All bytes 0x00) |

#### Table 7-37. Input Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>KeyTransfer</td>
<td>1   0x1A</td>
</tr>
<tr>
<td>Mode</td>
<td>Mode</td>
<td>1   Must be 0x00.</td>
</tr>
<tr>
<td>Param1</td>
<td>TargetKeyID</td>
<td>2   Upper byte is always 0x00. Lower byte is the location where the key will be stored. Legal values: 0x00 to 0x0F (standard keys), 0xFF (volatile key).</td>
</tr>
<tr>
<td>Param2</td>
<td>Address</td>
<td>2   Starting address of the key data structure in User Memory.</td>
</tr>
</tbody>
</table>
Table 7-38. Output Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReturnCode</td>
<td>1</td>
<td>Upon success, 0x00 will be returned. Any command execution or validation failure generates a nonzero error code, per Section 6.3, ReturnCode.</td>
</tr>
</tbody>
</table>

The command and response packet is transmitted as a block, beginning with the Count and ending with a packet Checksum. This block format is described in Section 6.1, Command Block and Packet.
7.17 Legacy Command

The Legacy command executes a single block of the AES engine in the Electronic Code Book mode, with no input or output formatting. This is known as AES-ECB mode, and can be used to perform primitive AES encryption or decryption operations. This command does not use the Nonce Register value in the computation since the entire 16-byte AES input value comes from the input packet.

This command can be executed only if it is enabled for the device by setting ChipConfig.LegacyE to 1b and for the key by setting KeyConfig[LKeyID].LegacyOK is 1b.

Atmel recommends that any key with KeyConfig[LKeyID].LegacyOK = 1b should never be used with any other command; the Legacy command can be used to exhaustively attack the key. If the KeyConfig[LKeyID].AuthKey bit is 1b, then prior authentication is required using the KeyID stored in KeyConfig[LKeyID].LinkPointer.

Key usage limits are enforced if KeyConfig[LKeyID].CounterLimit is 1b (see Section 4.2, Key Configuration). See Appendix E.2.16, ChipConfig Register for the ChipConfig Register definition.

Table 7-39. Input Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>Legacy</td>
<td>1 0xAF</td>
</tr>
<tr>
<td>Mode</td>
<td>Mode</td>
<td>1 Must be 0x00.</td>
</tr>
<tr>
<td>Param1</td>
<td>LKeyID</td>
<td>2 Upper byte is always 0x00. Lower byte is the KeyID for the AES key.</td>
</tr>
<tr>
<td>Param2</td>
<td>Zero</td>
<td>2 Always 0x0000.</td>
</tr>
<tr>
<td>Data</td>
<td>InData</td>
<td>16 Input to the AES block (plaintext).</td>
</tr>
</tbody>
</table>

Table 7-40. Output Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReturnCode</td>
<td>1</td>
<td>Upon success, 0x00 will be returned. Any command execution or validation failure generates a nonzero error code, per Section 6.3, ReturnCode.</td>
</tr>
<tr>
<td>OutData</td>
<td>16</td>
<td>The output of the AES block (ciphertext).</td>
</tr>
</tbody>
</table>

The command and response packet is transmitted as a block beginning with the Count and ending with a packet Checksum. This block format is described in Section 6.1, Command Block and Packet.
7.18 Lock Command

The Lock command permanently locks various segments of the EEPROM, including the Configuration Memory, the Key Memory, and the SmallZone register. Key, Counter, and User Memory access restrictions are locked when the Configuration Memory is locked. SmallZone is locked independently of the other Configuration Memory registers.

The Atmel recommendation is the Key Memory be locked immediately after loading the keys.

The Configuration Memory must be locked before locking the Key Memory. Trying to lock the Key Memory before the Configuration Memory is locked will result in the Lock command failing.

Three registers in the Configuration Memory control the Lock/Unlock status of the memory segments:

1. The Configuration Memory is controlled by the LockConfig Register (see Appendix E.2.11, LockConfig Register).
2. The Key Memory is controlled by the LockKeys Register (see Appendix E.2.9, LockKeys Register).
3. The SmallZone Register is controlled by the LockSmall Register (see Appendix E.2.10, LockSmall Register).

If a Lock Control Register contains 0x55, then the memory segment is unlocked. The Lock command writes 0x00 to the specified lock register to lock the segment. The Lock Control Registers can be written only with the Lock command, but they can always be read with the BlockRead command. (See Section 7.4, BlockRead Command).

The Lock command Param2 is an optional checksum (CRC-16) generated over the memory segment being locked. The value in the Checksum field must match the CRC-16 calculated within the device for the lock operation to succeed. If the Lock command returns a LockError ReturnCode, then the Host system should rewrite the memory segment and try the lock operation again.

7.18.1 User Zone ReadOnly Activation

After the Configuration Memory is locked, the Lock command can be used to activate the ReadOnly user zone feature on appropriately configured user zones. The Lock command changes the user zone from Read/Write to read-only if the following requirements are satisfied:

- ZoneConfig[Zone].WriteMode must be 10b or 11b.
- Lock command Mode bits 0 and 1 must be set to 11b.
- The Lock command zone field contains the target user zone number (Zone).

If Lock command Mode bit 2 is 1b, then the Checksum field contains the CRC-16 of the user zone contents.

If ZoneConfig[Zone].WriteMode is 11b, then the command must include an InMAC generated using the KeyID stored in ZoneConfig[Zone].WriteID; otherwise, the MAC is ignored.

The Lock command changes the ZoneConfig[Zone].ReadOnly byte from 0x55 (Read/Write) to 0x00 when the ReadOnly feature is activated. It is not possible to change a read-only user zone to read/write after Configuration Memory is locked.
Table 7-41. Input Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>Lock</td>
<td>1</td>
</tr>
<tr>
<td>Mode</td>
<td>Mode</td>
<td>1</td>
</tr>
<tr>
<td>Param1</td>
<td>Zone</td>
<td>2</td>
</tr>
<tr>
<td>Param2</td>
<td>Checksum</td>
<td>2</td>
</tr>
<tr>
<td>Data</td>
<td>InMAC</td>
<td>0 or 16</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>Lock</td>
<td>1</td>
</tr>
<tr>
<td>Mode</td>
<td>Mode</td>
<td>Bit 0-1: 00b = Lock the SmallZone Register. 01b = Lock the Key Memory. 10b = Lock the Configuration Memory, excluding SmallZone. 11b = Set the ZoneConfig[Zone].ReadOnly byte to ReadOnly. Bit 2: 1b = Validate the memory checksum in Param2. 0b = Suppress the Checksum validation (not recommended by Atmel). Bits 3-4: Reserved. Must be 0x00. Bit 5: 1b = Include the Usage Counter associated with the ZoneConfig[Zone].WriteID key in the MAC (ignored unless Mode[0:1] is 11b). Bit 6: 1b = Include SerialNum in the MAC (ignored unless Mode[0:1] is 11b). Bit 7: 1 = Include the first four bytes of SmallZone in the MAC (ignored unless Mode[0:1] is 11b).</td>
</tr>
<tr>
<td>Param1</td>
<td>Zone</td>
<td>Upper byte is always 0x00. If Mode[0:1] is 11b, the lower byte is the user zone to be locked (see Section 7.18.1, User Zone ReadOnly Activation). For any other values of Mode[0:1], this field must be 0x0000.</td>
</tr>
<tr>
<td>Param2</td>
<td>Checksum</td>
<td>If Mode bit 2 is 1b, contains the CRC-16 checksum generated over the memory segment being locked. If Mode bit 2 is 0b, this parameter must be 0x0000.</td>
</tr>
<tr>
<td>Data</td>
<td>InMAC</td>
<td>If Mode[0:1] is 11b, contains the MAC authorizing update of ZoneConfig[Zone].ReadOnly, as described in Section 7.18.1. For all other modes, this field is ignored.</td>
</tr>
</tbody>
</table>

Table 7-42. Output Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReturnCode</td>
<td>1</td>
<td>Upon success, 0x00 will be returned. Any command execution or validation failure generates a nonzero error code, per Section 6.3, ReturnCode.</td>
</tr>
</tbody>
</table>

The command and response packet is transmitted as a block, beginning with the Count and ending with a packet Checksum. This block format is described in Section 6.1, Command Block and Packet.
7.19 Nonce Command

The Nonce command generates and/or stores a 96-bit Nonce in the SRAM Nonce Register for use by subsequent cryptographic commands. It is not necessary to generate a new Nonce before each cryptographic operation because the ATAES132 includes the MacCount in the MAC calculations (see Appendix I.1, MacCount) to guarantee uniqueness.

There are two Nonce command options:

- **Inbound Nonce**
  The InSeed value is written directly to the Nonce Register. No random number generation or cryptographic Nonce calculation is performed.
  
  Note: This option provides no defense against replay attacks or known plaintext attacks.

- **Random Nonce**
  The InSeed value is cryptographically combined with the new output of the random number generator and stored in the Nonce Register. The random number used for the Nonce calculation is returned to the Host in the response. See Appendix I.28, Nonce Command for the Nonce algorithm.

If the LockConfig Register is unlocked (0x55), then the random number generator is latched in the Test mode, and executing the Nonce command with Mode bit 0 = 1b will generate nonrandom values. If the LockConfig Register is locked (0x00), then the RNG generates random numbers and the Nonce command functions normally.

The Nonce remains valid until one of the following events occurs:

- A MAC compare operation fails.
- MacCount reaches the maximum count (see Appendix I.1, MacCount).
- The cryptographic state machine is reset due to either receipt of a Reset command, power cycling (POR), or activation of the initialization sequence due to Wake-up from the Sleep power state (see Appendix G.2.2, Wake-Up from Sleep).
- Execution of the Nonce command resets MacCount to zero (see Appendix I.1, MacCount).

If a cryptographic operation involves two ATAES132 devices and a synchronized Nonce is required, then the Nonce synchronization procedure in Section 7.20.1, Nonce Synchronization must be used. The Nonce command cannot be used to generate a synchronized random Nonce.

---

**WARNING**

There is one Random Number Generator (RNG) Seed Register in the EEPROM memory, which is used by the KeyCreate, Nonce, and Random Commands. The RNG Seed Register is subject to the same Write endurance limitations as the other bytes in the EEPROM (see Section 9.2, Reliability for the EEPROM specifications). The application developer must not exceed the Write endurance limit.
Table 7-43. Input Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>Nonce</td>
<td>1</td>
</tr>
<tr>
<td>Mode</td>
<td>Mode</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Param1</td>
<td>Zero</td>
<td>2</td>
</tr>
<tr>
<td>Param2</td>
<td>Zero</td>
<td>2</td>
</tr>
<tr>
<td>Data</td>
<td>InSeed</td>
<td>12</td>
</tr>
</tbody>
</table>

Note: 1. The RNG Seed Register in the EEPROM will be updated automatically if Mode bit 1 = 0b, unless the Seed Register was previously updated after the most recent Power-On Reset, Wake from the Sleep state, Reset command, or Tamper Event. Updating the RNG Seed Register increases the randomness of the Nonce; however, the EEPROM Write endurance specification must be respected.

Table 7-44. Output Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReturnCode</td>
<td>1</td>
<td>Upon success, 0x00 will be returned. Any command execution failure or validation failure generates a nonzero error code, per Section 6.3, ReturnCode.</td>
</tr>
<tr>
<td>Random</td>
<td>0 or 16</td>
<td>In Random Nonce mode, the random number used to generate the Nonce is returned. In Inbound Nonce mode, no data is returned.</td>
</tr>
</tbody>
</table>

The command and response packet is transmitted as a block, beginning with the Count and ending with a packet Checksum. This block format is described in Section 6.1, Command Block and Packet.
7.20 NonceCommand

The NonceCommand generates the Nonce in a manner that allows two ATAES132 devices to have identical random Nonces based on random numbers generated by both devices. The identical Nonce values and identical MacCount values are required to encrypt data on one device for decryption by the other device.

The Random Command must be executed with Mode bit 2 = 1b prior to execution of the NonceCommand. The Random Command generates a random number, which the NonceCommand command combines with the RandomSeed provided by the second ATAES132 to generate the random Nonce.

The Nonce remains valid until one of the following events occurs:

- A MAC compare operation fails.
- MacCount reaches the maximum count (see Appendix I.1, MacCount).
- Cryptographic state machine is reset due to:
  - Receipt of a Reset command,
  - Power Cycling (POR), or
  - Activation of the initialization sequence due to Wake-Up from the Sleep power state (see Appendix G.2.2, Wake-Up from Sleep).

This command resets MacCount to zero only if the operation succeeds (see Appendix I.1). If an error occurs, the contents of the Nonce Register and the MacCount Register remained unchanged. The NonceValid flag also remains unchanged.

7.20.1 Nonce Synchronization

The following procedure synchronizes the Nonce and the MacCount Register on two ATAES132 devices. In this procedure, the device where the procedure begins is referred to as “A”, and the device it is synchronized with is referred to as “B”.

1. The Random Command is executed on Device A with Mode bit 2 set to 1b. The first 12 bytes of the random field value in the response are stored for use in step 2.
2. The Nonce command is executed on Device B with Mode bit 1 set to 1b. The 12-byte random number generated in step 1 is used as the Nonce command InSeed field value. The 12-byte random field value in the response is stored for use in step 3.
3. The NonceCompute command is executed on Device A using the 12-byte random number generated in step 2 as the RandomSeed field value.
4. Successful execution of this procedure sets the Nonce status flags on both devices to:
   - NonceValid = YesNonce
   - NonceRandom = Random
   - NonceCompute = No
   - MacCount is zero on both devices.
### Table 7-45. Input Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>NonceCompute</td>
<td>1</td>
</tr>
<tr>
<td>Mode</td>
<td>Mode</td>
<td>1</td>
</tr>
<tr>
<td>Param1</td>
<td>Zero</td>
<td>2</td>
</tr>
<tr>
<td>Param2</td>
<td>Zero</td>
<td>2</td>
</tr>
<tr>
<td>Data</td>
<td>RandomSeed</td>
<td>12</td>
</tr>
</tbody>
</table>

### Table 7-46. Output Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReturnCode</td>
<td>1</td>
<td>Upon success, 0x00 will be returned. Any command execution failure or validation failure generates a nonzero error code, per Section 6.3, ReturnCode.</td>
</tr>
</tbody>
</table>

The command and response packet is transmitted as a block, beginning with the Count and ending with a packet Checksum. This block format is described in Section 6.1, Command Block and Packet.
7.21 Random Command

The Random command generates a random number using the internal high-quality random number generator and the random number generation procedure recommended by NIST in SP800-90 (see Appendix A, Standards and Reference Documents). The Random Command returns the generated random number to the Host.

There are two Random command options:

- **Random Number Generation:**
  
  If Mode bit 2 = 0b, the 16-byte random number is returned only to the Host; it is not stored internally. This option does not affect the cryptographic state of the device.

- **Nonce Synchronization:**
  
  If Mode bit 2 = 1b, then the first 12 bytes of the random number are stored in the Nonce Register for later use by the NonceCompute command. The 16-byte random number is returned to the Host. The Nonce status flags are changed to:
  
  - NonceValid = YesNonce
  - NonceRandom = Fixed
  - NonceCompute = Yes (See Section 7.20, NonceCompute Command for the NonceCompute command and the Nonce synchronization procedure.)

If the LockConfig Register is unlocked (0x55), then the random number generator is latched in the test mode, and the Random Command will always return 16 bytes of 0xA5. If the LockConfig register is locked (0x00), then the RNG generates random numbers.

There is one Random Number Generator (RNG) Seed Register in the EEPROM memory, which is used by the KeyCreate, Nonce, and Random Commands. The RNG Seed Register is subject to the same Write endurance limitations as the other bytes in the EEPROM (see Section 9.2, Reliability for the EEPROM specifications). The application developer must not exceed the write endurance limit.
### Table 7-47. Input Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode Random</td>
<td>1</td>
<td>0x02</td>
</tr>
<tr>
<td>Mode Mode</td>
<td>1</td>
<td>Bit 0: Reserved. Must be 0b. Bit 1: 0b = Update the EEPROM RNG Seed Register prior to random number generation(1) 1b = Generate random number using the existing RNG Seed. Bit 2: 0b = Then return the random number. Do not change the Nonce. 1b = Then store the first 12 bytes of the random number in the Nonce Register, and return the 16-byte random number. Bits 3 to 7: Reserved. Must be 0b.</td>
</tr>
<tr>
<td>Param1 Zero</td>
<td>2</td>
<td>Always 0x0000.</td>
</tr>
<tr>
<td>Param2 Zero</td>
<td>2</td>
<td>Always 0x0000.</td>
</tr>
<tr>
<td>Data -</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

Note: 1. The RNG Seed Register in the EEPROM will be updated automatically if Mode bit 1 = 0b, unless the Seed Register was previously updated after the most recent Power On Reset, Wake from the Sleep state, Reset command, or Tamper Event. Updating the RNG Seed Register increases the randomness of the Random Command output; however, the EEPROM Write endurance specification must be respected.

### Table 7-48. Output Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReturnCode</td>
<td>1</td>
<td>Upon success, 0x00 will be returned. Any command execution failure or validation failure generates a nonzero error code, per Section 6.3, ReturnCode.</td>
</tr>
<tr>
<td>Random</td>
<td>16</td>
<td>The random number.</td>
</tr>
</tbody>
</table>

The command and response packet is transmitted as a block, beginning with the Count and ending with a packet Checksum. This block format is described in Section 6.1, Command Block and Packet.
7.22 Reset Command

The Reset command forces ATAES132 to reset the logic, including the AES engine, Nonce, and Authentication status flag. This command does not return a response.

When a Reset command is received, ATAES132 performs the same power-up reset sequence that occurs during Wake from the Sleep state. The reset is complete after the WakeUp Ready time, \( t_{\text{WupSL.RDY}} \) (see Section 9.4.1, Power-Up, Sleep, Standby, and Wake-Up Timing).

7.22.1 SPI Reset

During the reset of an ATAES132 configured for SPI interface mode, the device will answer the SPI Read Status Register command with 0xFF to indicate it is busy. When reset is complete, the WIP Status bit changes to 0b to indicate the device is in the Active state. The ATAES132 will only accept the SPI Read Status Register command while it is resetting; all other commands will be ignored. The SPI Read Status Register command is described in Appendix K.3.5, Read Status Register Command (RDSR).

7.22.2 I2C Reset

During the reset of an ATAES132 configured for I2C interface mode, the Host is required to perform ACK polling using the matching I2C Device Address. The ATAES132 will answer the ACK poll with an I2C NAK to indicate the device is busy during reset. The ACK poll reply will change to ACK when the device is in the Active state. ATAES132 will not accept any I2C commands while it is busy. ACK polling is described in Appendix J.3.7, Acknowledge Polling.

### Table 7-49. Input Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>Reset</td>
<td>1</td>
</tr>
<tr>
<td>Mode</td>
<td>Mode</td>
<td>1</td>
</tr>
<tr>
<td>Param1</td>
<td>Zero</td>
<td>2</td>
</tr>
<tr>
<td>Param2</td>
<td>Zero</td>
<td>2</td>
</tr>
<tr>
<td>Data</td>
<td>-</td>
<td>0</td>
</tr>
</tbody>
</table>

### Table 7-50. Output Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>No response packet is returned by the Reset command.</td>
</tr>
</tbody>
</table>

The command packet is transmitted as a block, beginning with the Count and ending with a packet Checksum. This block format is described in Section 6.1, Command Block and Packet.
7.23 Sleep Command

The Sleep command forces the ATAES132 into one of two Low-Power states; Sleep or Standby. This command does not return a response.

The Sleep state can be used to extend battery life in portable systems by powering down the ATAES132 internal circuitry when the device is sleeping. The Standby state puts the internal circuitry in a low-power state to reduce power consumption while preserving the volatile memory contents and the security state.

A device in the Sleep state will not retain any volatile memory contents or security states. A device in the Sleep state goes through a full power-up sequence upon Wake-Up.

A device in the Standby state will retain all volatile memory contents. A device in the Standby state does not go through a power-up sequence upon Wake-Up.

The ATAES132 exits the Sleep or Standby state if a Wake-Up event occurs on the I/O pins. Wakeup is discussed in Appendix L.2, Power State Transitions.

See Appendix L, Power Management for a detailed description of the ATAES132 sleep, standby, wake-up, and power management functions.

### Table 7-51. Input Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>Sleep</td>
<td>1</td>
</tr>
<tr>
<td>Mode</td>
<td>Mode</td>
<td>1</td>
</tr>
<tr>
<td>Param1</td>
<td>Zero</td>
<td>2</td>
</tr>
<tr>
<td>Param2</td>
<td>Zero</td>
<td>2</td>
</tr>
<tr>
<td>Data</td>
<td>-</td>
<td>0</td>
</tr>
</tbody>
</table>

### Table 7-52. Output Parameters

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>No response packet is returned by the Reset command.</td>
</tr>
</tbody>
</table>

The command packet is transmitted as a block, beginning with the Count and ending with a packet Checksum. This block format is described in Section 6.1, Command Block and Packet.
7.24 WriteCompute Command

The WriteCompute command encrypts data and computes the MAC required to execute the EncWrite command on a second ATAES132 device.

To use this command, the Nonce must be identical on both devices (see Section 7.20.1, Nonce Synchronization) and MacCount must have the same value on each device. Both devices must also contain identical key values, but it is not necessary for the KeyID on the origin device to match the KeyID on the Destination device. In this section, the device that encrypts data and generates the MAC is referred to as the Origin device. The device that checks the MAC is referred to as the Destination device.

If Mode bit 5, 6, or 7 is 1b, then the associated Usage Counter, SerialNum Register value, or the first four bytes of the SmallZone Register must be identical on both devices. The ManufacturingID Register must be identical on both devices, since it is always included in the MAC calculation.

A valid Nonce is required to run the WriteCompute command. If the KeyConfig[EKeyID].RandomNonce bit is 1b, then the Nonce must be random.

The value of Param2 in the FirstBlock field must match the Count field value.

**Table 7-53. Input Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Opcode</td>
<td>WriteCompute</td>
<td>1 0x16</td>
</tr>
<tr>
<td>Mode</td>
<td>Mode</td>
<td>Always 0x0000.</td>
</tr>
<tr>
<td>Param1</td>
<td>EKeyID</td>
<td>2 Upper byte is always 0x00. Lower byte is the pointer to the encrypt key. Legal values: 0x00 to 0x0F, 0xFF.</td>
</tr>
<tr>
<td>Param2</td>
<td>Count</td>
<td>2 Upper byte is always 0x00. Lower byte is the number of Data bytes to be encrypted.</td>
</tr>
<tr>
<td>Data1</td>
<td>FirstBlock</td>
<td>6 The value of this field must match the first authenticate-only block to be used when executing the EncWrite command on the Destination device.</td>
</tr>
<tr>
<td>Data2</td>
<td>SecondBlock</td>
<td>16 The value of this field must match the second authenticate-only block to be used when executing the EncWrite command on the Destination device. If Mode bits 5, 6, and 7 are 0b, then this field must be present, but is ignored.</td>
</tr>
<tr>
<td>Data3</td>
<td>InData</td>
<td>1 to 32 Input data to be encrypted (plaintext).</td>
</tr>
</tbody>
</table>

**Table 7-54. Output Parameters**

<table>
<thead>
<tr>
<th>Name</th>
<th>Size (bytes)</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReturnCode</td>
<td>1</td>
<td>Upon success, 0x00 will be returned. Any command execution or validation failure generates a nonzero error code, per Section 6.3, ReturnCode.</td>
</tr>
<tr>
<td>OutMac</td>
<td>16</td>
<td>The input MAC for the EncWrite command on the destination device.</td>
</tr>
<tr>
<td>OutData</td>
<td>16 or 32</td>
<td>The encrypted data (ciphertext) to be written to the destination device using the EncWrite command.</td>
</tr>
</tbody>
</table>

The command and response packet is transmitted as a block, beginning with the Count and ending with a packet Checksum. This block format is described in Section 6.1, Command Block and Packet.
8. Pin Lists

8.1 Package Pin List (SOIC and UDFN)

Table 8-1. Package Pin List

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CS</td>
<td>SPI Mode = CS</td>
<td>Input</td>
</tr>
<tr>
<td></td>
<td></td>
<td>I2C Mode = Not used</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>SO</td>
<td>SPI Mode = Serial Data Out</td>
<td>Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td>I2C Mode = Not used or AuthO Out</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>NC</td>
<td>No Connect</td>
<td>NC</td>
</tr>
<tr>
<td>4</td>
<td>VSS</td>
<td>Ground</td>
<td>Ground</td>
</tr>
<tr>
<td>5</td>
<td>SI/SDA</td>
<td>SPI Mode = Serial Data In</td>
<td>Input/Output</td>
</tr>
<tr>
<td></td>
<td></td>
<td>I2C Mode = Serial Data I/O</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>SCK</td>
<td>Serial Data Clock</td>
<td>Input</td>
</tr>
<tr>
<td>7</td>
<td>NC</td>
<td>No Connect</td>
<td>NC</td>
</tr>
<tr>
<td>8</td>
<td>VCC</td>
<td>Power Supply</td>
<td>Power</td>
</tr>
</tbody>
</table>

8.2 Pin Descriptions

Table 8-2. Pin Descriptions

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CS</td>
<td><strong>SPI Chip Select Bar Input pin.</strong> In the SPI communication mode, this pin functions as the slave select input. In the I2C communication mode, this pin is not used, and should be tied to VCC or VSS.</td>
</tr>
<tr>
<td>2</td>
<td>SO</td>
<td><strong>Serial Data Out pin.</strong> In the SPI communication mode, this pin functions as the serial data output. In the I2C communication mode, this pin is not used in the default configuration. It is always in the high-impedance state. If Auth signaling is enabled, then this pin functions as the AuthO output (see Appendix J.5, I2C Auth Signaling).</td>
</tr>
<tr>
<td>3</td>
<td>NC</td>
<td><strong>No Connect pin.</strong> This package pin is not used, and can be left open by the user.</td>
</tr>
<tr>
<td>4</td>
<td>VSS</td>
<td><strong>Ground.</strong></td>
</tr>
<tr>
<td>5</td>
<td>SI/SDA</td>
<td><strong>Serial Data In pin.</strong> In SPI communication mode, this pin functions as the serial data input. In I2C communication mode, this pin functions as the serial data I/O.</td>
</tr>
<tr>
<td>6</td>
<td>SCK</td>
<td><strong>Serial Clock Input pin.</strong> In both SPI and I2C serial communication modes, this pin is used as the serial interface clock.</td>
</tr>
<tr>
<td>7</td>
<td>NC</td>
<td><strong>No Connect pin.</strong> This package pin is not used, and can be left open by the user.</td>
</tr>
<tr>
<td>8</td>
<td>VCC</td>
<td><strong>Supply Voltage.</strong> To insure a stable VCC level, it is recommended that VCC be decoupled with a a high quality capacitor, in the order of 0.01μF, positioned close to the VCC and VSS pins of the ATAES132.</td>
</tr>
</tbody>
</table>
9. Electrical Characteristics

9.1 Absolute Maximum Ratings*

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typical</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Operating Temperature</td>
<td>−40°C to +85°C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Storage Temperature</td>
<td>−65°C to +150°C</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Maximum Operating Voltage</td>
<td>6.0V</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DC Output Current</td>
<td>5.0mA</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Voltage on any pin</td>
<td>-0.7V to (Vcc + 0.7V)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HBM ESD</td>
<td>3kV minimum</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notice*: Stresses beyond those listed under “Absolute Maximum Ratings” may cause permanent damage to the device. This is a stress rating only, and the 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.

9.2 Reliability

The ATAES132 is fabricated with the Atmel high reliability CMOS EEPROM manufacturing technology. The reliability ratings in Table 9-1 apply to each byte of the EEPROM memory.

Table 9-1. EEPROM Reliability

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Min</th>
<th>Typical</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>Write Endurance (each byte)</td>
<td>100,000</td>
<td></td>
<td></td>
<td>Write Cycles</td>
</tr>
<tr>
<td>Data Retention (at 55°C)</td>
<td>10</td>
<td></td>
<td></td>
<td>Years</td>
</tr>
<tr>
<td>Data Retention (at 35°C)</td>
<td>30-50</td>
<td></td>
<td></td>
<td>Years</td>
</tr>
<tr>
<td>Read Endurance</td>
<td>Unlimited</td>
<td></td>
<td></td>
<td>Read Cycles</td>
</tr>
</tbody>
</table>

Note: 1. These specifications apply to every byte of the User Memory, Configuration Memory, and Key Memory. The Write Endurance specification also applies to the Random Number Generator EEPROM Seed Register.
9.3 DC Characteristics

9.3.1 Supply Characteristics

Table 9-2. Supply Voltage and Current Characteristics

Applicable over recommended operating range from $T_A = -40^\circ C$ to $+85^\circ C$, $V_{CC} = +2.5V$ to $+5.5V$ (unless otherwise noted). (1)

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Test Conditions</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$V_{CC}$ (1)</td>
<td>Supply Voltage</td>
<td>$V_{CC} = 3.3V$ at $f_{max}$ (4) $SO = $ Open (3), Read, Write, or AES operation.</td>
<td>2.50</td>
<td>5.50</td>
<td></td>
<td>V</td>
</tr>
<tr>
<td>$I_{CC1}$</td>
<td>Supply Current</td>
<td>$V_{CC} = 3.3V$ at $f_{max}$ (4) $SO = $ Open (3), Read, Write, or AES operation.</td>
<td>6</td>
<td></td>
<td>10</td>
<td>mA</td>
</tr>
<tr>
<td>$I_{CC2}$</td>
<td>Supply Current</td>
<td>$V_{CC} = 5.5V$ at $f_{max}$ (4) $SO = $ Open (3), Read, Write, or AES operation.</td>
<td>10</td>
<td></td>
<td></td>
<td>mA</td>
</tr>
<tr>
<td>$I_{CC3}$</td>
<td>Idle Current</td>
<td>$V_{CC} = 3.3V$ or $5.5V$ at $f_{max}$ (4) $SO = $ Open (3), Waiting for a command.</td>
<td>600</td>
<td>800</td>
<td></td>
<td>µA</td>
</tr>
<tr>
<td>$I_{SL1}$</td>
<td>Sleep Current</td>
<td>$V_{CC} = 3.3V$ $\overline{CS} = V_{CC}$ (3), Sleep State (5)</td>
<td>0.10</td>
<td>0.25</td>
<td></td>
<td>µA</td>
</tr>
<tr>
<td>$I_{SL2}$</td>
<td>Sleep Current</td>
<td>$V_{CC} = 5.5V$ $\overline{CS} = V_{CC}$ (3), Sleep State (5)</td>
<td>0.25</td>
<td>0.50</td>
<td></td>
<td>µA</td>
</tr>
<tr>
<td>$I_{SB1}$</td>
<td>Standby Current</td>
<td>$V_{CC} = 3.3V$ $\overline{CS} = V_{CC}$ (3), Standby State (5)</td>
<td>15</td>
<td>30</td>
<td></td>
<td>µA</td>
</tr>
<tr>
<td>$I_{SB2}$</td>
<td>Standby Current</td>
<td>$V_{CC} = 5.5V$ $\overline{CS} = V_{CC}$ (3), Standby State (5)</td>
<td>20</td>
<td>40</td>
<td></td>
<td>µA</td>
</tr>
</tbody>
</table>

Notes: 1. Typical values are at $25^\circ C$, and are for reference only. Typical values are not tested or guaranteed.
2. On power-up, $V_{CC}$ must rise continuously from $V_{SS}$ to the operating voltage, with a rise time no faster than $1V/\mu s$.
3. All input pins must be held at either $V_{SS}$ or $V_{CC}$ during this measurement. In SPI interface mode, the $\overline{CS}$ pin must be at $V_{CC}$. In I2C interface mode, the $\overline{CS}$ pin may be in either state.
4. Measurement is performed at the maximum serial clock frequency. In the I2C interface mode, $f_{max}$ is $1MHz$. In the SPI interface mode, $f_{max}$ is $10MHz$.
5. See Appendix L, Power Management for Sleep and Standby state information. The Sleep command is described in Section 7.23, Sleep Command.
6. The ATAES132 does not support hot swapping or hot plugging. Connecting or disconnecting this device to a system while power is energized can cause permanent damage to the ATAES132.
9.3.2 I/O Characteristics

Table 9-3. DC Characteristics

Applicable over recommended operating range from $T_A = -40^\circ C$ to $+85^\circ C$, $V_{CC} = +2.5V$ to $+5.5V$ (unless otherwise noted).

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Test conditions</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$I_{IL}$</td>
<td>Input Current</td>
<td>$V_{IN} = 0V$ or $V_{CC}$</td>
<td>$-3.0$</td>
<td>$3.0$</td>
<td>$\mu A$</td>
</tr>
<tr>
<td>$I_{LO}$</td>
<td>Output Leakage</td>
<td>$V_{OUT} = 0V$ or $V_{CC}$</td>
<td>$-3.0$</td>
<td>$3.0$</td>
<td>$\mu A$</td>
</tr>
<tr>
<td>$V_{IL}^{(1)}$</td>
<td>Input Low-Voltage</td>
<td></td>
<td>$-0.5$</td>
<td>$V_{CC} \times 0.3$</td>
<td>$V$</td>
</tr>
<tr>
<td>$V_{IH}^{(1)}$</td>
<td>Input High-Voltage</td>
<td>$V_{OUT} = 0V$ or $V_{CC}$</td>
<td>$V_{CC} \times 0.7$</td>
<td>$V_{CC} + 0.5$</td>
<td>$V$</td>
</tr>
<tr>
<td>$V_{OL1}^{(2)}$</td>
<td>Output Low-Voltage, Except SI/SDA in I$^2$C Mode</td>
<td>$I_{OL} = 3.0mA$</td>
<td>$0$</td>
<td>$0.4$</td>
<td>$V$</td>
</tr>
<tr>
<td>$V_{OH1}^{(2)}$</td>
<td>Output High-voltage, Except SI/SDA in I$^2$C Mode</td>
<td>$I_{OH} = -3.0mA$</td>
<td>$V_{CC} - 0.8$</td>
<td>$V_{CC}$</td>
<td>$V$</td>
</tr>
<tr>
<td>$V_{OL2}$</td>
<td>Output Low-voltage, SI/SDA Pin in the I$^2$C Mode Only</td>
<td>$I_{OL} = 3.0mA$</td>
<td>$0$</td>
<td>$0.4$</td>
<td>$V$</td>
</tr>
</tbody>
</table>

Notes: 1. $V_{IL}$ min and $V_{IH}$ max are for reference only, and are not tested.  
2. In the I$^2$C interface mode, if Auth signaling is enabled, the SO pin functions as the AuthO output (see Appendix J.5, I$^2$C Interface). When AuthO is high, the $V_{OH1}$ specification applies. When AuthO is not high, the pin is in the high-impedance state; the $V_{OL1}$ specification is not applicable.

9.4 AC Characteristics

Table 9-4. AC Characteristics

Applicable over recommended operating range from $T_A = -40^\circ C$ to $+85^\circ C$, $V_{CC} = +2.5V$ to $+5.5V$.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{WC1}$</td>
<td>User Zone Write Cycle Time$^{(1)}$</td>
<td>$6.0$</td>
<td>$9.0$</td>
<td>$ms$</td>
</tr>
<tr>
<td>$t_{WC2}$</td>
<td>Key Zone Write Cycle Time$^{(1)}$</td>
<td>$12.0$</td>
<td>$16.0$</td>
<td>$ms$</td>
</tr>
<tr>
<td>Command Response Time</td>
<td></td>
<td></td>
<td>See Appendix N.</td>
<td></td>
</tr>
</tbody>
</table>

Note: 1. The write cycle time includes the EEPROM Erase, Write, and Automatic Data Write verification operations.
9.4.1 Power-Up, Sleep, Standby, and Wake-Up Timing

Table 9-5. Power-Up, Sleep, and Wake-Up Timing Characteristics\(^{(1)(2)}\)

Applicable over recommended operating range from $T_A = -40^\circ C$ to $+85^\circ C$, $V_{CC} = +2.5V$ to $+5.5V$.

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Typ</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t_{PU:STATUS}$</td>
<td>Power-Up Time, Status</td>
<td>500</td>
<td>600</td>
<td>µs</td>
<td></td>
</tr>
<tr>
<td>$t_{PU:RDY}$</td>
<td>Power-Up Ready Time</td>
<td>1200</td>
<td>1500</td>
<td>µs</td>
<td></td>
</tr>
<tr>
<td>$t_{SB}$</td>
<td>Sleep Time, Entering the Standby State</td>
<td>65</td>
<td>100</td>
<td>µs</td>
<td></td>
</tr>
<tr>
<td>$t_{SL}$</td>
<td>Sleep Time, Entering the Sleep State</td>
<td>55</td>
<td>90</td>
<td>µs</td>
<td></td>
</tr>
<tr>
<td>$t_{WupSB:STATUS}$</td>
<td>Wake-Up Status Time, Standby State</td>
<td>50</td>
<td>100</td>
<td>µs</td>
<td></td>
</tr>
<tr>
<td>$t_{WupSB:RDY}$</td>
<td>Wake-Up Ready Time, Standby State</td>
<td>200</td>
<td>240</td>
<td>µs</td>
<td></td>
</tr>
<tr>
<td>$t_{WupSL:STATUS}$</td>
<td>Wake-Up Status, Sleep State</td>
<td>500</td>
<td>600</td>
<td>µs</td>
<td></td>
</tr>
<tr>
<td>$t_{WupSL:RDY}$</td>
<td>Wake-Up Ready Time, Sleep State</td>
<td>1200</td>
<td>1500</td>
<td>µs</td>
<td></td>
</tr>
</tbody>
</table>

Notes: 1. All values are based on characterization and are not tested. Typical values are at 25°C and are for reference only.
2. See Appendix L, Power Management for Power-Up, Sleep, Standby, and Wake-Up specifications. The Sleep command is described in Section 7.23, Sleep Command.
9.4.2 $i^2$C Interface Timing

Table 9-6. AC Characteristics of $i^2$C Interface

Applicable over recommended operating range from $T_A = -40^\circ C$ to $+85^\circ C$, $V_{CC} = +2.5V$ to $+5.5V$, $CL = 1$ TTL Gate and 100pF (unless otherwise noted).

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_{SCK}$</td>
<td>SCK Clock Frequency</td>
<td></td>
<td>1</td>
<td>MHz</td>
</tr>
<tr>
<td></td>
<td>SCK Clock Duty Cycle</td>
<td>30</td>
<td>70</td>
<td>percent</td>
</tr>
<tr>
<td>$t_{HIGH}$</td>
<td>SCK High Time</td>
<td>400</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>$t_{LOW}$</td>
<td>SCK Low Time</td>
<td>400</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>$t_{SU.STA}$</td>
<td>Start Setup Time</td>
<td>250</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>$t_{HD.STA}$</td>
<td>Start Hold Time</td>
<td>250</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>$t_{SU.STO}$</td>
<td>Stop Setup Time</td>
<td>250</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>$t_{SL.DAT}$</td>
<td>Data in Setup Time</td>
<td>100</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>$t_{HD.DAT}$</td>
<td>Data in Hold Time</td>
<td>0</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>$t_R$</td>
<td>Input Rise Time$^{(1)}$</td>
<td></td>
<td>300</td>
<td>ns</td>
</tr>
<tr>
<td>$t_F$</td>
<td>Input Fall Time$^{(1)}$</td>
<td></td>
<td>100</td>
<td>ns</td>
</tr>
<tr>
<td>$t_{AA}$</td>
<td>Clock Low to Data Out Valid</td>
<td>50</td>
<td>550</td>
<td>ns</td>
</tr>
<tr>
<td>$t_{DH}$</td>
<td>Data Out Hold Time</td>
<td>50</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>$t_{BUF}$</td>
<td>Time bus must be free before a new transmission can start.$^{(1)}$</td>
<td>500</td>
<td></td>
<td>ns</td>
</tr>
</tbody>
</table>

Notes: 1. Values are based on characterization, and are not tested.
2. AC measurement conditions:
   - RL (connects between SDA and $V_{CC}$): 2.0k$\Omega$ (for $V_{CC} +2.5V$ 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: $0.5V_{CC}$

Figure 1-1. $i^2$C Synchronous Data Timing
9.4.3 SPI Interface Timing

Table 9-7. AC Characteristics of SPI Interface

Applicable over recommended operating range from $T_A = -40^\circ C$ to $+85^\circ C$, $V_{CC} = +2.5V$ to $+5.5V$, $CL = 1$ TTL Gate and 30pF (unless otherwise noted).

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Parameter</th>
<th>Min</th>
<th>Max</th>
<th>Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>$f_{SCK}$</td>
<td>SCK Clock Frequency</td>
<td>0</td>
<td>10</td>
<td>MHz</td>
</tr>
<tr>
<td></td>
<td>SCK Clock Duty Cycle</td>
<td>30</td>
<td>70</td>
<td>percent</td>
</tr>
<tr>
<td>$t_{WH}$</td>
<td>SCK High Time</td>
<td>40</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>$t_{WL}$</td>
<td>SCK Low Time</td>
<td>40</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>$t_{CS}$</td>
<td>CS High Time</td>
<td>50</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>$t_{CSS}$</td>
<td>CS Setup Time</td>
<td>50</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>$t_{CSH}$</td>
<td>CS Hold Time</td>
<td>50</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>$t_{SU}$</td>
<td>Data In Setup Time</td>
<td>10</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>$t_{H}$</td>
<td>Data In Hold Time</td>
<td>10</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>$t_{RI}$</td>
<td>Input Rise Time(^{(1)})</td>
<td>2</td>
<td></td>
<td>$\mu s$</td>
</tr>
<tr>
<td>$t_{FI}$</td>
<td>Input Fall Time(^{(1)})</td>
<td>2</td>
<td></td>
<td>$\mu s$</td>
</tr>
<tr>
<td>$t_{V}$</td>
<td>Output Valid</td>
<td>0</td>
<td>40</td>
<td>ns</td>
</tr>
<tr>
<td>$t_{HO}$</td>
<td>Output Hold Time</td>
<td>0</td>
<td></td>
<td>ns</td>
</tr>
<tr>
<td>$t_{DIS}$</td>
<td>Output Disable Time</td>
<td>50</td>
<td></td>
<td>ns</td>
</tr>
</tbody>
</table>

Note: 1. Values are based on characterization, and are not tested.

Figure 1-2. SPI Synchronous Data Timing
Appendix A. Standards and Reference Documents

A.1 National and International Standards
The ATAES132 is designed to comply with the requirements of the AES Standard.

Available at: http://csrc.nist.gov/groups/ST/toolkit/block_ciphers.html.

A.2 References


Available at: http://csrc.nist.gov/groups/ST/toolkit/random_number.html.

Available at http://www.jedec.org.

Available at: http://www.iso.org or http://www.ansi.org or from National Standards Body.

Available at: http://www.iso.org or http://www.ansi.org or from National Standards Body.
## Appendix B. Memory Map

### B.1 Memory Map

Reserved memory cannot be written or read.

<table>
<thead>
<tr>
<th>Byte Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000h-0FFFh</td>
<td>User Memory</td>
</tr>
<tr>
<td>1000h-EFFFh</td>
<td>Reserved</td>
</tr>
<tr>
<td>F000h-F05Fh</td>
<td>Configuration Memory – Device Config</td>
</tr>
<tr>
<td>F060h-F07Fh</td>
<td>Configuration Memory – CounterConfig</td>
</tr>
<tr>
<td>F080h-F0BFh</td>
<td>Configuration Memory – KeyConfig</td>
</tr>
<tr>
<td>F0C0h-F0FFh</td>
<td>Configuration Memory – ZoneConfig</td>
</tr>
<tr>
<td>F100h-F17Fh</td>
<td>Configuration Memory - Counters</td>
</tr>
<tr>
<td>F180h-F1DFh</td>
<td>Configuration Memory – FreeSpace</td>
</tr>
<tr>
<td>F1E0h-F1FFh</td>
<td>Configuration Memory – SmallZone</td>
</tr>
<tr>
<td>F200h-F2FFh</td>
<td>Key Memory</td>
</tr>
<tr>
<td>F300h-FDFFh</td>
<td>Reserved</td>
</tr>
<tr>
<td>FE00h</td>
<td>Command / Response Memory Buffer</td>
</tr>
<tr>
<td>FE01h-FFDFh</td>
<td>Reserved</td>
</tr>
<tr>
<td>FFE0h</td>
<td>I/O Address Reset</td>
</tr>
<tr>
<td>FFE1h-FFEFh</td>
<td>Reserved</td>
</tr>
<tr>
<td>FFF0h</td>
<td>STATUS Register</td>
</tr>
<tr>
<td>FFF1h-FFFFFh</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
B.2 EEPROM Page Boundary

The ATAES132 EEPROM has 32-byte physical pages. An EEPROM Write can never cross the boundary between two physical pages. BlockRead and EncRead operations cannot cross the boundary between two physical pages. Table B-2 illustrates the page boundary locations for the ATAES132.

Table B-2. ATAES132 EEPROM Page Boundary Locations

<table>
<thead>
<tr>
<th>Address</th>
<th>0h</th>
<th>1h</th>
<th>2h</th>
<th>3h</th>
<th>4h</th>
<th>5h</th>
<th>6h</th>
<th>7h</th>
<th>8h</th>
<th>9h</th>
<th>Ah</th>
<th>Bh</th>
<th>Ch</th>
<th>Dh</th>
<th>Eh</th>
<th>Fh</th>
</tr>
</thead>
<tbody>
<tr>
<td>XX00h-XX0Fh</td>
<td>32-byte EEPROM Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XX10h-XX1Fh</td>
<td>32-byte EEPROM Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XX20h-XX2Fh</td>
<td>32-byte EEPROM Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XX30h-XX3Fh</td>
<td>32-byte EEPROM Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XX40h-XX4Fh</td>
<td>32-byte EEPROM Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XX50h-XX5Fh</td>
<td>32-byte EEPROM Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XX60h-XX6Fh</td>
<td>32-byte EEPROM Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XX70h-XX7Fh</td>
<td>32-byte EEPROM Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XX80h-XX8Fh</td>
<td>32-byte EEPROM Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XX90h-XX9Fh</td>
<td>32-byte EEPROM Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XXA0h-XXAFh</td>
<td>32-byte EEPROM Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XXB0h-XXBFh</td>
<td>32-byte EEPROM Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XXC0h-XXCFh</td>
<td>32-byte EEPROM Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XXD0h-XXDFh</td>
<td>32-byte EEPROM Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XXE0h-XXEFh</td>
<td>32-byte EEPROM Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>XXF0h-XXFFh</td>
<td>32-byte EEPROM Page</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Appendix C. User Memory Map

The 32Kb User Memory consists of 16 user zones, each containing 2Kb (256 bytes) of memory. The physical page size is 32 bytes; Write operations cannot cross page boundaries.

Every Memory Zone has an independent set of access restrictions, and all bytes within a zone have the same access restrictions. The Configuration Memory (Appendix E, Configuration Memory Map) contains an access register for each Memory Zone that defines the access requirements for the User Zone.

Table C-1. User Memory Map

<table>
<thead>
<tr>
<th>Byte Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000h-00FFh</td>
<td>User Zone 0</td>
</tr>
<tr>
<td>0100h-01FFh</td>
<td>User Zone 1</td>
</tr>
<tr>
<td>0200h-02FFh</td>
<td>User Zone 2</td>
</tr>
<tr>
<td>0300h-03FFh</td>
<td>User Zone 3</td>
</tr>
<tr>
<td>0400h-04FFh</td>
<td>User Zone 4</td>
</tr>
<tr>
<td>0500h-05FFh</td>
<td>User Zone 5</td>
</tr>
<tr>
<td>0600h-06FFh</td>
<td>User Zone 6</td>
</tr>
<tr>
<td>0700h-07FFh</td>
<td>User Zone 7</td>
</tr>
<tr>
<td>0800h-08FFh</td>
<td>User Zone 8</td>
</tr>
<tr>
<td>0900h-09FFh</td>
<td>User Zone 9</td>
</tr>
<tr>
<td>0A00h-A0FFh</td>
<td>User Zone A</td>
</tr>
<tr>
<td>0B00h-0BFFh</td>
<td>User Zone B</td>
</tr>
<tr>
<td>0C00h-0CFFh</td>
<td>User Zone C</td>
</tr>
<tr>
<td>0D00h-0DFFh</td>
<td>User Zone D</td>
</tr>
<tr>
<td>0E00h-0EFFh</td>
<td>User Zone E</td>
</tr>
<tr>
<td>0F00h-0FFFh</td>
<td>User Zone F</td>
</tr>
</tbody>
</table>
Appendix D. Command Memory Map

The ATAES132 commands are executed by writing the command packet to the virtual memory using standard SPI or I2C Write commands. The response packet is retrieved by reading it from the virtual memory using standard SPI or I2C Read commands. The Command/Response Memory Buffer is 64 bytes.

The ATAES132 commands are executed by writing the command packet to virtual memory at starting address 0xFE00 using standard Write commands (see Appendix J, I2C Interface and Appendix K, SPI Interface). The response packet is retrieved by reading from the virtual memory at starting address 0xFE00 using standard Read commands. The Device Status Register (STATUS) is located at 0xFFF0 (see Appendix G, Understanding the STATUS Register).

To reset the address pointer in the Command/Response Memory Buffer to the base address of the buffer, the Host writes one or more bytes to the IO Address Reset Register at address 0xFFE0 using the standard Write command. Any value can be written to the IO Address Reset Register to reset the buffer address pointer.

Table D-1. Command/Response Virtual Memory Map

<table>
<thead>
<tr>
<th>Byte Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>FE00h</td>
<td>Command/Response Memory Buffer</td>
</tr>
<tr>
<td>FE01h-FFDFh</td>
<td>Reserved</td>
</tr>
<tr>
<td>FFE0h</td>
<td>I/O Address Reset</td>
</tr>
<tr>
<td>FFE1h-FFEFh</td>
<td>Reserved</td>
</tr>
<tr>
<td>FFF0h</td>
<td>STATUS Register</td>
</tr>
<tr>
<td>FFF1h-FFFFh</td>
<td>Reserved</td>
</tr>
</tbody>
</table>

D.1 Command Memory Buffer

The Command Memory Buffer is a write-only buffer memory that is used by writing a command block to the buffer at the base address of 0xFE00. After the Host completes its Write operation to the buffer, the ATAES132 verifies the integrity of the block by checking the 16-bit Checksum, and then executes the requested operation. See Section 6.1, Command Block and Packet for a description of the crypto command block.

Write operations that begin at any other location within the buffer are invalid and will not be processed by the ATAES132.

Table D-2. Command Memory Buffer Map

<table>
<thead>
<tr>
<th>Base Address</th>
<th>Base + 1</th>
<th>Base + 2</th>
<th>Base + 3</th>
<th>......</th>
<th>......</th>
<th>......</th>
<th>......</th>
<th>Base + N-2</th>
<th>Base + N-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>Opcode</td>
<td>Mode</td>
<td>Param1</td>
<td>Param1</td>
<td>Param2</td>
<td>......</td>
<td>......</td>
<td>DataX</td>
<td>CRC1</td>
</tr>
</tbody>
</table>
D.1.1 Using the Command Memory Buffer

The Host should write a single byte to the IO Address Reset Register before writing a new command block to the Command Memory Buffer. This resets the buffer address pointer to the base address. The Host then writes the ATAES132 command block to the buffer using one or more standard SPI or I\(^2\)C Write commands. After the entire command block is written by the Host microcontroller, the ATAES132 checks the 16-bit Checksum and executes the command. The Host should read the STATUS Register to determine if an error occurred or if the response is ready to be read.

If a Checksum error occurs, then the buffer address pointer must be reset by the Host before the command block is retransmitted. If no errors occur, then the response can be read from the Response Memory Buffer, as described in Appendix D.2.10, Using the Response Memory Buffer (see Appendix G, Understanding the STATUS Register for examples).

The Command Memory Buffer size is 64 bytes. If the Host writes more than 64 bytes to the buffer, it will cause a buffer overflow error. If the Host hardware must send more bytes to the ATAES132 than are required to transmit a command block (due to Host hardware limitations), then all bytes transmitted after the block Checksum must contain 0xFF.

D.2 Response Memory Buffer

The Response Memory Buffer is a read-only memory buffer that is used by reading a response from the buffer at the base address of 0xFE00. The base address of the Response Memory Buffer contains the first byte of the response packet after a Crypto command is processed. See Section 6.1, Command Block and Packet for a description of the crypto response packet.

Read operations that begin at any location above the base address are invalid and will either be NAKed (in I\(^2\)C mode) or ignored (output will tri-state in SPI mode).

Table D-3. Response Memory Buffer Map Following a Crypto Command

<table>
<thead>
<tr>
<th>Base Address</th>
<th>Base + 1</th>
<th>Base + 2</th>
<th>Base + 3</th>
<th>......</th>
<th>......</th>
<th>......</th>
<th>......</th>
<th>Base + N-2</th>
<th>Base + N-1</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>ReturnCode</td>
<td>Data1</td>
<td>Data2</td>
<td>Data3</td>
<td>......</td>
<td>......</td>
<td>......</td>
<td>DataX</td>
<td>CRC1</td>
</tr>
</tbody>
</table>

The Response Memory Buffer is also used to report errors that occur during execution of standard I\(^2\)C or SPI Write commands. When the I\(^2\)C or SPI command execution is complete (as indicated by the STATUS Register), the Response Memory Buffer contains a block containing an error code (ReturnCode) if an error occurred; otherwise, it contains a block containing ReturnCode = 0x00. Reading the Response Memory Buffer does not alter the contents of the Response Memory Buffer or the STATUS Register (see Appendix G). See Section 6.3, ReturnCode for the error descriptions.

Table D-4. Response Memory Buffer Map Following a Standard I\(^2\)C or SPI Write Operation

<table>
<thead>
<tr>
<th>Base Address</th>
<th>Base + 1</th>
<th>Base + N-2</th>
<th>......</th>
<th>......</th>
<th>......</th>
<th>......</th>
<th>......</th>
<th>......</th>
<th>......</th>
<th>......</th>
</tr>
</thead>
<tbody>
<tr>
<td>Count</td>
<td>ReturnCode</td>
<td>CRC1</td>
<td>CRC2</td>
<td>FF(_h)</td>
<td>FF(_h)</td>
<td>FF(_h)</td>
<td>FF(_h)</td>
<td>FF(_h)</td>
<td>FF(_h)</td>
<td>FF(_h)</td>
</tr>
</tbody>
</table>
D.2.1 Using the Response Memory Buffer

After an ATAES132 command is executed, the RRDY bit of the STATUS Register is set to 1b to indicate that a new response is available in the Response Memory Buffer. The Host reads the response block from the buffer using one or more standard SPI or I²C Read commands. After the entire response block is read, the Host microcontroller checks the 16-bit Checksum.

If a Checksum error occurs, then the buffer address pointer must be reset by the Host before the response block is reread. If the Host reads more bytes from the response buffer than necessary to retrieve the block, then all bytes after the block Checksum will contain 0xFF (see Appendix G for examples). The Response Memory Buffer size is 64 bytes.

D.3 IO Address Reset Register

Writing the IO Address Reset Register (address 0xFFE0) with any value causes the address pointers in the Command Memory Buffer and the Response Memory Buffer to be reset to the base address of the buffer. The IO Address Reset Register can be written with 1 to 32 bytes of data without generating an error; the data bytes will be ignored.

Writing the IO Address Reset Register does not alter the contents of the Response Memory Buffer or the value of the STATUS Register. Writing the IO Address Reset Register clears the Command Memory Buffer (see Appendix G, Understanding the STATUS Register for examples).

D.4 Device Status Register (STATUS)

The Device Status Register is used for handshaking between the Host microcontroller and the ATAES132. The Host is expected to read the STATUS Register before sending a command or reading a response. See Appendix G for the definition and behavior of the STATUS Register. If the ATAES132 is configured in SPI interface mode, the STATUS Register can also be read using the SPI RDSR command, as described in Appendix K.3.5, Read Status Register Command (RDSR).

Reading the STATUS Register does not alter the contents of the Command Memory Buffer, the contents of the Response Memory Buffer, or the value of the STATUS Register.
Appendix E. Configuration Memory Map

The ATAES132 Configuration Memory is located from address 0xF000 to address 0xF1FF. The Configuration Memory can always be read using the BlockRead command (see Section 7.4, BlockRead Command). See Appendix E.2, Configuration Register Descriptions for descriptions of each configuration register. A memory map showing the default register values appears in Appendix O, Default Configuration.

E.1 Configuration Memory Map

Table E-1. ATAES132 Configuration Memory Map

<table>
<thead>
<tr>
<th>Address</th>
<th>0h / 8h</th>
<th>1h / 9h</th>
<th>2h / Ah</th>
<th>3h / Bh</th>
<th>4h / Ch</th>
<th>5h / Dh</th>
<th>6h / Eh</th>
<th>7h / Fh</th>
</tr>
</thead>
<tbody>
<tr>
<td>F000h-F007h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F008h-F00Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F010h-F017h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F018h-F01Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F020h-F027h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F028h-F02Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F030h-F037h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F038h-F03Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F040h-F047h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F048h-F04Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F050h-F057h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F058h-F05Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F060h-F067h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F068h-F06Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F070h-F077h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F078h-F07Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F080h-F087h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F088h-F08Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F090h-F097h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F098h-F09Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F0A0h-F0A7h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F0A8h-F0AFh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F0B0h-F0B7h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F0B8h-F0BFh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F0C0h-F0C7h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F0C8h-F0CFh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F0D0h-F0D7h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F0D8h-F0DFh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F0E0h-F0E7h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F0E8h-F0EFh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F0F0h-F0F7h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

**Legend:**
- **SerialNum**: Serial Number
- **LotHistory**: Lot History
- **JEDEC**: JEDEC
- **Reserved**: Reserved
- **Algorithm**: Algorithm
- **EEPSize**: EEPROM Size
- **EncReadSize**: Encrypted Read Size
- **EncWrtSize**: Encrypted Write Size
- **DeviceNum**: Device Number
- **Reserved**: Reserved
- **LockKeys**: Lock Keys
- **LockSmall**: Lock Small
- **LockConfig**: Lock Configuration
- **Reserved**: Reserved
- **ManufacturingID**: Manufacturing ID
- **PermConfig**: Permanent Configuration
- **Reserved**: Reserved
- **I2CAddr**: I2C Address
- **ChipConfig**: Chip Configuration
- **RFU**: Reserved
- **CounterConfig 00**: Counter Configuration 00
- **CounterConfig 01**: Counter Configuration 01
- **CounterConfig 02**: Counter Configuration 02
- **CounterConfig 03**: Counter Configuration 03
- **CounterConfig 04**: Counter Configuration 04
- **CounterConfig 05**: Counter Configuration 05
- **CounterConfig 06**: Counter Configuration 06
- **CounterConfig 07**: Counter Configuration 07
- **CounterConfig 08**: Counter Configuration 08
- **CounterConfig 09**: Counter Configuration 09
- **CounterConfig 0A**: Counter Configuration 0A
- **CounterConfig 0B**: Counter Configuration 0B
- **CounterConfig 0C**: Counter Configuration 0C
- **CounterConfig 0D**: Counter Configuration 0D
- **CounterConfig 0E**: Counter Configuration 0E
- **CounterConfig 0F**: Counter Configuration 0F
- **KeyConfig 00**: Key Configuration 00
- **KeyConfig 01**: Key Configuration 01
- **KeyConfig 02**: Key Configuration 02
- **KeyConfig 03**: Key Configuration 03
- **KeyConfig 04**: Key Configuration 04
- **KeyConfig 05**: Key Configuration 05
- **KeyConfig 06**: Key Configuration 06
- **KeyConfig 07**: Key Configuration 07
- **KeyConfig 08**: Key Configuration 08
- **KeyConfig 09**: Key Configuration 09
- **KeyConfig 0A**: Key Configuration 0A
- **KeyConfig 0B**: Key Configuration 0B
- **KeyConfig 0C**: Key Configuration 0C
- **KeyConfig 0D**: Key Configuration 0D
- **ZoneConfig 00**: Zone Configuration 00
- **ZoneConfig 01**: Zone Configuration 01
- **ZoneConfig 02**: Zone Configuration 02
- **ZoneConfig 03**: Zone Configuration 03
- **ZoneConfig 04**: Zone Configuration 04
- **ZoneConfig 05**: Zone Configuration 05
- **ZoneConfig 06**: Zone Configuration 06
- **ZoneConfig 07**: Zone Configuration 07
- **ZoneConfig 08**: Zone Configuration 08
- **ZoneConfig 09**: Zone Configuration 09
- **ZoneConfig 0A**: Zone Configuration 0A
- **ZoneConfig 0B**: Zone Configuration 0B
- **ZoneConfig 0C**: Zone Configuration 0C
- **ZoneConfig 0D**: Zone Configuration 0D
<table>
<thead>
<tr>
<th>Address</th>
<th>0h / 8h</th>
<th>1h / 9h</th>
<th>2h / Ah</th>
<th>3h / Bh</th>
<th>4h / Ch</th>
<th>5h / Dh</th>
<th>6h / Eh</th>
<th>7h / Fh</th>
</tr>
</thead>
<tbody>
<tr>
<td>F0F8h-F0FFh</td>
<td>ZoneConfig 0E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F100h-F107h</td>
<td>ZoneConfig 0F</td>
<td>Counter 00</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F108h-F10Fh</td>
<td></td>
<td>Counter 01</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F110h-F117h</td>
<td></td>
<td>Counter 02</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F118h-F11Fh</td>
<td></td>
<td>Counter 03</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F120h-F127h</td>
<td></td>
<td>Counter 04</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F128h-F12Fh</td>
<td></td>
<td>Counter 05</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F130h-F137h</td>
<td></td>
<td>Counter 06</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F138h-F13Fh</td>
<td></td>
<td>Counter 07</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F140h-F147h</td>
<td></td>
<td>Counter 08</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F148h-F14Fh</td>
<td></td>
<td>Counter 09</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F150h-F157h</td>
<td></td>
<td>Counter 0A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F158h-F15Fh</td>
<td></td>
<td>Counter 0B</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F160h-F167h</td>
<td></td>
<td>Counter 0C</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F168h-F16Fh</td>
<td></td>
<td>Counter 0D</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F170h-F177h</td>
<td></td>
<td>Counter 0E</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F178h-F17Fh</td>
<td></td>
<td>Counter 0F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F180h-F187h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F188h-F18Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F190h-F197h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F198h-F19Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1A0h-F1A7h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1A8h-F1AFh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1B0h-F1B7h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1B8h-F1BFh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1C0h-F1C7h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1C8h-F1CFh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1D0h-F1D7h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1D8h-F1DFh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1E0h-F1E7h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1E8h-F1EFh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1F0h-F1F7h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F1F8h-F1FFh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes:
1. Orange registers = Locked at the factory and cannot be changed by the customer.
2. Blue registers = Lock registers can be changed only by using the Lock command (see Section 7.18, EncRead Command).
3. Green registers = Configuration registers can be written by the customer prior to locking (by setting LockConfig to 0x00 using the Lock command).
4. Yellow registers = The SmallZone Register can be written by the customer prior to locking (by setting LockSmall to 0x00 using the Lock command). SmallZone is locked separately from the remainder of the Configuration Memory.
E.2 Configuration Register Descriptions

Each register in the Configuration Memory is briefly described in this section. References are provided to detailed information in other sections of this specification. The registers are described in the same order in which they occur in the memory map in Appendix E.1, Configuration Memory Map.

E.2.1 SerialNum Register

SerialNum is an 8-byte, read-only register that is programmed by Atmel at the factory. The contents of this register are guaranteed to be unique on each unit over the production life of the ATAES132 product family. The contents of this register can optionally be included in cryptographic calculations by setting Mode bit 6 to 1b, as described in the command definitions in Section 7, Command Definitions. This register cannot be changed by the customer.

It is recommended that the SerialNum Register value be used to perform key diversification.

E.2.2 LotHistory Register

LotHistory is an 8-byte, read-only register that is programmed by Atmel at the factory. This register contains proprietary data that is not intended for customer use. This register cannot be changed by the customer.

E.2.3 JEDEC Register

JEDEC is a 2-byte, read-only register that is programmed by Atmel at the factory. The JEDEC register always contains 0x001F, which is the JEDEC Manufacturing Identification Code assigned to Atmel. This register cannot be changed by the customer.

E.2.4 Algorithm Register

Algorithm is a 2-byte, read-only register that is programmed by Atmel at the factory. The default value of 0x0000 indicates 128-bit AES-CCM mode. This register cannot be changed by the customer.

E.2.5 EEPROMSize Register

EEPROMSize is a 1-byte, read-only register that is programmed by Atmel at the factory. The default value of 0x20 indicates a 32-byte physical EEPROM page size. This register cannot be changed by the customer.

E.2.6 EncReadSize Register

EncReadSize is a one-byte, read-only register that is programmed by Atmel at the factory. The default value of 0x20 indicates that 32 bytes is the maximum data length that can be returned by the EncRead command. This register cannot be changed by the customer.

E.2.7 EncWrtSize Register

EncWrtSize is a 1-byte, read-only register that is programmed by Atmel at the factory. The default value of 0x20 indicates that 32 bytes is the maximum data length that can be written using the EncWrite command. This register cannot be changed by the customer.
E.2.8 DeviceNum Register

DeviceNum is a 1-byte, read-only register that is programmed by Atmel at the factory. This byte indicates the device type (32Kb, ATAES1xx family). The INFO command returns this byte, along with a hardware revision byte, as shown in Table 7-29. This register cannot be changed by the customer.

Table E-2. DeviceNum Coding for INFO Command Response and DeviceNum in Configuration Memory Register

<table>
<thead>
<tr>
<th>Description</th>
<th>INFO DeviceNum</th>
<th>DeviceNum Register</th>
</tr>
</thead>
<tbody>
<tr>
<td>Production Parts</td>
<td>0x0A05</td>
<td>0x0A</td>
</tr>
</tbody>
</table>

See Section 7.12, INFO Command for the INFO command description.

E.2.9 LockKeys Register

LockKeys is a 1-byte register that controls write access to Key Memory. The default value of LockKeys is the unlocked state (0x55). The LockKeys Register can be changed only by using the Lock command (see Section 7.18, Lock Command). After the Lock command is run, this register will contain 0x00, and the Key Memory will be locked. It is impossible to unlock memory that has been locked.

E.2.10 LockSmall Register

LockSmall is a 1-byte register that controls write access to the SmallZone Register. The default value of LockSmall is the unlocked state (0x55). The LockSmall Register can be changed only by using the Lock command (see Section 7.18). After the Lock command is run, this register will contain 0x00, and the SmallZone Register will be locked. It is impossible to unlock memory that has been locked.

E.2.11 LockConfig Register

LockConfig is a 1-byte register that controls write access to Configuration Memory except the SmallZone Register. The default value of LockConfig is the unlocked state (0x55). The LockConfig Register can be changed only by using the Lock command (see Section 7.18). After the Lock command is run, this register will contain 0x00, and the Configuration Memory will be locked except for the SmallZone Register, which is controlled by the LockSmall Register. It is impossible to unlock memory that has been locked.

If the LockConfig register is unlocked (0x55), then the random number generator is latched in Test mode, and the Random Command will always return 16 bytes of 0xA5. The KeyCreate and Nonce commands will create nonrandom results when the RNG is in Test mode. If the LockConfig Register is locked (0x00), then the RNG generates random numbers, and the random KeyCreate and Nonce commands function normally.

E.2.12 Reserved Registers

Any Configuration Memory locations that are identified as reserved in Table E-1, the Configuration Memory map, are reserved by Atmel for future use. All reserved registers are read-only registers that are programmed by Atmel at the factory. These memory locations are programmed with Atmel proprietary data. The contents of the reserved registers will vary and are not intended for any customer use. These registers cannot be changed by the customer.

E.2.13 ManufacturingID Register

ManufacturingID is a 2-byte, read-only register that is programmed by Atmel at the factory. This register contains a customer-specific value. The default ManufacturingID Register contains 0x0000. This register cannot be changed by the customer.
E.2.14 PermConfig Register

PermConfig is a 1-byte read-only register that is programmed by Atmel at the factory. This register cannot be changed by the customer. The default value of 0x01 enables all cryptographic commands.

Table E-3. PermConfig Register Definition

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| Reserved for Future Use | EncryptE

If the EncryptE bit is 1b, then the Encrypt, Decrypt, and Legacy command availability is determined by the ChipConfig.EncDecrE and ChipConfig.LegacyE bits. If the EncryptE bit is 0b, then the Encrypt, Decrypt, and Legacy commands are disabled. See the ChipConfig Register definition in Appendix E.2.16, ChipConfig Register for additional information.

E.2.15 I2CAddr Register

I2CAddr is a 1-byte register that controls the ATAES132 serial interface. The customer can write the I2CAddr Register using standard I2C or SPI Write commands, unless the Configuration Memory has been locked (see Appendix E.2.11, LockConfig Register).

Table E-4. I2CAddr Register Definition

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
| I2C Device Address | SPI/I2C

Bit 0 selects the serial interface mode, 0b selects SPI interface mode, and 1b selects I2C interface mode. If bit 0 is 0b, then the contents of bits one to seven are ignored.

The default value of the I2CAddr Register depends on the ordering code (see Appendix Q, Ordering Information): I2CAddr is 0xA1 (the I2C Device Address is 0xA0) for catalog numbers with an I2C interface configuration, and I2CAddr is 0x00 for catalog numbers with a SPI interface configuration. See Appendix J, I2C Interface for the I2C interface specifications. See Appendix K, SPI Interface for the SPI interface specifications.

E.2.16 ChipConfig Register

ChipConfig is a 1-byte register that controls device-level functionality of the ATAES132. The customer can write the ChipConfig register using standard I2C or SPI Write commands, unless the Configuration Memory has been locked (see the LockConfig Register definition in Section E.2.11).

Table E-5. ChipConfig Register Definition

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
</table>
| PowerUpState | Reserved for Future Use | EncDecrE | LegacyE

If the ChipConfig.LegacyE bit is 1b, then the Legacy command (Section 7.17, Legacy Command) is enabled. If ChipConfig.LegacyE is 0b, then a parse error ReturnCode will be returned in response to a Legacy command. If the ChipConfig.EncDecrE bit is 1b, then the Encrypt command (Section 7.10, Encrypt Command) and Decrypt command (Section 7.8, Decrypt Command) are enabled. If ChipConfig.EncDecrE is 0b, then a parse error ReturnCode will be returned in response to an Encrypt command or Decrypt command.
The default configuration of the PermConfig Register allows the customer to control the availability of the Encrypt, Decrypt, and Legacy commands using the ChipConfig Register; however, the ChipConfig.EncDecrE bit and ChipConfig.LegacyE bit will be ignored if the ATAES132 is configured at the factory to disable external encryption (see the PermConfig Register definition in Appendix E.2.14, PermConfig Register).

Table E-6. Coding of the Power-UpState Bits in the ChipConfig Register

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Device goes to the Active state at Power-Up.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Device goes to the Standby state at Power-Up.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>Device goes to the Sleep state at Power-Up.</td>
</tr>
</tbody>
</table>

The ChipConfig.PowerUpState bits are used to configure the behavior of the ATAES132 at initial power-up. Table E-5 shows the definition of the ChipConfig.PowerUpState bits. See Appendix L, Power Management for detailed information regarding the ATAES132 power management functions.

The default value of the ChipConfig Register is 0xC3. In this configuration, the ATAES132 goes to the Active state at Power-Up, and the Encrypt, Decrypt, and Legacy commands are enabled.

E.2.17 RFU Registers

Any Configuration Memory locations that are identified as RFU in Table E-1, the Configuration Memory map, are registers in customer-writable memory that are reserved by Atmel for future use (in a future ATAES family product or in a major product revision). The default value of the RFU registers is 0xFF.

The customer can write the RFU registers using standard I2C or SPI Write commands, unless the Configuration Memory has been locked (see the LockConfig Register definition in Appendix E.2.11, LockConfig Register). The RFU registers should be programmed to 0xFF only; all other values are prohibited.

E.2.18 CounterConfig Registers

The 16 CounterConfig Registers are used to individually configure the 16 Counters. Each CounterConfig Register controls one Counter. CounterConfig 00 controls Counter 00, CounterConfig 01 controls Counter 01, etc.

Each CounterConfig register is a 2-byte array that is stored as shown in Table E-6. The customer can write the CounterConfig Registers using standard I2C or SPI Write commands unless the Configuration Memory has been locked (see the LockConfig Register definition in Section E.2.11). See Appendix H, Understanding Counters for additional Counter information.

Table E-7. Partial Configuration Memory Map Showing CounterConfig Register Byte Locations for Four Registers

<table>
<thead>
<tr>
<th>Address</th>
<th>0h</th>
<th>1h</th>
<th>2h</th>
<th>3h</th>
<th>4h</th>
<th>5h</th>
<th>6h</th>
<th>7h</th>
</tr>
</thead>
<tbody>
<tr>
<td>F060h-F067h</td>
<td>CounterConfig 0</td>
<td>CounterConfig 1</td>
<td>CounterConfig 2</td>
<td>CounterConfig 3</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Byte 0 | Byte 1 | Byte 0 | Byte 1 | Byte 0 | Byte 1 | Byte 0 | Byte 1 |
The CounterConfig Register imposes restrictions on the usage of the Counter command (see Section 7.5) with a Counter. The CounterConfig bits have no impact on the functionality of a Key Usage Counter. If a Counter is identified in a KeyConfig Register (see Appendix E.2.19) as a Key Usage Counter, then the Counter will increment each time the Key is used. The CounterConfig[CntID].IncrementOK is typically set to 0b to prohibit the Counter Command from incrementing a Key Usage Counter.

Table E-8. Definition of the CounterConfig Register Bits

<table>
<thead>
<tr>
<th>CounterConfig Field</th>
<th>Byte</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>IncrementOK</td>
<td>0</td>
<td>0</td>
<td>1b = Increments using the Counter command are permitted. 0b = Increments using the Counter command are prohibited.</td>
</tr>
<tr>
<td>RequireMAC</td>
<td>0</td>
<td>1</td>
<td>1b = Increment operation requires an input MAC. 0b = An input MAC is prohibited.</td>
</tr>
<tr>
<td>Reserved</td>
<td>0</td>
<td>2 to 7</td>
<td>Reserved for future use. All bits must be 0b.</td>
</tr>
<tr>
<td>IncrID</td>
<td>1</td>
<td>0 to 3</td>
<td>KeyID of the key used to generate the Counter command input MAC for increment operations.</td>
</tr>
<tr>
<td>MacID</td>
<td>1</td>
<td>4 to 7</td>
<td>KeyID of the key used to generate the Counter command output MAC for counter Read operations.</td>
</tr>
</tbody>
</table>

Note: 1. Changes to the CounterConfig Registers take effect immediately, which allows the functionality to be verified during the personalization process.

E.2.19 KeyConfig Registers

The 16 KeyConfig Registers are used to individually configure the 16 keys. Each KeyConfig Register controls one key. KeyConfig 00 controls Key 00, KeyConfig 01 controls Key 01, etc.

Each KeyConfig Register is a 4-byte array that is stored as shown in Table E-8. The customer can write the KeyConfig Registers using standard I²C or SPI Write commands, unless the Configuration Memory has been locked (see the LockConfig Register definition in Appendix E.2.11, LockConfig Register).

Table E-9. Partial Configuration Memory Map Showing KeyConfig Register Byte Locations for Two Registers

<table>
<thead>
<tr>
<th>Address</th>
<th>0h</th>
<th>1h</th>
<th>2h</th>
<th>3h</th>
<th>4h</th>
<th>5h</th>
<th>6h</th>
<th>7h</th>
</tr>
</thead>
<tbody>
<tr>
<td>F080h–F087h</td>
<td>KeyConfig 0</td>
<td>KeyConfig 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte 0</td>
<td>Byte 1</td>
<td>Byte 2</td>
<td>Byte 3</td>
<td>Byte 0</td>
<td>Byte 1</td>
<td>Byte 2</td>
<td>Byte 3</td>
<td></td>
</tr>
</tbody>
</table>

A key can be disabled by setting KeyConfig[KeyN].AuthKey to 1b and KeyConfig[KeyN].LinkPointer to contain “KeyN,” where the KeyN = KeyID of the key being configured.
### Table E-10. Definition of the KeyConfig Register Bits

<table>
<thead>
<tr>
<th>CounterConfig Field</th>
<th>Byte</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
</table>
| ExternalCrypto      | 0    | 0   | 1b = Key can be used with the Encrypt and Decrypt commands. <sup>(2)</sup>  
0b = Encrypt and Decrypt commands are prohibited. |
|                     |      |     |             |
| InboundAuth         | 0    | 1   | 1b = Key can be used only by the Auth command for inbound-only or mutual authentication. Key cannot be used by any other command, but KeyID can be the target of a Key Management command.  
0b = Key can be used for any purpose not prohibited by another KeyConfig bit, including outbound-only authentication. |
|                     |      |     |             |
| RandomNonce         | 0    | 2   | 1b = Operations using this key require a random Nonce (see Section 7.19).  
0b = The Nonce is not required to be random. |
|                     |      |     |             |
| LegacyOK            | 0    | 3   | 1b = Key can be used with the Legacy command.  
0b = Key cannot be used with the Legacy command. |
|                     |      |     |             |
| AuthKey             | 0    | 4   | 1b = Key requires prior authentication using the KeyID stored in LinkPointer.  
0b = Prior authentication is not required. |
|                     |      |     |             |
| Child               | 0    | 5   | 1b = Key is permitted to be the target of a KeyCreate for Child and Parent and/or KeyLoad command.  
0b = This use is prohibited. |
|                     |      |     |             |
| Parent              | 0    | 6   | 1b = Key may be used as the VolatileKey parent by the Key Create or KeyLoad commands. The key may also be used as the Decrypt Key by the KeyImport command when the target key is VolatileKey (see Section 4.3).  
0b = This use is prohibited. |
|                     |      |     |             |
| ChangeKeys          | 0    | 7   | 1b = Key updates are permitted after locking. The new key is written using the EncWrite command with a MAC generated with the current value of key (see Section 7.11).  
0b = Key updates with the EncWrite command are prohibited. |
|                     |      |     |             |
| CounterLimit        | 1    | 0   | 1b = Usage count limits are enabled for this key (see CounterNum).  
0b = There are no usage limits. |
|                     |      |     |             |
| ChildMac            | 1    | 1   | 1b = An input MAC is required to modify this key using the KeyCreate command.  
0b = KeyCreate command does not require an input MAC (it will be ignored if provided). |
|                     |      |     |             |
| AuthOut             | 1    | 2   | 1b = I2C Auth signaling is enabled for this key (see Appendix J.5).  
0b = I2C Auth signaling is disabled for this key. |
|                     |      |     |             |
| AuthOutHold         | 1    | 3   | 1b = I2C AuthO output state is unchanged when an Authentication Reset is executed using this key.  
0b = I2C AuthO output is reset when an Authentication Reset is executed using this key (see Appendix J.5). |
|                     |      |     |             |
| ImportOK            | 1    | 4   | 1b = Key is permitted to be the target of a KeyImport command.  
0b = KeyImport command is prohibited. |
|                     |      |     |             |
| ChildAuth           | 1    | 5   | 1b = The KeyCreate command requires prior authentication using the KeyID stored in LinkPointer.  
0b = Prior authentication is not required. |
|                     |      |     |             |
| TransferOK          | 1    | 6   | 1b = Key is permitted to be the target of a KeyTransfer command (see Section 7.16).  
0b = The KeyTransfer command is prohibited. |
<table>
<thead>
<tr>
<th>CounterConfig Field</th>
<th>Byte</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>AuthCompute</td>
<td>1</td>
<td>7</td>
<td>1b = Key can be used with the AuthCompute command. 0b = Key cannot be used with the AuthCompute command.</td>
</tr>
<tr>
<td>LinkPointer</td>
<td>2</td>
<td>0 to 3</td>
<td>For child keys, stores the ParentKeyID. For all other keys, the KeyID of the authorizing key (see AuthKey).</td>
</tr>
<tr>
<td>CounterNum</td>
<td>2</td>
<td>4 to 7</td>
<td>Stores the CntID of the counter attached to this key for usage limits and/or for MAC calculation. MAC calculations will include the counter if Command Mode bit 5 is 1b, even if key usage limits are disabled.</td>
</tr>
<tr>
<td>Reserved</td>
<td>3</td>
<td>0 to 7</td>
<td>Reserved for future use. All bits must be 0b.</td>
</tr>
</tbody>
</table>

Notes:
1. Changes to the KeyConfig Registers take effect immediately, which allows the functionality to be verified during the personalization process.
2. **Warning:** Since the Encrypt command does not include an input MAC, the Encrypt command can be exhaustively run with selected input data to attack the Key. Requiring authentication prior to allowing encryption makes these attacks more difficult. To require prior authentication, the AuthKey and RandomNonce bits must be set to 1b.
3. A Key can be disabled by setting KeyConfig[KeyN].AuthKey to 1b and KeyConfig[KeyN].LinkPointer to contain "KeyN", where KeyN = KeyID of the key being configured.

### E.2.20 ZoneConfig Registers

The 16 ZoneConfig Registers are used to individually configure the 16 user zones. Each ZoneConfig Register controls one user zone. ZoneConfig 00 controls User Zone 00, ZoneConfig 01 controls User Zone 01, etc.

Each ZoneConfig Register is a 4-byte array that is stored as shown in Table E-10. The customer can write the ZoneConfig Registers using standard I²C or SPI Write commands, unless the Configuration Memory has been locked (see the LockConfig Register definition in Appendix E.2.11, LockConfig Register).

**Table E-11. Partial Configuration Memory Map Showing ZoneConfig Register Byte Locations for the Two Registers**

<table>
<thead>
<tr>
<th>Address</th>
<th>0h</th>
<th>1h</th>
<th>2h</th>
<th>3h</th>
<th>4h</th>
<th>5h</th>
<th>6h</th>
<th>7h</th>
</tr>
</thead>
<tbody>
<tr>
<td>F0C0h-</td>
<td>ZoneConfig 0</td>
<td>ZoneConfig 1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Byte 0</td>
<td>Byte 1</td>
<td>Byte 2</td>
<td>Byte 3</td>
<td>Byte 0</td>
<td>Byte 1</td>
<td>Byte 2</td>
<td>Byte 3</td>
<td></td>
</tr>
</tbody>
</table>
Table E-12. Definition of the ZoneConfig Register Bits (1)

<table>
<thead>
<tr>
<th>CounterConfig Field</th>
<th>Byte</th>
<th>Bit</th>
<th>Description</th>
</tr>
</thead>
</table>
| AuthRead            | 0    | 0   | 1b = Authentication is required to read data.  
0b = Authentication is not required to read data. |
| AuthWrite           | 0    | 1   | 1b = Authentication is required to write data.  
0b = Authentication is not required to write data. |
| EncRead             | 0    | 2   | 1b = Encryption is required to read data.  
0b = Encryption is not required to read data. |
| EncWrite            | 0    | 3   | 1b = Encryption is required to write data.  
0b = Encryption is not required to write data. |
| WriteMode           | 0    | 4 to 5 | 00b = Zone is permanently Read/Write.  
01b = Zone is permanently Read-only.  
10b = The ReadOnly byte determines if writes are permitted.  
11b = The ReadOnly byte determines if writes are permitted, and the Lock command must include an authenticating MAC calculated using the KeyID stored in ZoneConfig[UZ].WriteID. |
| UseSerial           | 0    | 6   | UseSerial = 1b and EncWrite = 1b, then SerialNum must be included in EncWrite operations.  
EncWrite = 0b, then this bit is ignored. |
| UseSmall            | 0    | 7   | UseSmall = 1b and EncWrite = 1b, the first four bytes of SmallZone must be included in EncWrite operations.  
EncWrite = 0b, then this bit is ignored. |
| ReadID              | 1    | 0 to 3 | KeyID that is used to encrypt data read from this zone.  
The same key is used to generate the MAC. |
| AuthID              | 1    | 4 to 7 | KeyID that is used for inbound authentication before access is permitted. |
| Reserved            | 2    | 0 to 3 | Reserved for future use. All bits must be 0b. |
| WriteID             | 2    | 4 to 7 | KeyID that is used to decrypt data written to this zone.  
The same key is used to verify the MAC. |
| ReadOnly            | 3    | 0 to 7 | The contents of this byte are ignored unless WriteMode contains 10b or 11b.  
0x55 = User zone is Read/Write.  
If any other value = User zone is Read-only.  
This byte can be updated after the Configuration Memory is locked by using the Lock command (see Section 7.18, Lock Command). |

Note: 1. Most changes to the ZoneConfig Registers take effect immediately. Changes to the AuthRead and EncRead bits do not affect the SPI or I2C Read command until the next reset or power-up.

E.2.21 Counter Registers

The 16 Counter Registers are used to store the Counter values. The default value of the Counters is equivalent to a count value of zero. See Appendix H, Understanding Counters for Counter information.

The customer can write the Counter Registers using standard I2C or SPI Write commands, unless the Configuration Memory has been locked (see the LockConfig register definition in Appendix E.2.11, LockConfig Register).
E.2.22 FreeSpace Register

The FreeSpace Register is 96 bytes of memory for storage of customer data. The customer can write the FreeSpace Register using standard I2C or SPI Write commands, unless the Configuration Memory has been locked (see the LockConfig register definition in Appendix E.2.11).

The default value of the FreeSpace Register is 0xFF in all bytes. The FreeSpace Register can be programmed with any value; the contents will not change the behavior of the ATAES132.

E.2.23 SmallZone Register

The SmallZone Register is 32 bytes of memory for storage of customer data. Optionally, the first four bytes of the SmallZone Register may be included in cryptographic calculations by setting Mode bit 7 to 1b, as described in the command definitions in Section 7, Command Definitions. The customer can write the SmallZone Register using standard I2C or SPI Write commands, unless the SmallZone Register has been locked (see the LockSmall Register definition in Appendix E.2.10, LockSmall Register).

The default value of the SmallZone Register is 0xFF in all bytes. The SmallZone Register can be programmed with any value; the contents will not change the behavior of the ATAES132.
## Appendix F. Key Memory Map

Table F-1. ATAES132 Key Memory Map; Starts at Address 0xF200

<table>
<thead>
<tr>
<th>Address</th>
<th>$0_{h}$ / $8_{h}$</th>
<th>$1_{h}$ / $9_{h}$</th>
<th>$2_{h}$ / $A_{h}$</th>
<th>$3_{h}$ / $B_{h}$</th>
<th>$4_{h}$ / $C_{h}$</th>
<th>$5_{h}$ / $D_{h}$</th>
<th>$6_{h}$ / $E_{h}$</th>
<th>$7_{h}$ / $F_{h}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F200h-F207h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Key 00</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F208h-F20Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Key 01</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F210h-F217h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Key 02</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F218h-F21Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Key 03</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F220h-F227h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Key 04</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F228h-F22Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Key 05</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F230h-F237h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Key 06</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F238h-F23Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Key 07</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F240h-F247h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Key 08</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F248h-F24Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Key 09</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F250h-F257h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Key 0A</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F258h-F25Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Key 0B</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F260h-F267h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Key 0C</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F268h-F26Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Key 0D</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F270h-F277h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Key 0E</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F278h-F27Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Key 0F</td>
<td></td>
<td></td>
</tr>
<tr>
<td>F280h-F287h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F288h-F28Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F290h-F297h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F298h-F29Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2A0h-F2A7h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2A8h-F2AFh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2B0h-F2B7h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2B8h-F2BFh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2C0h-F2C7h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2C8h-F2CFh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2D0h-F2D7h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2D8h-F2DFh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2E0h-F2E7h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2E8h-F2EFh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2F0h-F2F7h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F2FFh-F2FFh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Key 00</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

VolatileKey (KeyID = 0xFF) does not exist in EEPROM. It is a temporary key that resides in the internal SRAM memory. The internal SRAM cannot be accessed directly. See Section 4.3, VolatileKey Configuration for VolatileKey information.

Prior to locking the Key Memory, it can be written with either encrypted or cleartext data. Encrypted writes are performed using the EncWrite command (see Section 7.11, Encrypted Key Writes). Cleartext writes are performed using standard SPI or I²C Write commands (see Section 5.2, Write). The Key Memory can never be read with the BlockRead command or the EncRead command, or with standard I²C or SPI Read commands.
Appendix G. Understanding the STATUS Register

The 8-bit Device Status Register is used for handshaking between the Host microcontroller and the ATAES132. The Host microcontroller is expected to read the STATUS Register before sending a command or reading a response.

G.1 Device Status Register (STATUS) Definition

Address 0xFFF0 contains the read-only Device Status Register, which indicates the current status of the ATAES132 device. The SPI Read Status Register command can be used to read the STATUS Register, as described in Appendix K.3.5, Read Status Register Command (RDSR).

This register can also be read with the standard I2C or SPI Read Memory commands. Reading the STATUS Register does not increment the memory read address, and so a Host microcontroller can easily monitor the ATAES132 device status by repeatedly reading the STATUS Register.

Table G-1. Device Status Register Definition

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EERR</td>
<td>RRDY</td>
<td>Reserved</td>
<td>CRCE</td>
<td>Reserved</td>
<td>WAKEb</td>
<td>WEN</td>
<td>WIP</td>
</tr>
</tbody>
</table>

Table G-2. Definition of the STATUS Register Bits

<table>
<thead>
<tr>
<th>Bit</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0 (WIP)</td>
<td>0b = The device is ready, waiting for a command. 1b = A Write cycle or a cryptographic operation is in progress.</td>
</tr>
<tr>
<td>Bit 1 (WEN)</td>
<td>0b = The device is not SPI Write enabled or is in I2C interface mode. 1b = The device is SPI Write enabled.</td>
</tr>
<tr>
<td>Bit 2 (WAKEb)</td>
<td>0b = The device is not in the Sleep or Standby power state. 1b = The device is in the Sleep or Standby power state.</td>
</tr>
<tr>
<td>Bit 3 (Reserved)</td>
<td>Always 0b. This bit is reserved for future use.</td>
</tr>
<tr>
<td>Bit 4 (CRCE)</td>
<td>0b = The most recent command block contained a correct Checksum (CRC). 1b = The most recent command block contained an error.</td>
</tr>
<tr>
<td>Bit 5 (Reserved)</td>
<td>Always 0b. This bit is reserved for future use.</td>
</tr>
<tr>
<td>Bit 6 (RRDY)</td>
<td>0b = The Response Memory Buffer is empty. 1b = The Response Memory Buffer is ready to read.</td>
</tr>
<tr>
<td>Bit 7 (EERR)</td>
<td>0b = The most recent command did not generate an error during execution. 1b = The most recent command generated an execution error.</td>
</tr>
</tbody>
</table>

Note: 1. When the SPI RDSR command is used to read the STATUS Register during an EEPROM Write or during execution of any ATAES132 command, then Status bits 0 – 7 are 1b (see Appendix K.3.5, Read Status Register Command (RDSR). When the STATUS Register is read from address 0xFFF0 under the same circumstances, the reserved bits will read as 0b.

2. STATUS Register bits 0 – 7 are 1b during wake-up. During the first phase of wake-up and power-up. See Appendix L, Power Management for additional information.
The Device Status Register can always be read when the ATAES132 is configured for SPI interface mode, even if ATAES132 is processing a command or writing the EEPROM. When the ATAES132 is configured for I²C interface mode, the Random Read command can only be used to read the STATUS Register only when the device address is ACKed.

If the ATAES132 is in the Sleep or Standby power state, reading the STATUS Register forces the ATAES132 to wake-up; the STATUS Register is 0xFF until the wake-up process is complete.

G.1.1 WIP Status Bit [0]
The WIP status bit is used to indicate the device is busy or there is a “Write in progress.” If WIP = 0b, then the ATAES132 is in the Active state and is waiting to receive a command. If WIP = 1b, then ATAES132 is in the Active state and is performing an EEPROM Write or processing an ATAES132 command.

G.1.2 WEN Status Bit [1]
If ATAES132 is configured in I²C interface mode, then the WEN Status bit is always 0b (see Appendix J, I²C Interface for I²C information).

If the ATAES132 is configured in SPI interface mode, then the WEN status bit is 0b after the device initially powers up or exits the Sleep state (see Appendix K, SPI Interface for SPI interface information). When WEN = 0b, the User Memory is Write protected and any attempt to write the User Memory using the SPI Write command will fail. The Host must send a SPI WREN command to the device to set WEN = 1b prior to each SPI Write command.

If the ATAES132 is configured in SPI interface mode, then the WEN Status bit will return to 0b when any Write instruction is received. The WEN Status bit can be forced to 0b by sending a SPI WRDI command (See Appendix K.3.2, Write Disable Command (WRDI)), by sending a RESET command (See Section 7.22, Reset Command), or by putting the device in the Sleep state. Powering the device off will reset the WEN bit to 0b. The SPI Read command and SPI RDSR command do not affect the state of the WEN bit.

It is not necessary to set WEN = 1b prior to writing to the Command Memory Buffer or the IO Address Reset Register (see Appendix D, Command Memory Map). Writing the Command Memory Buffer or the IO Address Reset Register forces WEN to 0b.

G.1.3 WAKEb Status Bit [2]
The WAKEb status bit is 0b when the ATAES132 has completed a power-up sequence and is in the active state. WAKEb is 1b when the ATAES132 is in the Sleep or Standby state, or is in the process of waking up.

Note: Reading the STATUS Register will cause a device in the Sleep state or Standby state to wake-up. (See Appendix L, Power Management for power state and power management information.)

G.1.4 CRCE Status Bit [4]
The CRCE status bit is set to 1b if a block is received with a short Count or bad Checksum or if the block causes a buffer overrun. If only the Checksum (CRC) was incorrect, then the block may be resent without change. If the Command Memory Buffer contains a partial command block, then the CRCE status bit is 1b and all other status bits are 0b. This indicates that the correct Checksum has not yet been received. If the CRCE Status bit is 1b and all the other Status bits are 0b after the entire block has been sent, the IO Address Reset Register should be written before resending the block (see Appendix D.2, Response Memory Buffer for more information on the IO Address Reset Register).

The EERR bit will remain 0b when a Checksum error occurs, and the Response Memory Buffer will remain empty because these errors do not result in a ReturnCode being generated. If a buffer overrun occurs, then the CRCE and EERR bits will be set to 1b.
G.1.5 RRDY Status Bit [6]
The RRDY Status bit is 0b when the Response Memory Buffer is empty. If RRDY = 1b, then the Response Memory Buffer contains a response block or a ReturnCode resulting from the most recent command or command block received (see Appendix D.2, Response Memory Buffer for Response Memory Buffer information).

G.1.6 EERR Status Bit [7]
If the command is processed without error, the EERR bit is set to 0b. When any error other than a Checksum error occurs, the EERR Status bit is set to 1b to indicate an error. The Host can read the error code (ReturnCode) from the Response Memory Buffer (address 0xFE00) using a Read command if the RRDY Status bit is 1b.

Reading the STATUS Register does not reset the Status Register bits or alter the Response Memory Buffer contents. Reading the Response Memory Buffer does not alter the contents of the Response Memory Buffer or the STATUS Register. Reading beyond the end of the Response Memory Buffer will not cause the STATUS Register bits to change.

The EERR status bit will be set to 1b if a SPI or I²C Read is attempted using an invalid address or an address pointing to a protected portion of the memory. EERR will also be set to 1b if a SPI or I²C read begins at an authorized address but continues into protected memory. In both of these cases, the RRDY status bit is 0b and the Response Memory Buffer will remain empty because these errors do not generate a ReturnCode. Reading beyond the end of user zone F will not cause the EERR bit to be set to 1b.

Note: If a SPI or I²C Read begins at an authorized address and continues into protected memory, the EERR bit will be set to 1b.

G.1.7 Reserved Status Bits [3, 5]
The Reserved Status bits are always 0b when the ATAES132 is capable of accepting a command. The Reserved Status bits are 1b during Power-Up and during Wake-Up from the Sleep state or the Standby state.

G.2 STATUS Register Behavior in the I²C Interface Mode
The following sections describe the device behavior and expected STATUS Register values during commonly performed operations. In the I²C interface mode, the ATAES132 will always NAK instructions containing a nonmatching I²C Device Address. The ATAES132 will ACK instructions with a matching I²C Device Address if the device is capable of accepting an instruction. See Appendix J, I²C Interface for the I²C interface specifications.

When the ATAES132 is busy or unable to respond for any reason, it will NAK a matching I²C Device Address. The ACK/NAK response to the I²C Device Address operates similar to the way the WIP status bit changes value in the SPI interface mode.

G.2.1 Power-Up
The ATAES132 will NAK all instructions received during Power-Up to indicate that it is not ready to accept a command from the Host. When the Power-Up process is complete (after time tPU.RDY), then the ATAES132 will enter the state specified by ChipConfig Register bits 6 and 7; the Active state, the Standby state, or the Sleep state (see Appendix L.2.1, Power-Up). In I²C interface mode, it is impossible to read the STATUS Register until the completion of Power-Up.

Upon completion of Power-Up, the Command Memory Buffer is empty, the Response Memory Buffer is all 0xFFs, and ChipState = 0xFFFF. The default EEPROM address is set to 0x0000, and the command and Response Memory Buffer pointers are set to the base address of the buffers. If the device is configured to enter the Active state at Power-Up, then STATUS will be 0x00, as shown in Table G-3.
### Table G-3. Contents of the STATUS Register After Power-up to the Active State

<table>
<thead>
<tr>
<th>Bit</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0 (WIP)</td>
<td>0b = Device is ready, waiting for a command.</td>
</tr>
<tr>
<td>Bit 1 (WEN)</td>
<td>0b = Device is in I²C interface mode.</td>
</tr>
<tr>
<td>Bit 2 (WAKEb)</td>
<td>0b = Device is not in the Sleep or Standby Power state.</td>
</tr>
<tr>
<td>Bit 3 (Reserved)</td>
<td>Always 0b.</td>
</tr>
<tr>
<td>Bit 4 (CRCE)</td>
<td>0b = No checksum error.</td>
</tr>
<tr>
<td>Bit 5 (Reserved)</td>
<td>Always 0b.</td>
</tr>
<tr>
<td>Bit 6 (RRDY)</td>
<td>0b = Response Memory Buffer is empty.</td>
</tr>
<tr>
<td>Bit 7 (EERR)</td>
<td>0b = No errors during execution.</td>
</tr>
</tbody>
</table>

If the device is configured to enter the Sleep state, then the ATAES132 will NAK any attempt to read the STATUS Register at the completion of Power-Up, as described in Appendix G.2.2, Wake-Up from Sleep. If the device is configured to enter the Standby state, then the ATAES132 will NAK any attempt to read the STATUS Register at the completion of Power-Up, as described in Appendix G.2.3, Wake-Up from Standby; ChipState will remain 0xFFFF in the Standby state.

**Note:** ACK polling or attempting to read the STATUS Register after Power-Up is completed will cause the device to Wake-Up.

### G.2.2 Wake-Up from Sleep

The ATAES132 will NAK all instructions received during Wake-Up from the Sleep Power state to indicate that it is not ready to accept a command from the Host. When the Wake-Up process is complete (after time $t_{WupSL.RDY}$), then the ATAES132 will enter the Active state. In I²C interface mode, it is impossible to read the STATUS Register until the Wake-Up is complete.

Upon completion of Wake-Up from Sleep, the Command Memory Buffer is empty, the Response Memory Buffer is all 0xFFs, and ChipState = 0x5555. The default EEPROM address is set to 0x0000, and the command and Response Memory Buffer pointers are set to the base address of the buffers. Upon completion of Wake-Up, the STATUS Register will be 0x00, as shown in Table G-3.

### G.2.3 Wake-Up from Standby

The ATAES132 will NAK all instructions received during Wake-Up from the Standby Power state to indicate that it is not ready to accept a command from the Host. When the Wake-Up process is complete (after time $t_{WupSB.RDY}$), the ATAES132 will enter the Active state. In I²C interface mode, it is impossible to read the STATUS Register until the Wake-Up is complete.

Upon completion of Wake-Up from Standby, the Command Memory Buffer is empty, and the Response Memory Buffer is all 0xFFs. ChipState will have the value it had prior to entering the Standby state. Upon completion of Wake-Up, the STATUS Register will be 0x00, as shown in Table G-3.
G.2.4 Read STATUS Register

To read the STATUS Register, the Host sends a Random Read Instruction (RREAD) with a starting address of 0xFFF0 when ATAES132 ACKs the I²C Device Address. Reading the STATUS Register does not increment the Read address, so the Host can poll the STATUS by reading any number of bytes, beginning with address 0xFFF0.

Reading the STATUS Register does not change the Command Memory Buffer contents or the Response Memory Buffer contents. Reading the STATUS Register does not change the Command Memory Buffer pointer or the Response Memory Buffer pointer. Reading the STATUS Register does not change the STATUS Register.

G.2.5 Read User Memory

The ATAES132 instructions for directly reading the User Memory are identical to the standard Atmel Serial EEPROM instructions. The Host can send a read memory instruction (READ, RREAD, SREAD) whenever the ATAES132 ACKs the I²C Device Address. If the address being read is valid and access is not prohibited, then the contents of that byte will be returned to the Host. If the address is invalid, or access is prohibited for any reason, then 0xFF will be returned to the Host in place of the prohibited byte.

A Read operation begins with an I²C Start condition and ends with an I²C NAK by the Host. If one or more bytes are accessed during the Read operation at an invalid or protected address, then the EERR bit will be set to 1b (see Table G-4). If all bytes accessed by the Read operation are valid and the Host satisfied the required access conditions, then the EERR bit will be set to 0b. The contents of the Command Memory Buffer and the Response Memory Buffer will remain unchanged.

Note: If an I²C Read begins at an authorized address and continues into protected memory, the EERR bit will be set to 1b.

Table G-4. Contents of the STATUS Register After an I²C Read Memory Operation

<table>
<thead>
<tr>
<th>Bit</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0 (WIP)</td>
<td>0b = Device is ready, waiting for a command.</td>
</tr>
<tr>
<td>Bit 1 (WEN)</td>
<td>0b = Device is in I²C interface mode.</td>
</tr>
<tr>
<td>Bit 2 (WAKEb)</td>
<td>0b = Device is not in the Sleep or Standby power state.</td>
</tr>
<tr>
<td>Bit 3 (Reserved)</td>
<td>Always 0b.</td>
</tr>
<tr>
<td>Bit 4 (CRCE)</td>
<td>0b = No Checksum error.</td>
</tr>
<tr>
<td>Bit 5 (Reserved)</td>
<td>Always 0b.</td>
</tr>
<tr>
<td>Bit 6 (RRDY)</td>
<td>0b = Response Memory Buffer is unchanged. (1)</td>
</tr>
<tr>
<td>Bit 7 (EERR)</td>
<td>0b = No errors during execution of the Read operation. 1b = 0xFF was returned in place of one or more invalid or prohibited bytes read.</td>
</tr>
</tbody>
</table>

Note: 1. A Read Memory operation does not change the contents of the Response Memory Buffer. The EERR status bit is used to indicate success or an error. No ReturnCode is generated by a memory read error.
G.2.6 Write User Memory

The ATAES132 instructions for directly writing the User Memory are identical to the standard Atmel Serial EEPROM. The Host can send a Write Memory instruction (BWRITE, PWRITE) whenever the ATAES132 ACKs the I²C Device Address. If the address being written is valid, access requirements have been satisfied and no page boundaries are crossed, then the data provided by the Host will be written after the Host generates an I²C Stop condition. If the address is invalid or access is prohibited for any reason, then the ATAES132 will discard the data and no EEPROM Write will occur.

A Memory Write operation begins with an I²C Start condition and ends with an I²C Stop condition by the Host. If the Host does not provide an I²C Stop condition, then no Write will occur, no ReturnCode will be generated, and the STATUS Register is 0x00 to indicate the ATAES132 is waiting for a command.

If the Host provides the required I²C Stop condition, then the ATAES132 will NAK the I²C Device Address during the EEPROM Write operation. When the Write operation is complete, then ATAES132 will ACK the I²C Device Address.

Upon completion of a Memory Write operation, the Command Memory Buffer is empty, and the Response Memory Buffer contains a ReturnCode. The command and the Response Memory Buffer pointers are set to the base address of the buffers. The STATUS will be as shown in Table G-5.

Table G-5. STATUS Register Contents After an I²C Write Memory Operation

<table>
<thead>
<tr>
<th>Bit</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0 (WIP)</td>
<td>0b = Device is ready, waiting for a command.</td>
</tr>
<tr>
<td>Bit 1 (WEN)</td>
<td>0b = Device is in I²C interface mode.</td>
</tr>
<tr>
<td>Bit 2 (WAKEb)</td>
<td>0b = Device is not in the Sleep or Standby power state.</td>
</tr>
<tr>
<td>Bit 3 (Reserved)</td>
<td>Always 0b.</td>
</tr>
<tr>
<td>Bit 4 (CRCE)</td>
<td>0b = No Checksum error.</td>
</tr>
<tr>
<td>Bit 5 (Reserved)</td>
<td>Always 0b.</td>
</tr>
<tr>
<td>Bit 6 (RRDY)</td>
<td>1b = Response Memory Buffer contains a response block.</td>
</tr>
<tr>
<td>Bit 7 (EERR)</td>
<td>0b = No errors during execution of the Write operation. 1b = Write operation generated an error; see the ReturnCode for the cause.</td>
</tr>
</tbody>
</table>

G.2.7 Write Command Memory Buffer

To write the Command Memory Buffer, the Host sends a Write Memory instruction (BWRITE, PWRITE) with a starting address of 0xFE00 when the ATAES132 ACKs the I²C Device Address. As each byte is written, the Command Memory Buffer pointer increments by one.

A command block begins with the COUNT byte and ends with the 2-byte Checksum (see Section 6.1, Command Block and Packet). If the entire command block is not received, then the device will not attempt to process the command and will not generate a response block. The STATUS Register will have CRCE = 1b until the entire command block is received (as shown in Table G-6).
### Table G-6. Contents of the STATUS Register if the Command Memory Buffer Contains a Partial Command Block

<table>
<thead>
<tr>
<th>Bit</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0 (WIP)</td>
<td>0b = Device is ready, waiting for a command.</td>
</tr>
<tr>
<td>Bit 1 (WEN)</td>
<td>0b = Device is in I(^2)C interface mode.</td>
</tr>
<tr>
<td>Bit 2 (WAKEb)</td>
<td>0b = Device is not in the Sleep or Standby power state.</td>
</tr>
<tr>
<td>Bit 3 (Reserved)</td>
<td>Always 0b.</td>
</tr>
<tr>
<td>Bit 4 (CRCE)</td>
<td>1b = Checksum error (the Checksum has not yet been received).</td>
</tr>
<tr>
<td>Bit 5 (Reserved)</td>
<td>Always 0b.</td>
</tr>
<tr>
<td>Bit 6 (RRDY)</td>
<td>0b = Response Memory Buffer is unchanged.</td>
</tr>
<tr>
<td>Bit 7 (EERR)</td>
<td>0b = No errors during execution of the command block (it was not executed yet).</td>
</tr>
</tbody>
</table>

If the Host provides a complete command block, then the ATAES132 will NAK the I\(^2\)C Device Address during command processing. When command processing is complete, then the ATAES132 will ACK the I\(^2\)C Device Address.

If the command block contains a bad Checksum or a short Count or if the block causes a buffer overrun, then the CRCE bit of the STATUS Register will be set to 1b, as shown in Table G-7. The Response Memory Buffer will be unchanged because no ReturnCode is generated by these error conditions. The EERR bit is 1b if a buffer overrun error occurs. The EERR bit is 0b if a bad Checksum or short Count error occurs.

If the command block contains a good Checksum, then the ATAES132 will process the command and load the response in the Response Memory Buffer. Upon completion of command processing, the RRDY bit of the STATUS Register is set to 1b, as shown in Table G-7.

### Table G-7. Contents of the STATUS Register After an I\(^2\)C Write Command Memory Buffer Resulting in CRCE = 1b

<table>
<thead>
<tr>
<th>Bit</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0 (WIP)</td>
<td>0b = Device is ready, waiting for a command.</td>
</tr>
<tr>
<td>Bit 1 (WEN)</td>
<td>0b = Device is in I(^2)C interface mode.</td>
</tr>
<tr>
<td>Bit 2 (WAKEb)</td>
<td>0b = Device is not in the Sleep or Standby power state.</td>
</tr>
<tr>
<td>Bit 3 (Reserved)</td>
<td>Always 0b.</td>
</tr>
<tr>
<td>Bit 4 (CRCE)</td>
<td>1b = Checksum, Short Count, or command buffer overrun error.</td>
</tr>
<tr>
<td>Bit 5 (Reserved)</td>
<td>Always 0b.</td>
</tr>
<tr>
<td>Bit 6 (RRDY)</td>
<td>0b = Response Memory Buffer is unchanged.</td>
</tr>
<tr>
<td>Bit 7 (EERR)</td>
<td>0b = No errors during execution of the command block (it was not executed). 1b = Command buffer overrun error.</td>
</tr>
</tbody>
</table>
Table G-8. Contents of the STATUS Register After an I²C Write Command Memory Buffer Resulting in CRCE = 0b

<table>
<thead>
<tr>
<th>Bit</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0 (WIP)</td>
<td>0b = Device is ready, waiting for a command.</td>
</tr>
<tr>
<td>Bit 1 (WEN)</td>
<td>0b = Device is in I²C interface mode.</td>
</tr>
<tr>
<td>Bit 2 (WAKEb)</td>
<td>0b = Device is not in the Sleep or Standby power state.</td>
</tr>
<tr>
<td>Bit 3 (Reserved)</td>
<td>Always 0b.</td>
</tr>
<tr>
<td>Bit 4 (CRCE)</td>
<td>0b = No Checksum error.</td>
</tr>
<tr>
<td>Bit 5 (Reserved)</td>
<td>Always 0b.</td>
</tr>
<tr>
<td>Bit 6 (RRDY)</td>
<td>1b = Response Memory Buffer contains a response block.</td>
</tr>
<tr>
<td>Bit 7 (EERR)</td>
<td>0b = No errors during execution of the command block.</td>
</tr>
<tr>
<td></td>
<td>1b = Command block generated an error; see the ReturnCode for the cause.</td>
</tr>
</tbody>
</table>

Writing the Command Memory Buffer resets the Response Memory Buffer pointer to the base address. Writing the Command Memory Buffer does not change the Response Memory Buffer contents until the entire command block is received and processed.

The Host can rewrite the contents of the Command Memory Buffer by resetting the buffer pointer (by writing the IO Address Reset Register) and sending a Write Memory instruction (BWRITE, PWRITE) with a starting address of 0xFE00.

Note: If the Host must write the Command Memory Buffer with more bytes than is required to send the command block due to hardware limitations, then the Host should transmit 0xFF bytes after the checksum. The extra bytes will be discarded by the ATAES132 and will not result in a buffer overrun or any other error.

G.2.8 Read Response Memory Buffer

To read the Response Memory Buffer, the Host sends a Random Read Memory instruction (RREAD) with a starting address of 0xFE00 when the ATAES132 ACKs the I²C Device Address. The Host can read any number of bytes from the Response Memory Buffer without causing an error. As each byte is read, the Response Memory Buffer pointer increments by one. If the Host reads beyond the end of the response block, then 0xFF will be returned for any byte after the Checksum.

Reading the Response Memory Buffer does not change the Command Memory Buffer contents or the Response Memory Buffer contents. Reading the Response Memory Buffer resets the Command Memory Buffer pointer to the base address. Reading the Response Memory Buffer does not change the STATUS Register.

The Host can reread the contents of the Response Memory Buffer by resetting the buffer pointer (by writing the IO Address Reset Register) and sending a Random Read Memory instruction (RREAD) with a starting address of 0xFE00.

G.2.9 Write IO Address Reset Register

To reset the pointer for the Command Memory Buffer and the pointer for the Response Memory Buffer, the Host sends a Write Memory instruction (BWRITE, or PWRITE) with a starting address of 0xFFE0. The IO Address Reset Register can be written with 1 to 32 bytes of data without generating an error; the data bytes will be ignored. The command and the Response Memory Buffer pointers are set to the base address of the buffers. The Command Memory Buffer is empty, and the Response Memory Buffer contents are unchanged. Writing the IO Address Reset Register changes the CRCE Status bit to 0b; all of the other status bits are unchanged.
G.3 STATUS Register Behavior in the SPI Interface Mode

The following sections describe the device behavior and expected STATUS Register values during commonly performed operations. See Appendix K, SPI Interface for the SPI interface specifications. In SPI interface mode, there are two ways to read the STATUS Register:

- Using the SPI RDSR command, or
- Reading STATUS from address 0xFFF0.

When the ATAES132 is busy or unable to respond for any reason, the WIP Status bit is 1b.

G.3.1 Power-Up

ATAES132 will return 0xFF in response to a SPI RDSR command during Power-Up to indicate that it is not ready to accept a command from the Host. When the power-up process is complete (after time \( t_{\text{PU.RDY}} \)), the ATAES132 will enter the state specified by ChipConfig Register bits 6 and 7 (see Appendix L.2.1, Power-Up): the Active state, the Standby state, or the Sleep state.

Upon completion of Power-Up, the Command Memory Buffer is empty, the Response Memory Buffer is all 0xFFs, and ChipState = 0xFFFF. The default EEPROM address is set to 0x0000, and the command and Response Memory Buffer pointers are set to the base address of the buffers. If the device is configured to enter the Active state, then the STATUS will be 0x00, as shown in Table G-9.

### Table G-9. Contents of the STATUS Register After Power-up to the Active State

<table>
<thead>
<tr>
<th>Bit</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0 (WIP)</td>
<td>0b = Device is ready, waiting for a command.</td>
</tr>
<tr>
<td>Bit 1 (WEN)</td>
<td>0b = Device is not Write enabled.</td>
</tr>
<tr>
<td>Bit 2 (WAKEb)</td>
<td>0b = Device is not in the Sleep or Standby power state.</td>
</tr>
<tr>
<td>Bit 3 (Reserved)</td>
<td>Always 0b.</td>
</tr>
<tr>
<td>Bit 4 (CRCE)</td>
<td>0b = No Checksum error.</td>
</tr>
<tr>
<td>Bit 5 (Reserved)</td>
<td>Always 0b.</td>
</tr>
<tr>
<td>Bit 6 (RRDY)</td>
<td>0b = Response Memory Buffer is empty.</td>
</tr>
<tr>
<td>Bit 7 (EERR)</td>
<td>0b = No errors during execution.</td>
</tr>
</tbody>
</table>

If the device is configured to enter the Standby or Sleep mode after power-up, then the STATUS will be 0xFF at the completion of the power-up process as described in this section. STATUS will remain 0xFF while the device is in Standby or Sleep mode.

Note: Reading the STATUS Register after Power-Up is completed will cause the device to Wake-Up.
G.3.2 Wake-Up State from Sleep State

ATAES132 will return 0xFF in response to a SPI RDSR command during Wake-Up from the Sleep Power state to indicate it is not ready to accept a command from the Host. When the wake-up process is complete (after time \( t_{\text{WupSL.RDY}} \)), ATAES132 will enter the Active state. After time \( t_{\text{WupSL.STATUS}} \), it is possible to read the STATUS Register.

Upon completion of Wake-Up state from Sleep state, the following occurs:

- Command Memory Buffer is empty,
- Response Memory Buffer is all 0xFFs,
- ChipState = 0x5555,
- Default EEPROM address is set to 0x0000,
- Command and Response Memory buffer pointers are set to the base address of the buffers.

Upon completion of Wake-Up the STATUS will be 0x00 as shown in Table G-3.

G.3.3 Wake-Up State from Standby State

ATAES132 will return 0xFF in response to a SPI RDSR command during Wake-Up state from the Standby Power state to indicate that it is not ready to accept a command from the Host. When the wake-up process is complete (after time \( t_{\text{WupSB.RDY}} \)), ATAES132 will enter the Active state. After time \( t_{\text{WupSB.STATUS}} \), it is possible to read the STATUS Register.

Upon completion of the Wake-Up state from the Standby state, the

- Command Memory Buffer is empty,
- Response Memory Buffer is all 0xFFs,
- ChipState will be the value it had prior to entering the Standby state.

Upon completion of the wake-up process, the STATUS will be 0x00 as shown in Table G-3.

G.3.4 Read STATUS Register

To read the STATUS Register, the Host sends a Read Memory Instruction (READ) with a starting address of 0xFFF0.

Reading the STATUS Register does not change the Command Memory Buffer contents or the Response Memory Buffer contents. Reading the STATUS Register does not change the Command Memory Buffer pointer or the Response Memory Buffer pointer. Reading the STATUS Register does not change the STATUS Register.

G.3.5 Read User Memory

The ATAES132 instructions for directly reading the User Memory are identical to standard Atmel Serial EEPROM instructions. The Host can send a Read whenever WIP is 0b.

- If the address being read is valid and access is not prohibited, the contents of that byte will be returned to the Host.
- If the address is invalid or access is prohibited for any reason, 0xFF will be returned to the Host in place of the prohibited byte.
- If one or more bytes are accessed during the Read operation at an invalid or protected address, then the EERR bit will be set to 1b (see Table G-10).
- If all bytes accessed by the Read operation are valid and the Host satisfied the required access conditions, the EERR bit will be set to 0b.

The contents of the Command Memory Buffer and the Response Memory Buffer will remain unchanged.

Table G-10. STATUS Register Contents After a SPI Read Memory Operation
<table>
<thead>
<tr>
<th>Bit</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0 (WIP)</td>
<td>0b = Device is ready, waiting for a command.</td>
</tr>
<tr>
<td>Bit 1 (WEN)</td>
<td>0b = Device is not Write enabled.</td>
</tr>
<tr>
<td>Bit 2 (WAKEb)</td>
<td>0b = Device is not in the Sleep or Standby Power state.</td>
</tr>
<tr>
<td>Bit 3 (Reserved)</td>
<td>Always 0b.</td>
</tr>
<tr>
<td>Bit 4 (CRCE)</td>
<td>0b = No Checksum error.</td>
</tr>
<tr>
<td>Bit 5 (Reserved)</td>
<td>Always 0b.</td>
</tr>
<tr>
<td>Bit 6 (RRDY)</td>
<td>0b = Response Memory Buffer is unchanged. (1)</td>
</tr>
<tr>
<td>Bit 7 (EERR)</td>
<td>0b = No errors during the execution of the Read operation.</td>
</tr>
<tr>
<td></td>
<td>1b = 0xFF was returned in place of one or more invalid or prohibited bytes read</td>
</tr>
</tbody>
</table>

Note: 1. A Read memory operation does not change the contents of the Response Memory Buffer. The EERR Status bit is used to indicate success or to indicate an error. No ReturnCode is generated by a memory Read error.

### G.3.6 Write User Memory

The ATAES132 instructions for directly writing the User Memory are identical to standard Atmel Serial EEPROMs. The Host can send a Write Memory Instruction (WRITE) whenever WIP is 0b.

- Data provided by the Host will be written if:
  - The address being written is valid,
  - Access requirements have been satisfied, and
  - No page boundaries are crossed.

- ATAES132 will discard the data and no EEPROM Write will occur if:
  - The address is invalid or
  - Access is prohibited for any reason.

Upon completion of a Memory Write operation:

- Command Memory Buffer is empty,
- Response Memory Buffer contains a ReturnCode,
- Command and Response Memory buffer pointers are set to the base address of the buffers,
- STATUS will be as shown in Table G-11.
Table G-11.  STATUS Register Contents After a SPI Write Memory Operation

<table>
<thead>
<tr>
<th>Bit</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0 (WIP)</td>
<td>0b = Device is ready, waiting for a command.</td>
</tr>
<tr>
<td>Bit 1 (WEN)</td>
<td>0b = Device is not Write enabled.</td>
</tr>
<tr>
<td>Bit 2 (WAKEb)</td>
<td>0b = Device is not in the Sleep or Standby power state.</td>
</tr>
<tr>
<td>Bit 3 (Reserved)</td>
<td>Always 0b.</td>
</tr>
<tr>
<td>Bit 4 (CRCE)</td>
<td>0b = No Checksum error.</td>
</tr>
<tr>
<td>Bit 5 (Reserved)</td>
<td>Always 0b.</td>
</tr>
<tr>
<td>Bit 6 (RRDY)</td>
<td>0b = Response Memory Buffer contains a response block.(^1)</td>
</tr>
<tr>
<td>Bit 7 (EERR)</td>
<td>0b = No errors during the execution of the Write operation.</td>
</tr>
<tr>
<td></td>
<td>1b = Write operation generated an error. See the ReturnCode for the cause.</td>
</tr>
</tbody>
</table>

G.3.7 Write Command Memory Buffer

To write the Command Memory Buffer, the Host sends a Write Memory Instruction (WRITE) with a starting address of 0xFE00 whenever WIP is 0b. The Command Memory Buffer pointer increments by one as each byte is written.

A Command Block begins with the COUNT byte and ends with the two byte Checksum (see Section 6.1, Command Block and Packet). If the entire Command Block is not received, then the device will not attempt to process the command; it will not generate a Response Block. The STATUS Register will have the CRCE bit = 1b until the entire Command block is received (as shown in Table G-12).

Table G-12.  STATUS Register Contents If the Command Memory Buffer Contains a Partial Command Block

<table>
<thead>
<tr>
<th>Bit</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0 (WIP)</td>
<td>0b = Device is ready, waiting for a command.</td>
</tr>
<tr>
<td>Bit 1 (WEN)</td>
<td>0b = Device is not Write enabled.</td>
</tr>
<tr>
<td>Bit 2 (WAKEb)</td>
<td>0b = Device is not in the Sleep or Standby power state.</td>
</tr>
<tr>
<td>Bit 3 (Reserved)</td>
<td>Always 0b.</td>
</tr>
<tr>
<td>Bit 4 (CRCE)</td>
<td>0b = No Checksum error (The checksum has not yet been received).</td>
</tr>
<tr>
<td>Bit 5 (Reserved)</td>
<td>Always 0b.</td>
</tr>
<tr>
<td>Bit 6 (RRDY)</td>
<td>0b = Response Memory Buffer is unchanged.</td>
</tr>
<tr>
<td>Bit 7 (EERR)</td>
<td>0b = No errors during the execution of the Command Block (It was not executed yet).</td>
</tr>
</tbody>
</table>

If the Host provides a complete Command Block, then WIP will be 1b during command processing. When command processing is complete, then WIP will be 0b.

If the Command Block contains a bad Checksum and a short COUNT or the block causes a buffer overrun, then the CRCE bit of the STATUS Register will be set to 1b as shown in Table G-13. The Response Memory Buffer will be unchanged because no ReturnCode is generated by these error conditions. The EERR Status bit is 1b if a buffer overrun error occurs; the EERR bit is 0b if a bad Checksum or short COUNT error occurs.

If the Command Block contains a good Checksum, then ATAES132 will process the command and load the response in the Response Memory Buffer. Upon completion, command processing the RRDY bit of the STATUS Register is set to 1b as shown in Table G-14.
### Table G-13. STATUS Register Contents After a SPI Write Command Memory Buffer Resulting in CRCE = 1b

<table>
<thead>
<tr>
<th>Bit</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0 (WIP)</td>
<td>0b = Device is ready, waiting for a command.</td>
</tr>
<tr>
<td>Bit 1 (WEN)</td>
<td>0b = Device is not Write enabled.</td>
</tr>
<tr>
<td>Bit 2 (WAKEb)</td>
<td>0b = Device is not in the Sleep or Standby power state.</td>
</tr>
<tr>
<td>Bit 3 (Reserved)</td>
<td>Always 0b.</td>
</tr>
<tr>
<td>Bit 4 (CRCE)</td>
<td>1b = Checksum error, short COUNT, or command buffer overrun error.</td>
</tr>
<tr>
<td>Bit 5 (Reserved)</td>
<td>Always 0b.</td>
</tr>
<tr>
<td>Bit 6 (RRDY)</td>
<td>0b = Response Memory Buffer is unchanged.</td>
</tr>
<tr>
<td>Bit 7 (EERR)</td>
<td>0b = No errors during the execution of the Command Block. (It was not executed yet.)</td>
</tr>
<tr>
<td></td>
<td>1b = Command buffer overrun error.</td>
</tr>
</tbody>
</table>

### Table G-14. STATUS Register Contents After a SPI Write Command Memory Buffer Resulting in CRCE = 0b

<table>
<thead>
<tr>
<th>Bit</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Bit 0 (WIP)</td>
<td>0b = Device is ready, waiting for a command.</td>
</tr>
<tr>
<td>Bit 1 (WEN)</td>
<td>0b = Device is not Write enabled.</td>
</tr>
<tr>
<td>Bit 2 (WAKEb)</td>
<td>0b = Device is not in the Sleep or Standby power state.</td>
</tr>
<tr>
<td>Bit 3 (Reserved)</td>
<td>Always 0b.</td>
</tr>
<tr>
<td>Bit 4 (CRCE)</td>
<td>0b = No Checksum error.</td>
</tr>
<tr>
<td>Bit 5 (Reserved)</td>
<td>Always 0b.</td>
</tr>
<tr>
<td>Bit 6 (RRDY)</td>
<td>1b = Response Memory Buffer contains a Response block.</td>
</tr>
<tr>
<td>Bit 7 (EERR)</td>
<td>0b = No errors during the execution of the Command Block. (It was not executed yet.)</td>
</tr>
<tr>
<td></td>
<td>1b = Command buffer generated an error. See the ReturnCode for the cause.</td>
</tr>
</tbody>
</table>

Writing the Command Memory Buffer resets the Response Memory Buffer pointer to the base address. Writing the Command Memory Buffer does not change the Response Memory Buffer contents until the entire Command block is received and processed.

The Host can rewrite the contents of the Command Memory Buffer by resetting the buffer pointer (by writing the IO Address Reset Register) and sending a Write Memory Instruction (WRITE) with a starting address of 0xFE00.

**Note:** If the Host must write the Command Memory Buffer with more bytes than is required to send the Command Block due to hardware limitations, then the Host should transmit 0xFF bytes after the Checksum. The extra bytes will be discarded by ATAES132 and will not result in a buffer overrun or any other error.
G.3.8 Read Response Memory Buffer

To read the Response Memory Buffer, the Host sends a Read Memory Instruction (READ) with a starting address of 0xFE00. The Host can read any number of bytes from the Response Memory Buffer without causing an error. As each byte is read, the Response Memory Buffer pointer increments by one. If the Host reads beyond the end of the Response Block, then 0xFF will be returned for any byte after the Checksum.

Reading the Response Memory Buffer does not change the Command Memory Buffer contents or the Response Memory Buffer contents. Reading the Response Memory Buffer resets the Command Memory Buffer pointer to the base address. Reading the Response Memory Buffer does not change the STATUS Register.

The Host can reread the contents of the Response Memory Buffer by resetting the buffer pointer (by writing the IO Address Reset Register) and sending a Random Read Memory Instruction (RREAD) with a starting address of 0xFE00.

G.3.9 Write IO Address Reset Register

To reset the pointer for the Command Memory Buffer and the pointer for the Response Memory Buffer, the Host sends a Write Memory Instruction (WRITE) with a starting address of 0xFFE0. The IO Address Reset Register can be written with 1 to 32 bytes of data without generating an error; the data bytes will be ignored. The Command and Response Memory buffer pointers are set to the base address of the buffers. The Command Memory Buffer is empty, and the Response Memory Buffer contents are unchanged. Writing the IO Address Reset Register changes the CRCE Status bit to 0b; all of the other STATUS bits are unchanged.
Appendix H. Understanding Counters

Each Counter can increment up to a value of 2,097,151 using the Count command, after which, the Counter can no longer be changed. Counters attached to keys are incremented each time the key is used when the Usage Counter reaches its limit; the key is disabled. Counters can also be incremented using the Count Command. The value in the Counter can never be reset or lowered. The Counters include a power interruption protection feature to prevent corruption of the Count value if power is removed during the increment operation.

On shipment from Atmel, the Counter Registers are initialized to their lowest value. The initial value of each Counter may be written to a different value at personalization prior to locking the configuration.

H.1 Counter Registers

Each Counter Register contains two Count values to prevent the Count value from being corrupted if power is interrupted during a Counter increment operation. Each Count value is stored as a combination of two Count fields:

- Counter A is stored in LinCountA and BinCountA.
- Counter B is stored in LinCountB and BinCountB.

Table H-1 shows the location of the Count fields within the Counter register in Configuration Memory.

<table>
<thead>
<tr>
<th>Address</th>
<th>0h</th>
<th>1h</th>
<th>2h</th>
<th>3h</th>
<th>4h</th>
<th>5h</th>
<th>6h</th>
<th>7h</th>
</tr>
</thead>
<tbody>
<tr>
<td>F100h-F107h</td>
<td>Counter 00</td>
<td>LinCountA</td>
<td>LinCountB</td>
<td>BinCountB</td>
<td>BinCountA</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Counter Registers can always be read from the Configuration Memory using the BlockRead command; however, the Count command is the preferred method of reading the Counters. When the Counter is read using the Count command, ATAES132 automatically selects the appropriate Counter register fields and returns them to the Host in the Response Packet. See Section 7.5, Counter Command.
H.2 Reading the Counter

The Counter command is the recommended method for reading a Counter. The Counter command returns a four byte CountValue field which is formatted as shown in Figure H-1. Optionally, the Counter command can also return a MAC for cryptographic authentication of the CountValue. The definition of the CountValue field is shown in Table H-2. See Section 7.5, Counter Command.

Figure H-1. CountValue Field

<table>
<thead>
<tr>
<th>Byte 0</th>
<th>Byte 1</th>
<th>Byte 2</th>
<th>Byte 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>LinCount</td>
<td>CountFlag</td>
<td>BinCount</td>
<td></td>
</tr>
</tbody>
</table>

The CountValue contains a Linear Counter Field (LinCount), a Binary Counter field (BinCount), and the CountFlag field. The CountFlag field indicates if the Counter value was read from the Counter A or Counter B EEPROM location. CountFlag also indicates if the 8 bit LinCount field is the Most Significant Byte (MSB) or Least Significant Byte (LSB) of the 16 bit LinCount field in EEPROM. If the LSB of LinCount has been returned, then the LinCount field value is 1 to 8; if the MSB of LinCount has been returned, then the LinCount field value is 9 to 16.

Table H-2. Definition of the CountValue field in the Response to the Counter Command

<table>
<thead>
<tr>
<th>Byte</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>LinCount</td>
<td>Contains the 8 bit linear Counter value identified in the CountFlag field.</td>
</tr>
</tbody>
</table>
| 1    | CountFlag             | 0x00 = LinCount contains the LSB of LinCountA. BinCount contains the BinCountA value.  
                        | 0x02 = LinCount contains the MSB of LinCountA. BinCount contains the BinCountA value.  
                        | 0x04 = LinCount contains the LSB of LinCountB. BinCount contains the BinCountB value.  
                        | 0x06 = LinCount contains the MSB of LinCountB. BinCount contains the BinCountB value.  
                        | All other values are reserved for future use.                                |
| 2    | BinCount (MSB)        | Contains the Most Significant Byte of the binary counter identified in the CountFlag field. |
| 3    | BinCount (LSB)        | Contains the Least Significant Byte of the binary counter identified in the CountFlag field. |

The equivalent decimal value of the Counter can be determined using the following equation:

\[
\text{CountValue} = (\text{BinCount}\times32) + (\text{CountFlag}/2)\times8 + \text{Lin2Bin(LinCount)}
\]

Here, Lin2Bin defines a function that converts a linear Counter value to corresponding binary value. 0xFFFF converts to zero; 0xFFFFE converts to one; and so on; up to 0x8000 which converts to 15.
H.3 Personalizing the Counters

The Counter registers are personalized with initial values prior to locking the Configuration Memory. The standard Serial EEPROM Write commands are used to write Configuration Memory (see Section 5.1.3, Read the STATUS Register). The Lock command is used to lock the Configuration Memory (see Section 7.18, Lock Command).

The initial value of the Counter registers can be determined using the following procedure:

Divide the Counter preset value by 32. The quotient is the value of BinCountA.

- If the remainder is less than 0.5, then:
  - BinCountB is one less than BinCountA
  - The remainder x 32 = the number of zeros in LinCountA
  - LinCountB = 0x0000
- If the remainder is equal or greater than 0.5, then:
  - BinCountB is equal to BinCountA
  - (The remainder – 0.5) x 32 = the number of zeros in LinCountB
  - LinCountA = 0x0000

Example 1: Preset to 8,159

- 8,159/32 = 254.96875
  - Binary Counter A = 254 or 0x00fe
  - Binary Counter B = 0x00fe (remainder is greater than 0.5)
  - Linear Counter B = 0x8000 (0.46875 x 32 = 15, Linear Counter B has 15 zeros)
  - Linear Counter A = 0x0000

Example 2: Preset to 1,000,000

- 1,000,000/32 = 31250.0
  - Binary Counter A = 31250 or 0x7a12
  - Binary Counter B = 0x7a11 (remainder is less than 0.5)
  - Linear Counter A = 0xFFFF (remainder is zero, Linear Counter A has no zeros)
  - Linear Counter B = 0x0000
Appendix I. Cryptographic Computations

ATAES132 implements all of its cryptographic commands using AES in CCM mode with a 128-bit key length per NIST SP800-38C. CCM mode provides both confidentiality and integrity checking with a single key. The integrity MAC includes both the encrypted data and additional authenticate-only data bytes. The particular information authenticated with each command is described within the command descriptions in Section 7, Command Definitions.

The device construction ensures that the Nonce will be unique for each MAC calculated.

I.1 MacCount

The one byte MacCount is stored in an internal register, and is used in the AES-CCM computations. Since MacCount changes, it speeds up computation by eliminating the need to generate a new random Nonce for every crypto computation. This register is incremented prior to performing each MAC calculation.

The MacCount Register is set to zero when the Nonce command is executed, and is subsequently incremented prior to any MAC computation being performed. Because of this, the value that will be used for calculating the first MAC of the first command after the Nonce command is MAC = 1.

There are two commands (Auth and KeyCreate) which can be configured to both verify an input MAC and calculate an output MAC. When either of these two commands is run in mutual-authentication mode, MacCount will be incremented twice.

The value of MacCount for a particular MAC calculation is always one greater than that used for the previous MAC calculation. After 255 MAC calculations, the device will invalidate the internal Nonce, and commands that require a valid Nonce will fail. At this point, a new Nonce command must be run to generate a new Nonce.

The MacCount is set to zero if any of the following events occurs:

- The Nonce command is executed.
- A MAC compare operation fails.
- MacCount reaches the maximum count.
- A Reset event occurs: Power-Up (see Appendix L.3.1, ChipState = Power-Up), Wake-Up from Sleep (see Appendix L.3.2, ChipState = Wake-Up from Sleep), the Reset command (see Section 7.22, Reset Command), or a Security Tamper is activated, causing the hardware to reset.

If a CRC error occurs on the incoming command packet, then MacCount will not be incremented. If the device receives any command that does not involve MAC computation, the MacCount will not be incremented.

If a cryptographic command is received that involves MAC computation, then the MacCount will be incremented regardless of whether or not there is a subsequent success or failure of the command. The MacCount will also be incremented regardless of whether or not the particular instance of the command uses the cryptographic engine.

If a command fails due to a MAC comparison failure, then the Nonce is invalidated and the MacCount Register is set to zero.

The current value of this register should be known by the system; however, it may also be read out of the device at any time using the INFO command (See Section 7.12, INFO Command).
I.2 MacFlag

To prevent spoofing of the MAC value, a flag byte is included in each MAC calculation. MacFlag provides information about the state of the device during the MAC calculation.

Table I-1. Definition of the MacFlag bits

<table>
<thead>
<tr>
<th>Bit</th>
<th>Name</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Random</td>
<td>1b = The Nonce command was run with the RNG enabled, and the Nonce is guaranteed to be unique. 0b = The Nonce value has been sent to the device by the system and may not be unique.</td>
</tr>
<tr>
<td>1</td>
<td>Input</td>
<td>1b = For MAC values that are sent to the device as inputs. 0b = For MAC values output by the ATAES132.</td>
</tr>
<tr>
<td>3 – 7</td>
<td>Zero</td>
<td>All bits must be 0b.</td>
</tr>
</tbody>
</table>

I.3 MAC Generation

The following example shows how the integrity MAC is calculated for an authentication operation requiring up to 14 bytes of authenticate-only data. This operation involves three passes through the AES crypto engine; all three using the same key. If there are more than 14 bytes of authenticate-only data, then another pass through the AES crypto engine is required.

There are two passes through the AES crypto engine in CBC mode to create the cleartext MAC. The inputs to the crypto engine for those blocks are labeled B0 and B1, and the outputs are B’0 and B’1, respectively.

- **B0** is composed of the following 128 bits:
  - 1 byte flag, a fixed value of b0111 1001.
  - 12 byte Nonce, as generated by the Nonce command.
  - 1 byte MacCount, 1 for first MAC generation.
  - 2 byte length field, always 0x00 00 for authentication only.

- **B1** is the XOR of B’0 with the following 128 bits:
  - 2 byte length field, size of authenticate-only data.
  - 14 byte data to be authenticated only.

- **B’1** is the cleartext MAC, which must be encrypted before being sent to the system.

There is one additional pass through the AES crypto engine in CTR mode to create the key block that is used to encrypt the MAC. The input to the crypto engine for this block is labeled A0 and the output is A’0. A’0 is the MAC sent to the system as the output parameter of the Auth command.

- **A0** is composed of the following 128 bits:
  - 1 byte flag – fixed value of b0000 0001.
  - 12 byte Nonce – as generated by ATAES132 during Nonce command.
  - 1 byte MacCount – one for first MAC generation.
  - 2 byte counter field – always 0x00 00 for A0.

- **A’0** is XOR’d with the cleartext MAC (B’1) and sent to the system.

Input integrity MACs for Auth, Counter, KeyCreate, and Lock are also calculated using this procedure. If the input MAC does not match A’0, then the command returns an AUTH error.
I.4 Data Encryption

The following example shows how the encrypted data and integrity MAC are calculated for a 128 bit data read from the device with up to 14 bytes of authenticate-only data. This operation involves five passes through the AES crypto engine; all five using the same key. If there are more than 14 bytes of authenticate-only data and/or more than 128 bits of data being read, then one, two, or three more passes through the AES crypto engine are required.

There are three passes through the AES crypto engine in CBC mode to create the cleartext MAC. The inputs to the crypto engine for those blocks are labeled B0, B1, and B2, and the outputs are B’0, B’1 and B’2, respectively.

- **B0** is composed of the following 128 bits:
  - 1 byte flag – fixed value of b0111 1001.
  - 12 byte Nonce – as generated by the Nonce command.
  - 1 byte MacCount – 1 for first MAC generation.
  - 2 byte length field – max 0x00 20 if 256 bits of encrypted data, min 0x00 01 for one byte.

- **B1** is the XOR of B’0 with the following 128 bits:
  - 2 byte length field – size of authenticate-only data.
  - 14 byte data to be authenticated only.

- **B2** is the XOR of B’1 with the following 128 bits:
  - 16 bytes cleartext data.

- **B’2** is the cleartext MAC – which must be encrypted before being sent to the system.

There are two passes through the AES crypto engine in CTR mode to create the key block that is used to encrypt the data and the MAC. The inputs to the crypto engine for those blocks are labeled A0 and A1, and the outputs are A’0 and A’1, respectively. A’0 and A’1 are the blocks sent to the system as the output parameters of the EncRead and Decrypt commands.

- **A0** is composed of the following 128 bits:
  - 1 byte flag – fixed value of b0000 0001.
  - 12 byte Nonce – as generated by the Nonce command.
  - 1 byte MacCount – one for first MAC generation.
  - 2 byte counter field – always 0x00 00 for A0.

- **A’0** is XOR’d with the cleartext MAC and sent to the system.

- **A1** is composed of the following 128 bits:
  - 1 byte flag – fixed value of b0000 0001.
  - 12 byte Nonce – as generated by ATAES132 during Nonce command.
  - 1 byte MacCount – one for first MAC generation.
  - 2 byte counter field – always 0x00 01 for A1.

- **A’1** is XOR’d with the cleartext data and sent to the system.

This sequence is also used for the Encrypt command, in addition to EncRead.
I.5 Data Decryption

The following example shows how the encrypted data and integrity MAC are calculated for a 128 bit data block write to the device with up to 14 bytes of authenticate-only data. This operation involves five passes through the AES crypto engine; all five using the same key. If there are more than 14 bytes of authenticate-only data and/or more than 128 bits of data being written, then one, two, or three more passes through the AES crypto engine are required.

There are two passes through the AES crypto engine in CTR mode to create the key block that is used to decrypt the data and the MAC. The inputs to the crypto engine for those blocks are labeled A0 and A1, and the outputs are A'0 and A'1, respectively. A'0 and A'1 are the blocks sent to the system as the output parameters of the EncRead and Decrypt commands.

- **A0** is composed of the following 128 bits:
  - 1 byte flag – fixed value of b0000 0001.
  - 12 byte Nonce – as generated by the Nonce command.
  - 1 byte MacCount – one for first MAC generation.
  - 2 byte counter field – always 0x00 00 for A0.

- **A'0** is XOR'd with the encrypted input MAC and stored in the internal SRAM as the MAC T.

- **A1** is composed of the following 128 bits:
  - 1 byte flag – fixed value of b0000 0001.
  - 12 byte Nonce – as generated by ATAES132 during Nonce command.
  - 1 byte MacCount – one for first MAC generation.
  - 2 byte counter field – always 0x00 01 for A1.

- **A'1** is XOR'd with the encrypted input data and stored in the internal SRAM as the message M.

There are three passes through the AES crypto engine in CBC mode to create the expected MAC value. The inputs to the crypto engine for those blocks are labeled B0, B1, and B2, and the outputs are B'0, B'1, and B'2, respectively.

- **B0** is composed of the following 128 bits:
  - 1 byte flag – fixed value of b0111 1001.
  - 12 byte Nonce – as generated by the Nonce command.
  - 1 byte MacCount – one for first MAC generation.
  - 2 byte length field – max 0x00 20 if 256 bits of encrypted data, min 0x00 01 for one byte.

- **B1** is the XOR of B'0 with the following 128 bits:
  - 2 byte length field – size of authenticate-only data.
  - 14 byte data to be authenticated only.

- **B2** is the XOR of B'1 with the following 128 bits:
  - 16 bytes of cleartext message M.

- **B'2** is the cleartext MAC. If this matches the stored T value, then the write to memory proceeds. If there is no match, the device returns an error flag and does not modify memory.

This sequence is also used for the Decrypt and KeyLoad commands, in addition to EncWrite.
I.6 Auth Command MAC

The MACs are calculated using the following 14 bytes in the default authenticate-only block:

2 bytes ManufacturingID
11 bytes FirstBlock field containing:
   1 byte Auth Opcode (0x03)
   1 byte Mode
   2 bytes Param1
   2 bytes Param2
   1 byte MacFlag
   4 bytes 0x00
1 byte Padding of value 0x00

If any of the optional authenticate fields are selected in the mode parameter, then a second authenticate-only block is included in the MAC calculations:

4 bytes Usage Counter value, or 0x00 if not selected.
8 bytes SerialNum[0:7], or 0x00 if not selected.
4 bytes SmallZone[0:3], or 0x00 if not selected.

I.7 AuthCheck Command – Auth MAC

The Auth command MAC is calculated using the following 14 bytes in the default authenticate-only block:

2 bytes ManufacturingID
11 bytes FirstBlock field containing:
   1 byte Auth Opcode (0x03)
   1 byte Mode
   2 bytes Param1
   2 bytes Param2
   1 byte MacFlag
   4 bytes 0x00
1 byte Padding of value 0x00

If any of the optional authenticate fields are selected in the mode parameter, then a second authenticate-only block is included in the MAC calculations:

16 bytes SecondBlock field containing:
   4 bytes Usage counter value, or 0x00 if not selected.
   8 bytes SerialNum[0:7], or 0x00 if not selected.
   4 bytes SmallZone[0:3], or 0x00 if not selected.
I.8 AuthCheck Command – Counter MAC

The Counter command MAC is calculated using the following 14 bytes in the default authenticate-only block:

- 2 bytes  ManufacturingID
- 11 bytes  FirstBlock field containing:
  - 1 byte  Counter Opcode (0x0A)
  - 1 byte  Mode
  - 2 bytes  Param1
  - 2 bytes  Param2
  - 1 byte  MacFlag
  - 4 bytes  CountValue, the output parameter
- 1 byte  Padding of value 0x00

If any of the optional authenticate fields are selected in the mode parameter, then a second authenticate-only block is included in the MAC calculations:

- 16 bytes  SecondBlock field containing:
  - 4 bytes  Usage counter value, or 0x00 if not selected.
  - 8 bytes  SerialNum[0:7], or 0x00 if not selected.
  - 4 bytes  SmallZone[0:3], or 0x00 if not selected.

I.9 AuthCompute Command – Auth MAC

The Auth command MAC is calculated using the following 14 bytes in the default authenticate-only block:

- 2 bytes  ManufacturingID
- 11 bytes  FirstBlock field containing:
  - 1 byte  Auth Opcode (0x03)
  - 1 byte  Mode
  - 2 bytes  Param1
  - 2 bytes  Param2
  - 1 byte  MacFlag
  - 4 bytes  0x00
- 1 byte  Padding of value 0x00

If any of the optional authenticate fields are selected in the mode parameter, then a second authenticate-only block is included in the MAC calculations:

- 16 bytes  SecondBlock field containing:
  - 4 bytes  Usage counter value, or 0x00 if not selected.
  - 8 bytes  SerialNum[0:7], or 0x00 if not selected.
  - 4 bytes  SmallZone[0:3], or 0x00 if not selected.
I.10 **AuthCompute Command – Counter MAC**

The Counter command MAC is calculated using the following 14 bytes in the default authenticate-only block:

2 bytes  ManufacturingID
11 bytes  FirstBlock field containing:
          1 byte  Counter Opcode (0x0A)
          1 byte  Mode
          2 bytes  Param1
          2 bytes  Param2
          1 byte  MacFlag
          4 bytes  0x00
1 byte  Padding of value 0x00.

If any of the optional authenticate fields are selected in the mode parameter, then a second authenticate-only block is included in the MAC calculations:

16 bytes  SecondBlock field containing:
          4 bytes  Usage counter value, or 0x00 if not selected.
          8 bytes  SerialNum[0:7], or 0x00 if not selected.
          4 bytes  SmallZone[0:3], or 0x00 if not selected.

I.11 **BlockRead Command**

The BlockRead command does not perform a cryptographic operation, and does not use or generate a MAC.

I.12 **Counter Command MAC**

The InMAC is calculated using the following 14 bytes in the default authenticate-only block:

2 bytes  ManufacturingID
1 byte  Counter Opcode (0x0A)
11 bytes  FirstBlock field containing:
          1 byte  Mode
          2 bytes  Param1
          2 bytes  Param2
          1 byte  MacFlag
          4 bytes  CountValue
          1 byte  0x00

If any of the optional authenticate fields are selected in the mode parameter, then a second authenticate-only block is included in the InMAC calculation:

4 bytes  Usage counter value for MAC generation key, or 0x00 if not selected.
8 bytes  SerialNum[0:7], or 0x00 if not selected.
4 bytes  SmallZone[0:3], or 0x00 if not selected.
The OutMAC is calculated using the following 14 bytes in the default authenticate-only block:

- 2 bytes  ManufacturingID
- 1 byte  Counter Opcode (0x0A)
- 1 byte  Mode
- 2 bytes  Param1
- 2 bytes  Param2
- 1 byte  MacFlag
- 4 bytes  CountValue, the output parameter
- 1 byte  0x00

If any of the optional authenticate fields are selected in the mode parameter, then a second authenticate-only block is included in the OutMAC calculation:

- 4 bytes  Usage counter value for MAC generation key, or 0x00 if not selected.
- 8 bytes  SerialNum[0:7], or 0x00 if not selected.
- 4 bytes  SmallZone[0:3], or 0x00 if not selected.

I.13 Crunch Command

The Crunch command does not perform a cryptographic operation, and does not use or generate a MAC.

I.14 DecRead Command

The MAC is calculated using the following 14 bytes in the default authenticate-only block:

- 2 bytes  ManufacturingID
- 1 byte  EncRead Opcode (0x04)
- 6 bytes  FirstBlock field containing:
  - 1 byte  Mode
  - 2 bytes  Param1
  - 2 bytes  Param2
  - 1 byte  MacFlag
- 5 bytes  0x00

If any of the optional authenticate fields are selected in the mode parameter, then a second authenticate-only block is included in the MAC calculation:

- 16 bytes  SecondBlock field containing:
  - 4 bytes  Usage counter value, or 0x00 if not selected.
  - 8 bytes  SerialNum[0:7], or 0x00 if not selected.
  - 4 bytes  SmallZone[0:3], or 0x00 if not selected.
I.15 Decrypt Command MAC

In Normal Decryption mode, the InMAC is calculated using the following 14 bytes in the default authenticate-only block:

- 2 bytes  ManufacturingID
- 1 byte  Decrypt Opcode (0x07)
- 1 byte  Mode
- 2 bytes  Param1
- 2 bytes  Param2
- 1 byte  MacFlag
- 5 bytes  0x00

If any of the optional authenticate fields are selected in the mode parameter, then a second authenticate-only block is included in the MAC calculation:

- 4 bytes  Usage counter value, or 0x00 if not selected or if KeyID is VolatileKey.
- 8 bytes  SerialNum[0:7], or 0x00 if not selected.
- 4 bytes  SmallZone[0:3], or 0x00 if not selected.

I.15.1 Client Decrypt MAC

In Client Decryption mode, the InMAC is calculated using the following 14 bytes in the default authenticate-only block:

- 2 bytes  ManufacturingID
- 1 byte  Encrypt Opcode (0x06)
- 1 byte  Mode
- 2 bytes  Upper byte = 0x00, lower byte = EKeyID
- 2 bytes  Upper byte = 0x00, lower byte = lower byte of Param2
- 1 byte  MacFlag = 0x01
- 5 bytes  0x00

If any of the optional authenticate fields are selected in the mode parameter, then a second authenticate-only block is included in the MAC calculation:

- 4 bytes  0x00 if Usage Counter value is not selected, or 0x00 if KeyID is VolatileKey.
- 8 bytes  SerialNum[0:7], or 0x00 if not selected.
- 4 bytes  SmallZone[0:3], or 0x00 if not selected.

The Device MacCount will be changed to the EMacCount value when a Decrypt command is received with the Client Decryption mode is selected. The EMacCount will be used when decrypting the data and the MacCount will be incremented by the Decrypt operation. (After processing the command, the device MacCount will equal EMacCount plus one.)
I.16 EncRead Command MAC

The OutMAC is calculated using the following 14 bytes in the default authenticate-only block:

| 2 bytes | ManufacturingID |
| 1 byte  | EncRead opcode (0x05) |
| 6 bytes | FirstBlock field containing |
| 1 byte  | Mode |
| 2 bytes | Param1 |
| 2 bytes | Param2 |
| 1 byte  | MacFlag |
| 5 bytes | 0x00 |

If any of the optional authenticate fields are selected in the mode parameter, then a second authenticate-only block is included in the MAC calculation:

| 4 bytes | Usage counter value, or 0x00 if not selected. |
| 8 bytes | SerialNum[0:7], or 0x00 if not selected. |
| 4 bytes | SmallZone[0:3], or 0x00 if not selected. |

I.17 EncRead Command Configuration Memory Signature MAC

The following example shows how the integrity MAC is calculated for a 512 byte (32 block) certification of the data from the Configuration Memory. This operation involves multiple passes through the AES crypto engine; all using the same key, KeyID 00. If the mode parameter indicates that there is an additional block of authenticate-only data, then another pass through the AES crypto engine is required.

There are 35 passes through the AES crypto engine in CBC mode to create the cleartext MAC. The inputs to the crypto engine for those blocks are labeled B0, B1, B2 …, and the outputs are B’0, B’1, B’2 …, respectively.

- **B0** is composed of the following 128 bits:
  - 1 byte flag – fixed value of b0111 1001
  - 12 byte Nonce – as generated by the Nonce command.
  - 1 byte MacCount – one for first MAC generation.
  - 2 byte length field – always 0x00 00.

- **B1** is the XOR of B’0 with the following 128 bits:
  - 2 byte length field – value of 528 or 544.
  - 14 byte ManufacturingID – Opcode, etc.

- **B2** is the XOR of B’1 with the following 128 bits:
  - 16 bytes counter+serial+small, if mode indicates; otherwise, this block does not exist.

- **B3** is the XOR of B’2 with the following 128 bits:
  - First 16 bytes of Config – in the clear.

- **B4** is the XOR of B’3 with the following 128 bits:
  - Second 16 bytes of Config – in the clear.
  - … and so on …

- **B’34** is the clear text MAC which must be encrypted before being sent to the system.
There is one pass through the AES crypto engine in CTR mode to encrypt the MAC.

- **A0** is composed of the following 128 bits:
  - 1 byte flag – a fixed value of b0000 0001.
  - 12 byte Nonce – as generated by the Nonce command.
  - 1 byte MacCount – one for first MAC generation.
  - 2 byte counter field – always 0x00 00.
- **A’0** is XOR’d with the clear text MAC and sent to the system.

### I.18 EncRead Command Key Memory Signature MAC

The following example shows how the integrity MAC is calculated for a 256 byte (16 block) certification of the data from the Key Memory. This operation involves multiple passes through the AES crypto engine; all using the same key, KeyID 00. If the mode parameter indicates that there is an additional block of authenticate-only data, then another pass through the AES crypto engine is required.

There are 19 passes through the AES crypto engine in CBC mode to create the cleartext MAC. The inputs to the crypto engine for those blocks are labeled B0, B1, B2 ..., and the outputs are B’0, B’1, B’2 ..., respectively.

- **B0** is composed of the following 128 bits:
  - 1 byte flag – fixed value of b0111 1001.
  - 12 byte Nonce – as generated by the Nonce command.
  - 1 byte MacCount – 1 for first MAC generation.
  - 2 byte length field – always 0x00 00.
- **B1** is the XOR of B’0 with the following 128 bits:
  - 2 byte length field – value of 272 or 288.
  - 14 byte ManufacturingID – Opcode, etc.
- **B2** is the XOR of B’1 with the following 128 bits:
  - 16 bytes counter+serial+small, if mode indicates; otherwise, this block does not exist.
- **B3** is the XOR of B’2 with the following 128 bits:
  - First 16 bytes of config – in the clear.
- **B4** is the XOR of B’3 with the following 128 bits:
  - Second 16 bytes of config – in the clear.
  - ... and so on ...
- **B’18** is the clear text MAC which must be encrypted before being sent to the system.

There is one pass through the AES crypto engine in CTR mode to encrypt the MAC.

- **A0** is composed of the following 128 bits:
  - 1 byte flag – fixed value of b0000 0001.
  - 12 byte Nonce – as generated by the Nonce command.
  - 1 byte MacCount – one for first MAC generation.
  - 2 byte counter field – always 0x00 00.
- **A’0** is XOR’d with the clear text MAC and sent to the system.
I.19 Encrypt Command MAC

The OutMAC is calculated using the following 14 bytes in the default authenticate-only block:

- 2 bytes: ManufacturingID
- 1 byte: Encrypt Opcode (0x06)
- 1 byte: Mode
- 2 bytes: Param1
- 2 bytes: Param2
- 1 byte: MacFlag
- 5 bytes: 0x00

If any of the optional authenticate fields are selected in the mode parameter, then a second authenticate-only block is included in the MAC calculation:

- 4 bytes: Usage counter value or 0x00 if not selected or if KeyID is VolatileKey.
- 8 bytes: SerialNum[0:7] or 0x00 if not selected.
- 4 bytes: SmallZone[0:3] or 0x00 if not selected.

I.20 EncWrite Command MAC

The InMAC is calculated using the following 14 bytes in the default authenticate-only block:

- 2 bytes: ManufacturingID
- 1 byte: EncWrite Opcode (0x05)
- 6 bytes: FirstBlock field containing:
  - 1 byte: Mode
  - 2 bytes: Param1
  - 2 bytes: Param2
  - 1 byte: MacFlag
- 5 bytes: 0x00

If any of the optional authenticate fields are selected in the mode parameter, then a second authenticate-only block is included in the MAC calculation:

- 4 bytes: Usage counter value or 0x00 if not selected.
- 8 bytes: SerialNum[0:7] or 0x00 if not selected.
- 4 bytes: SmallZone[0:3] or 0x00 if not selected.

I.21 INFO command

The INFO command does not perform a cryptographic operation, and does not use or generate a MAC.
I.22 KeyCreate Command MAC

The input and output MACs are both calculated using the parent key.

Both MACs are calculated using the following 14 bytes in the default authenticate-only block:

<table>
<thead>
<tr>
<th>Bytes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 bytes</td>
<td>ManufacturingID</td>
</tr>
<tr>
<td>1 byte</td>
<td>KeyCreate Opcode (0x08)</td>
</tr>
<tr>
<td>6 bytes</td>
<td>FirstBlock field containing:</td>
</tr>
<tr>
<td>1 byte</td>
<td>Mode</td>
</tr>
<tr>
<td>2 bytes</td>
<td>Param1</td>
</tr>
<tr>
<td>2 bytes</td>
<td>Param2</td>
</tr>
<tr>
<td>1 byte</td>
<td>MacFlag</td>
</tr>
<tr>
<td>5 bytes</td>
<td>0x00</td>
</tr>
</tbody>
</table>

If any of the optional authenticate fields are selected in the mode parameter, then a second authenticate-only block is included in the MAC calculations:

<table>
<thead>
<tr>
<th>Bytes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>4 bytes</td>
<td>Usage counter value or 0x00 if not selected.</td>
</tr>
<tr>
<td>8 bytes</td>
<td>SerialNum[0:7] or 0x00 if not selected.</td>
</tr>
<tr>
<td>4 bytes</td>
<td>SmallZone[0:3] or 0x00 if not selected.</td>
</tr>
</tbody>
</table>

I.23 KeyImport Command — KeyCreate MAC

The MAC is calculated using the following 14 bytes in the default authenticate-only block:

<table>
<thead>
<tr>
<th>Bytes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2 bytes</td>
<td>ManufacturingID</td>
</tr>
<tr>
<td>1 byte</td>
<td>KeyCreate Opcode (0x08)</td>
</tr>
<tr>
<td>6 bytes</td>
<td>FirstBlock field containing:</td>
</tr>
<tr>
<td>1 byte</td>
<td>Mode</td>
</tr>
<tr>
<td>2 bytes</td>
<td>Param1</td>
</tr>
<tr>
<td>2 bytes</td>
<td>Param2</td>
</tr>
<tr>
<td>1 byte</td>
<td>MacFlag</td>
</tr>
<tr>
<td>5 bytes</td>
<td>0x00</td>
</tr>
</tbody>
</table>

If any of the optional authenticate fields are selected in the mode parameter, then a second authenticate-only block is included in the MAC calculations:

<table>
<thead>
<tr>
<th>Bytes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>16 bytes</td>
<td>SecondBlock field containing:</td>
</tr>
<tr>
<td>4 bytes</td>
<td>Usage counter value or 0x00 if not selected.</td>
</tr>
<tr>
<td>8 bytes</td>
<td>SerialNum[0:7] or 0x00 if not selected.</td>
</tr>
<tr>
<td>4 bytes</td>
<td>SmallZone[0:3] or 0x00 if not selected.</td>
</tr>
</tbody>
</table>
I.24 KeyLoad Command MAC

The InMAC is calculated using the following 14 bytes in the default authenticate-only block:

2 bytes ManufacturingID
1 byte KeyLoad opcode (0x09)
6 bytes FirstBlock field containing:
   1 byte Mode
   2 bytes Param1
   2 bytes Param2
   1 byte MacFlag

5 bytes 0x00

If any of the optional authenticate fields are selected in the mode parameter, then a second authenticate-only block is included in the MAC calculation:

4 bytes Usage counter value, or 0x00 if not selected.
8 bytes SerialNum[0:7], or 0x00 if not selected.
4 bytes SmallZone[0:3], or 0x00 if not selected.

I.25 KeyTransfer Command

The KeyTransfer command does not perform a cryptographic operation and does not use or generate a MAC.

I.26 Legacy Command

The Legacy command executes a single block of the AES engine with no input or output formatting. This is known as ECB mode and can be used to perform various AES encryption and/or authentication operations. This command does not use the Nonce Register value in the computation since the entire 16 byte AES engine input value comes from the input packet.

I.27 Lock Command MAC

If required, due to the value of the mode parameter and ZoneConfig[Zone].WriteMode, the MAC is calculated using the following 14 bytes in the default authenticate-only block:

2 bytes ManufacturingID
1 byte Lock Opcode (0x0D)
1 byte Mode
2 bytes Param1
2 bytes Param2
1 byte MacFlag
5 bytes 0x00

If any of the optional authenticate fields are selected in the mode parameter, then a second authenticate-only block is included in the MAC calculation:

4 bytes Usage counter value or 0x00 if not selected.
8 bytes SerialNum[0:7] or 0x00 if not selected.
4 bytes SmallZone[0:3] or 0x00 if not selected.

The AES key used for the MAC calculation is that specified in ZoneConfig[Zone].WriteID.
I.28 **Nonce Command**

If the Random Nonce option is selected, then the internal Random Nonce is generated using the following function:

Block A is:
- 1 byte Nonce Opcode (0x01)
- 1 byte Mode
- 2 bytes 0x00
- 12 bytes Input Seed

Block B is:
- 2 bytes ManufacturingID
- 2 bytes 0x00
- 12 bytes Internally generated random number

AES is executed in ECB mode with an input value of Block A and a key of Block B. The output of the AES crypto engine is XOR’d with Block A, and the first 12 bytes of the result are stored in the internal Nonce Register.

If the LockConfig Register is unlocked (0x55), then the random number generator is latched in test mode, and the Nonce command will generate nonrandom values. If the LockConfig Register is locked (0x00), then the random number generator generates random numbers and the Nonce command functions normally.

I.29 **NonceCompute Command**

The random Nonce is generated using the following function:

Block A is:
- 1 byte Nonce opcode (0x01)
- 1 byte Mode
- 2 bytes 0x00
- 12 bytes Nonce Register

Block B is:
- 2 bytes ManufacturingID
- 2 bytes 0x00
- 12 bytes Random Seed

AES is executed in ECB mode with an input value of Block A and a key of Block B. The output of the AES crypto engine is XOR’d with Block A, and the first 12 bytes of the result are stored in the internal Nonce Register.

I.30 **Random Command**

Generates a random number using the internal high-quality random number generator and the random number generation procedure recommended by NIST in SP800-90 (see Appendix A, Standards and Reference Documents).

I.31 **Reset Command**

The Reset command does not perform a cryptographic operation and does not use or generate a MAC.

I.32 **Sleep Command**

The Sleep command does not perform a cryptographic operation and does not use or generate a MAC.
I.33 **WriteCompute Command**

The MAC is calculated using the following 14 bytes in the default authenticate-only block:

- 2 bytes  ManufacturingID
- 1 byte  EncWrite Opcode (0x05)
- 6 bytes  FirstBlock field containing:
  - 1 byte  Mode
  - 2 bytes  Param1
  - 2 bytes  Param2
  - 1 byte  MacFlag
- 5 bytes  0x00

If any of the optional authenticate fields are selected in the mode parameter, then a second authenticate-only block is included in the MAC calculation:

- 16 bytes  SecondBlock field containing:
  - 4 bytes  Usage counter value or 0x00 if not selected.
  - 8 bytes  SerialNum[0:7] or 0x00 if not selected.
  - 4 bytes  SmallZone[0:3] or 0x00 if not selected.
Appendix J. \textit{I^2}C Interface

The ATAES132 2-Wire Serial Interface is designed to interface directly to microcontrollers with \textit{I^2}C interface ports. The serial interface and cleartext Read/Write operations operate similar to those of the Atmel \textit{I^2}C Serial EEPROM.

The Host sends ATAES132 extended commands to the device by writing the command packet to the Command Memory Buffer at address 0xFE00. The ATAES132 processes the command packet and places the response in the Response Memory Buffer. The Host retrieves the response by reading the response packet from address 0xFE00.

See Appendix G.2, STATUS Register Behavior in the \textit{I^2}C Interface Mode for additional information regarding the ATAES132 behavior in \textit{I^2}C interface mode. See Appendix J.6, \textit{I^2}C Compatibility for \textit{I^2}C compatibility information.

J.1 \textit{I^2}C Serial Interface Description

When ATAES132 is configured in \textit{I^2}C serial communication mode, the serial interface operates as an \textit{I^2}C compatible standard-mode \textit{I^2}C slave device as described in this appendix. \textit{I^2}C is a synchronous serial interface protocol that is a \textit{de facto} industry standard and is not formally documented or controlled. Multiple \textit{I^2}C devices can share the data bus; however, each \textit{I^2}C slave must have a unique \textit{I^2}C Device Address to prevent bus contention. SCK clock frequencies up to 1MHz are supported by the ATAES132.

The serial interface communication mode is selected by programming the I2CAddr Register in the Configuration Memory as described in Appendix E.2.15, I2CAddr Register. The \textit{I^2}C Device Address is also located in the I2CAddr Register. The ATAES132 will only respond to \textit{I^2}C instructions that have a matching \textit{I^2}C Device Address.

J.1.1 \textit{I^2}C Master

The \textit{I^2}C master device generates the serial clock and sends instructions to the \textit{I^2}C slave devices. In this specification, the \textit{I^2}C master is usually referred to as the Host or the Host microcontroller.

J.1.2 \textit{I^2}C Slave

\textit{I^2}C slave devices receive the serial clock as an input and receive instructions from the \textit{I^2}C master. \textit{I^2}C slaves can never generate traffic on the \textit{I^2}C interface. Slaves can only respond to instructions provided by the \textit{I^2}C master. The ATAES132 always operates as a slave. In this specification, the slave is usually referred to as the Client or the device.

J.1.3 \textit{I^2}C Device Address

Each ATAES132 has a seven bit \textit{I^2}C Device Address (stored in the I2CAddr Register, as described in Appendix E.2.15) which is used by the Host to direct commands to a specific device on the \textit{I^2}C interface. \textit{I^2}C devices will only respond to instructions with a matching \textit{I^2}C Device Address. When the ATAES132 is in the Standby state or Sleep state, a matching \textit{I^2}C Device Address will cause the device to wake-up (see Appendix L, Power Management for power management specifications).

The LSB of the \textit{I^2}C Device Address byte is the Read/Write operation select bit. A Read operation is initiated if the R/W bit is high, and a Write operation is initiated if the R/W bit is low.

J.1.4 Relationship of Clock to Data

Data on the SDA pin may change only during SCK low time periods. Data changes during SCK high periods indicate an \textit{I^2}C Start or Stop condition. The SDA pin is pulled high by an external resistor when no devices are driving the \textit{I^2}C data bus. The timing requirements for the clock and data signals are illustrated in Appendix 0.
Timing Diagrams.

**J.1.5 I^2C Start Condition**

A high-to-low transition of SDA with SCK high is an I^2C Start condition. An I^2C Start condition must precede the I^2C Device Address for any instruction. I^2C Start conditions are generated only when the Host is driving the bus; slaves are not allowed to generate an I^2C Start condition.

The slave will reset its serial interface immediately when an I^2C Start condition is received. An I^2C Start condition cannot be followed immediately with an I^2C Stop condition. Figure J-1 illustrates an I^2C Start condition.

**J.1.6 I^2C Stop Condition**

A low-to-high transition of SDA with SCK high is an I^2C Stop condition. I^2C Stop conditions are only generated when the Host is driving the bus; slaves are not allowed to generate an I^2C Stop condition. Figure J-1 illustrates an I^2C Stop condition.

Figure J-1. I^2C Start Condition and I^2C Stop Condition Definitions

![I^2C Start and Stop Condition Diagram](image)

**J.1.7 I^2C ACK**

All addresses and data words are serially transmitted to and from ATAES132 in 8-bit words. The receiving I^2C device sends a zero (ACK) during the ninth clock cycle to acknowledge receipt of each byte.

An I^2C Host can use acknowledge polling to monitor the progress of an EEPROM Write and to determine if the slave is ready to accept a new instruction. See Appendix J.3.7, Acknowledge Polling for a discussion of ACK polling.

**J.1.8 I^2C NAK**

When the receiving I^2C device fails to send a zero during the ninth clock cycle to acknowledge that it has received a byte, then SDA remains high due to the external pull-up resistor. This generates a NO ACK (NAK) signal to the device sending the byte.

**J.1.9 Data Format**

All instructions and data on the I^2C bus must be formatted as 8-bit bytes, followed by a ninth bit (ACK or NAK) generated by the receiving device. The MSB is the first bit of each byte transmitted and received.
### J.2 Pin Descriptions

When the ATAES132 is configured in the I²C interface communication mode, the package pins are assigned the functionality described in this section.

Note: The pin numbers listed here are the SOIC and UDFN package pin numbers.

#### Table 9-8. I²C Communication Mode Pin Descriptions

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CS</td>
<td>SPI Chip Select Bar Input pin. In the I²C Communication mode, this pin is not used, and should be tied to VCC or VSS. The state of this pin does not affect the functionality or Active state power consumption of the ATAES132 when I²C Communication mode is selected.</td>
</tr>
<tr>
<td>2</td>
<td>SO</td>
<td>Serial Data Out pin. In the I²C Communication mode, this pin is not used in the default configuration. It is always in the high-impedance state. In this configuration, the pin can be tied to VCC or VSS. The state of this pin does not affect the functionality or Active state power consumption of the ATAES132 when I²C Communication mode is selected. If Auth signaling is enabled, then the SO pin functions as the AuthO signal output. In this configuration, the AuthO signal is high after a specified key is authenticated. The AuthO output is in the high-impedance state when the device has not authenticated. (See Appendix J.5, I²C Auth Signaling).</td>
</tr>
<tr>
<td>3</td>
<td>NC</td>
<td>No Connect pin. This package pin is not used, and can be left open by the user. The state of this pin does not affect the functionality or power consumption of the ATAES132.</td>
</tr>
<tr>
<td>4</td>
<td>VSS</td>
<td>Ground.</td>
</tr>
<tr>
<td>5</td>
<td>SI/SDA</td>
<td>Bidirectional Serial Data I/O pin. In the I²C communication mode, this pin functions as the Serial Data I/O (SDA). This pin is an open-drain buffer and may be wire-ORed with any number of other open-drain or open-collector devices. The SDA pin must be pulled high with an external resistor for the I²C bus to operate correctly. Data on the SDA pin may change only during the SCK low time periods. Data changes during SCK high periods indicate an I²C Start or Stop condition. Data transfer on the SDA line is half-duplex, as described by the I²C command definitions in Appendix J.3, I²C Instruction Set; the Host and Client cannot simultaneously drive the SDA line.</td>
</tr>
<tr>
<td>6</td>
<td>SCK</td>
<td>Serial Clock Input pin. In the I²C Communication mode, this pin is used as the Serial Interface Clock (SCK). The SCK input is used to transfer data into the ATAES132 on the rising edge of clock and to transfer data out on the falling edge of clock. The ATAES132 never drives SCK because it is a standard-mode I²C slave device. Slave device clock stretching is not supported. The SCK line is high when the bus is idle. If the I²C master uses a normal totem pole output to drive SCK, then no pull-up resistor is required on the SCK line. If the I²C master uses an open-drain or open-collector output to drive SCK, then an external pull-up resistor is required.</td>
</tr>
<tr>
<td>7</td>
<td>NC</td>
<td>No Connect pin. This package pin is not used, and can be left open by the user. The state of this pin does not affect the functionality or power consumption of the ATAES132.</td>
</tr>
<tr>
<td>8</td>
<td>VCC</td>
<td>Supply Voltage. Power cannot be removed from the ATAES132 when the I²C interface is active. The device may be permanently damaged if the requirements in Section 9.1, Absolute Maximum Ratings* and Section 9.3, DC Characteristics are exceeded..</td>
</tr>
</tbody>
</table>
J.3 \( i^2C \) Instruction Set

The ATAES132 utilizes the Atmel AT24C32C Serial EEPROM instruction set. The ATAES132 \( i^2C \) instruction set is shown in Table J-1.

Table J-1. ATAES132 \( i^2C \) Instruction Set

<table>
<thead>
<tr>
<th>Instruction Name</th>
<th>Operation</th>
<th>Details</th>
</tr>
</thead>
<tbody>
<tr>
<td>BWRITE</td>
<td>Byte Write</td>
<td>Writes one byte to memory.</td>
</tr>
<tr>
<td>PWRITE</td>
<td>Page Write</td>
<td>Writes 2 to 32 bytes to memory.</td>
</tr>
<tr>
<td>READ</td>
<td>Read</td>
<td>Reads data from memory starting at the current address.</td>
</tr>
<tr>
<td>RREAD</td>
<td>Random Read</td>
<td>Reads data from memory starting at the specified address.</td>
</tr>
<tr>
<td>SREAD</td>
<td>Sequential Read</td>
<td>Reads additional data from memory.</td>
</tr>
<tr>
<td>SRESET</td>
<td>Software Reset</td>
<td>Resets the internal memory address counter to 0000h.</td>
</tr>
</tbody>
</table>

If ATAES132 receives an invalid or undefined instruction code, it will be ignored and the associated data bytes will be discarded. When any error occurs, the ERR bit of the STATUS Register is set to 1b to indicate an error. The Host can read the error code from the Response Memory Buffer at address 0xFE00 using the READ command.

J.3.1 Byte Write (BWRITE)

A Byte Write operation requires two 8-bit data word addresses following the \( i^2C \) Device Address byte. Upon receipt of the Start condition and device address, the ATAES132 will respond with \( i^2C \) ACK and then clock in the two address bytes (ACKing each byte). The ATAES132 will ACK the receipt of the data byte from the Host. The Host microcontroller must terminate the write sequence with a Stop condition to initiate the Write operation.

At this time, the EEPROM enters an internally-timed write cycle to the nonvolatile memory. All inputs are disabled during this write cycle, and the EEPROM will NAK the device address until the write is complete.

If the Host transmits an invalid address, the EEPROM will NAK the second address byte and any data bytes.

When any error occurs, the RRDY and ERR bits of the STATUS Register are set to 1b to indicate an error. The Host can read the error code from the Response Memory Buffer (address 0xFE00) using the RREAD command. If the command is processed without error, the ERR bit is set to 0b. Reading the Response Memory Buffer does not reset the error code or the STATUS Register.

Figure J-2. Byte Write
### J.3.2 Page Write (PWRITE)

The ATAES132 is capable of 32-byte Page Writes. A Page Write is initiated in the same way as a Byte Write, but the Host microcontroller does not send a Stop condition after the first data byte is clocked in. Instead, after the device ACKs receipt of the first data byte, the Host microcontroller can transmit up to 31 more data bytes (each byte will be ACKed by the ATAES132). The EEPROM will respond with an I²C ACK after each data byte is received. The Host must terminate the Page Write sequence with a Stop condition. The data address is internally incremented following the receipt of each data byte.

If more than 32 bytes of data are transmitted or the page boundary is crossed, then no data will be written.

If the Host transmits an invalid word address, the EEPROM will NAK the second address byte and all data bytes. When any error occurs, the RRDY and EERR bits of the STATUS Register are set to 1b to indicate an error. The Host can read the error code from the Response Memory Buffer (address 0xFE00) using the RREAD command. If the command is processed without error, the EERR bit is set to 0b. Reading the Response Memory Buffer does not reset the error code or the STATUS Register.

#### Figure J-3. Page Write

### J.3.3 Current Address Read (READ)

The internal data byte address Counter maintains the last address accessed during the last Read or Write operation incremented by one. This address stays valid between operations as long as the device power is maintained.

To perform a Current Address Read, the Host sends the device address with the Read/Write Select bit set to one (READ), and this byte is ACKed by the EEPROM. Then, the Host clocks out the data byte located at the current address. After the byte is received, the Host responds with an I²C NAK and a following Stop condition to terminate the Read operation.

When any error occurs, the EERR bit of the STATUS Register is set to 1b to indicate an error. If the command is processed without error, the EERR bit is set to 0b.

#### Figure J-4. Current Address Read of One Data Byte
J.3.4 Random Read (RREAD)

A Random Read requires a dummy Byte Write sequence to load in the data byte address. Once the device address and data byte address are clocked in and acknowledged by the ATAES132, the Host microcontroller must generate another Start condition. The microcontroller then initiates a Current Address Read by sending the device address with the Read/Write Select bit high (READ). The ATAES132 I^2^C ACKs the device address, and serially clocks out the data byte. After the byte is received, the Host responds with an I^2^C NAK and a following Stop condition to terminate the Read operation.

If the Host transmits an invalid word address, the EEPROM will NAK the second address byte.

When any error occurs, the EERR bit of the STATUS Register is set to 1b to indicate an error. If the command is processed without error, the EERR bit is set to 0b.

Figure J-5. Random Read

J.3.5 Sequential Read (SREAD)

Sequential Reads are initiated by either a Current Address Read or a Random Read. After the Host microcontroller receives a data byte, it responds with an I^2^C ACK. As long as the EEPROM receives an acknowledge, it will continue to increment the data byte address and serially clock out sequential data bytes. The Sequential Read operation is terminated when the microcontroller responds with an I^2^C NAK and a following Stop condition.

When any error occurs, the EERR bit of the STATUS Register is set to 1b to indicate an error. If the command is processed without error, the EERR bit is set to 0b.

Note: If an I^2^C Read begins at an authorized address and continues into protected memory, the EERR bit will be set to 1b. Attempting to read protected memory will result in 0xFF data returned to the Host for each protected byte address.

Figure J-6. Sequential Read
J.3.6 Software Reset (SRESET)

After an interruption in protocol, powerloss, or system reset, the ATAES132 in I²C interface mode can be protocol reset by following these steps:

- Send a Start condition,
- Clock nine cycles,
- Send another Start condition followed by Stop condition, as shown below.

The device is ready for the next communication after these steps have been completed. The internal data address is also reset to 0000h by this procedure.

Figure J-7. Software Reset

The ATAES132 requires that the clock be pulled low between the Start condition and the Stop condition at the end of the sequence, as illustrated in Figure J-7. It will not reset if this clock transition is omitted. See Appendix J.4, I²C Interface Synchronization Procedure for detailed I²C interface resynchronization instructions.

J.3.7 Acknowledge Polling

The Host can initiate Acknowledge (ACK) Polling immediately after a Write command or the ATAES132 extended Crypto command is transmitted. Acknowledge polling involves sending a Start condition followed by the I²C Device Address. The Read/Write bit of the I²C Device Address is representative of the operation desired by the Host.

During an EEPROM Write operation, the ATAES132 will NAK the I²C Device Address, indicating the device is busy. When the internal write cycle has completed, the ATAES132 will ACK the I²C Device Address, allowing the Read or Write sequence to continue. The ATAES132 also NAKs during the processing of Crypto commands, and so Acknowledge Polling can also be used to determine when processing of the ATAES132 extended commands is complete.
J.4 \textit{I}^2\textit{C} Interface Synchronization Procedure

If the Host and Client \textit{I}^2\textit{C} interfaces lose synchronization for any reason, the Host should send clocks until SDA goes high followed by the SRESET command to reset the ATAES132 interface. See Appendix J.3.6, Software Reset (SRESET).

J.5 \textit{I}^2\textit{C} Auth Signaling

The Auth signaling option allows an Authentication Signal (AuthO) to be output by ATAES132. Auth signaling is available only in the \textit{I}^2\textit{C} interface mode in standard plastic packages.

The Auth signaling option is controlled by two bits in the KeyConfig Registers: the KeyConfig[KeyID].AuthOut bit and the KeyConfig[KeyID].AuthOutHold bit (see Table J-2). By default, the KeyConfig[KeyID].AuthOut bit is 0b for all keys disabling the Auth signaling option.

Table J-2. Auth Signaling KeyConfig Bit Functions

<table>
<thead>
<tr>
<th>AuthOut Bit</th>
<th>AuthOutHold Bit</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>1b</td>
<td>X</td>
<td>First successful Auth command forces AuthO high. Additional Auth commands do not change AuthO and the AuthO output remains latched high.</td>
</tr>
<tr>
<td>0b</td>
<td>X</td>
<td>Successful or unsuccessful Auth commands cause no AuthO change.</td>
</tr>
<tr>
<td>X</td>
<td>1b</td>
<td>Authentication Reset does not change the AuthO output state.</td>
</tr>
<tr>
<td>X</td>
<td>0b</td>
<td>Authentication Reset forces AuthO to the high-impedance state.</td>
</tr>
</tbody>
</table>

If the KeyConfig[AKeyID].AuthOut bit is 1b for the Authentication Key (AKeyID), then Auth signaling is enabled for that key and the AuthO signal is output on the SO pin. AuthO is latched high after a successful Inbound-Only Authentication or Mutual Authentication using the Auth command (see Section 7.1, Auth Command). AuthO will remain high until the device is powered off, unless an Authentication Reset is received.

If the KeyConfig[AKeyID].AuthOutHold bit is 0b for the key (AKeyID) used to execute an Authentication Reset, then the AuthO signal latch will be latched in the high-impedance state when the command is received (with a correct Checksum). If the KeyConfig[AKeyID].AuthOutHold bit is 1b, then AuthO will be unchanged by execution of an Authentication Reset sequence.
An Authentication Reset is an Auth command with Mode bits 0 and 1 set to 00b. Knowledge of the key value is not required to execute an Authentication Reset (see Section 7.1). The ATAES132 does not memorize the KeyID used to activate Auth signaling. Each Auth command is processed using the KeyConfig[AKeyID] bits of the AKeyID in the command packet.

Auth signaling is not a security feature. The AuthO signal does not reflect the real-time state of the AuthComplete status flag. The Reset command, the Sleep command, and the Tamper detectors will not change the state of AuthO. The state of the AuthO latch is determined only by success or failure of the Auth command and the configuration of the KeyConfig bits. The INFO command should be used to determine the authentication status of the device (see Section 7.12, INFO Command).

The KeyConfig[AKeyID].AuthOut bit and the KeyConfig[AKeyID].AuthOutHold bit are ignored when the ATAES132 is configured in SPI Interface mode.

### J.5.1 Using the AuthO Output

When Auth signaling is enabled, the AuthO signal output is either a Logic high or in the high-impedance state. AuthO can be used to drive an LED or as a control signal to other circuitry. When AuthO is used as a control signal, a pull-down resistor should be used to transform the high-impedance state into a logic low.

### J.6 I²C Compatibility

ATAES132 is design to operate on a bus with other I²C-compatible devices. ATAES132 is a standard-mode Client device capable of operating at clock speeds up to 1MHz (with bus timing scaled accordingly). The ATAES132 is not a Fast-Mode or High-Speed mode device.

This section lists the I²C options or features that are not supported by the ATAES132. Any feature that differs from the I²C specification is also listed.

- ATAES132 does not perform Client clock stretching.
- ATAES132 will not respond to an I²C general call command.
- ATAES132 may be damaged if the clock or data signal levels are above VCC. The power supply to the ATAES132 cannot be switched off while the bus is active. All of the voltage limits in Section 9.1, Absolute Maximum Ratings* must be respected.
- ATAES132 inputs include Schmitt Triggers and spike suppression; however, the outputs do not include falling edge slope control.
- On I²C devices, a Start condition followed immediately by a Stop condition is never permitted. On the ATAES132, this sequence is permitted only as part of the SRESET command sequence (see Appendix J.3.6, Software Reset (SRESET)).
J.7 Timing Diagrams

Figure J-9. \(^{\text{i}2\text{C}}\) Synchronous Data Timing (see Section 9.4.1 for PC Timing Specifications)

![SCL and SDA IN and SDA OUT Timing Diagram](attachment:image1.png)

Figure J-10. \(^{\text{i}2\text{C}}\) Write Cycle Timing

![SCL and SDA Timing Diagram](attachment:image2.png)
Appendix K. SPI Interface

The ATAES132 Serial Peripheral Interface (SPI) is designed to interface directly to microcontrollers using SPI Mode 0 or Mode 3. I/O and Cleartext Read/Write operations operate similarly to those of the Atmel SPI Serial EEPROM.

The Host sends ATAES132 commands to the device by writing the command packet to the Command Memory Buffer at address 0xFE00. The ATAES132 processes the command packet and places the response in the Response Memory Buffer. The Host retrieves the response by reading the response packet from address 0xFE00.

See Appendix G.3, STATUS Register Behavior in the SPI Interface Mode for additional information on the ATAES132 behavior in SPI interface mode.

K.1 SPI Serial Interface Description

When ATAES132 is configured in the SPI communication mode, the serial interface operates as a Mode 0 and Mode 3 slave device as described in this appendix. Serial Peripheral Interface (SPI) is a synchronous serial interface protocol that is a de facto industry standard and is not formally documented or controlled. Multiple SPI devices can share the data bus; however, each SPI slave must have a separate CS control line to prevent bus contention.

The serial interface communication mode is selected by programming the I2CAddr Register in the Configuration Memory as described in Section E.2.15.

K.1.1 SPI Master

The SPI bus master device generates the serial clock and sends instructions to the SPI slave devices. In this specification, the bus master is usually referred to as the Host or the Host microcontroller.

K.1.2 SPI Slave

SPI slave devices receive the serial clock as an input and receive instructions from the bus master. SPI slaves can never generate traffic on the SPI bus, and slaves can only respond to instructions provided by the bus master. The ATAES132 always operates as a slave. In this specification the slave is usually referred to as the Client.

K.1.3 Relationship of Clock to Data

The ATAES132 supports two of the four standard SPI interface modes; Mode 0 and Mode 3.

- In Mode 0:
  - The default state of SCK is low.
  - The data is clocked in (SI) on the rising edge of the clock.
  - Data out (SO) changes on the falling edge of the clock.

- In Mode 3:
  - The default state of SCK is high.
  - The data is clocked in (SI) on the rising edge of the clock.
  - Data out (SO) changes on the falling edge of the clock.
**K.1.4 SPI Instruction Code**

Each SPI command begins with the SPI master bringing the \( \overline{CS} \) input low to select the device followed by transmission of an eight bit SPI instruction code to the SI input of the SPI slave. Following the instruction code, additional bytes may be clocked into SI or out of SO as required by the SPI command (see Appendix K.3, SPI Instruction Set). When the exchange of data bytes related to the SPI instruction code is complete, the \( \overline{CS} \) input is brought high to deactivate the SPI slave interface.

If an invalid instruction code is received, then the ATAES132 will ignore any data received on the Data Input pin (SI), and the Data Output pin (SO) will remain in a high-impedance state.

**K.1.5 Data Format**

All instructions and data on the SPI bus must be formatted as eight bit bytes. The Most-Significant bit (MSB) is the first bit of each byte transmitted and received.

**K.2 SPI Communication Mode Pin Descriptions**

When ATAES132 is configured in SPI communication mode, the package pins are assigned the functionality described in this section.

**Table K-1. Pin Descriptions**

<table>
<thead>
<tr>
<th>Pin</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>( \overline{CS} )</td>
<td><strong>SPI Chip Select Bar Input pin.</strong> In SPI communication mode, this pin functions as the slave select input. The ATAES132 is selected when the ( \overline{CS} ) pin is low, allowing instructions and data to be accepted on the Serial Data Input pin (SI), and allowing data to be transmitted on the Serial Data Output pin (SO). When the device is not selected, data will not be accepted via the SI pin, and the Serial Output pin (SO) will remain in a high-impedance state. When the ATAES132 is in the Standby state or Sleep state, a high-to-low transition on the ( \overline{CS} ) pin will cause the device to wake-up (see Appendix L, Power Management for power management specifications). It is recommended that the ( \overline{CS} ) pin be connected to VCC with a pull-up resistor so that the ( \overline{CS} ) pin follows VCC during power-up and power-down.</td>
</tr>
<tr>
<td>2</td>
<td>SO</td>
<td><strong>Serial Data Out pin.</strong> In the SPI communication mode, this pin functions as the Serial Data output. When the ( \overline{CS} ) pin is high, the SO pin will always be in a high-impedance state because the SPI interface is disabled.</td>
</tr>
<tr>
<td>3</td>
<td>NC</td>
<td><strong>No Connect pin.</strong> This package pin is not used, and can be left open by the user. The state of this pin does not affect the functionality or power consumption of the ATAES132.</td>
</tr>
<tr>
<td>4</td>
<td>VSS</td>
<td><strong>Ground.</strong></td>
</tr>
<tr>
<td>5</td>
<td>SI/SDA</td>
<td><strong>Serial Data Input pin.</strong> In the SPI communication mode, this pin functions as the serial data input. When the ( \overline{CS} ) pin is high, the SI pin will not accepted data because the SPI interface is disabled.</td>
</tr>
<tr>
<td>6</td>
<td>SCK</td>
<td><strong>Serial Clock Input pin.</strong> In the SPI communication mode, this pin is used as the serial interface clock. All data on the SI and SO pins is synchronized by SCK, as described in Appendix K.1.3, Relationship of Clock to Data.</td>
</tr>
<tr>
<td>7</td>
<td>NC</td>
<td><strong>No Connect pin.</strong> This package pin is not used, and can be left open by the user. The state of this pin does not affect the functionality or power consumption of the ATAES132.</td>
</tr>
<tr>
<td>8</td>
<td>VCC</td>
<td><strong>Supply Voltage.</strong> Power cannot be removed from the ATAES132 when the SPI bus is active. The device may be permanently damaged if the requirements in Section 9.1, Absolute Maximum Ratings* and Section 9.3, DC Characteristics are exceeded.</td>
</tr>
</tbody>
</table>
K.3 SPI Instruction Set

ATAES132 utilizes an 8-bit SPI instruction register. The SPI instruction set is listed in Table K-2.

Table K-2. ATAES132 SPI Instruction Set

<table>
<thead>
<tr>
<th>Instruction Name</th>
<th>Instruction Code</th>
<th>Operation</th>
</tr>
</thead>
<tbody>
<tr>
<td>WRITE</td>
<td>0000 0010 b</td>
<td>Write data to memory.</td>
</tr>
<tr>
<td>READ</td>
<td>0000 0011 b</td>
<td>Read data from memory.</td>
</tr>
<tr>
<td>WRDI</td>
<td>0000 0100 b</td>
<td>Reset Write Enable Register</td>
</tr>
<tr>
<td>RDSR</td>
<td>0000 0101 b</td>
<td>Read Status Register</td>
</tr>
<tr>
<td>WREN</td>
<td>0000 0110 b</td>
<td>Set WriteEnable Latch</td>
</tr>
</tbody>
</table>

If the ATAES132 receives an invalid instruction code or an invalid memory address, then no response will be sent; the SO output will remain in the high-impedance state. When any error occurs, the EERR bit of the STATUS Register is set to 1b to indicate an error. The Host can read the error code from the Response Memory Buffer at address 0xFE00 using the READ command. Reading the Response Memory Buffer does not reset the error code or change the STATUS.

K.3.1 Write Enable Command (WREN)

The device will power-up in the Write Disable state when VCC is applied. All EEPROM Write instructions must therefore be preceded by a Write Enable instruction. It is not necessary to send the Write Enable instruction prior to sending command packets to the Command Memory Buffer.

Figure K-1. SPI Write Enable (WREN) Timing
K.3.2 Write Disable Command (WRDI)

The Write Enable flag can be disabled by sending the Write Disable instruction.

Figure K-2. SPI Write Disable (WRDI) Timing

K.3.3 Read Memory Command (READ)

Reading data from the ATAES132 requires the following sequence:

1. The Host drives the CS line low to select a device,
2. Then transmits the Read instruction code on the SI line,
3. Then followed by the address of the byte to be read.
4. The Client ignores any data on the SI line that follows a Read Memory instruction.

The Client shifts out the data at the specified address on the SO line. If only one byte is to be read, the CS line must be driven high after the data byte comes out. If multiple bytes are to be read, the Host can sequentially clock the data out of the ATAES132 since the byte address is automatically incremented. The CS line must be driven high by the Host after the last data byte is read. If the highest address is reached, the Address Counter will not roll over.

Figure K-3. SPI READ Memory Timing

When any error occurs, the EERR bit of the STATUS Register is set to 1b to indicate an error. If the command is processed without error, the EERR bit is set to 0b.

Note: If an SPI Read begins at an authorized address but continues into protected memory; the EERR bit will be set to 1b.
K.3.4 Write Memory Command (WRITE)

In order to write to the ATAES132, two separate instructions must be executed. First, the device must be write enabled via the Write Enable (WREN) instruction. Then a Write Memory instruction may be executed. All commands received while a write cycle is in progress will be ignored, except the Read Status Register (RDSR) instruction.

A Write Memory command requires the following sequence:
1. The Host drives the CS line low to select a device,
2. Then transmits the Write instruction code on the SI line,
3. Then followed by the address of the byte to write and the 1 to 32 data bytes to be written.

The byte address is automatically incremented as each byte is clocked in. The CS line must be driven high by the Host during the SCK low time immediately after clocking in the last data bit. The low-to-high transition of the CS pin initiates the EEPROM Write process. The SO pin remains in the high-impedance state during the entire Write sequence.

The Ready/Busy Status of the device can be determined by initiating a Read Status Register (RDSR) instruction. If the WIP status bit is 1b, the write cycle is still in progress. If the WIP Status bit is 0b, the write cycle has ended, and the ATAES132 is ready to accept a new command. Only the Read Status Register (RDSR) instruction is enabled during the EEPROM Write cycle.

The ATAES132 is capable of a 32-byte Page Write operation. After each byte of data is received, the data address is internally incremented by one. If more than 32 bytes of data are transmitted or if the page boundary is crossed, then no data will be written. The ATAES132 is automatically returned to the write disable state at the completion of a write cycle.

Figure K-4. SPI Write Memory Timing

When any error occurs, the RRDY and EERR bits of the STATUS Register are set to 1b to indicate an error. The Host can read the error code from the Response Memory Buffer (address 0xFE00) using the Read command. If the command is processed without error, the EERR bit is set to 0b. Reading the Response Memory Buffer does not reset the error code or the STATUS Register.

If the device is not Write Enabled (WREN), the device will ignore the Write instruction and will return to the waiting for a command. A new CS falling edge is required prior to the new instruction code.
K.3.5 Read Status Register Command (RDSR)

The Read Status Register instruction provides access to the STATUS Register. The Ready/Busy status of the device can be determined using the RDSR instruction. Alternately, the STATUS Register can be read directly from memory, as described in Appendix G.2.4, Read STATUS Register.

If the ATAES132 is performing an EEPROM Memory Write or is processing a command when the STATUS read is performed, then all eight bits are ones if the RDSR command is used to read the STATUS Register, emulating the behavior of Atmel Serial EEPROM. See Appendix G, Understanding the STATUS Register for a detailed description of the STATUS Register bits and Status bit behavior.

Table K-3. Device Status Register Definition

<table>
<thead>
<tr>
<th>Bit</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>EERR</td>
</tr>
<tr>
<td>6</td>
<td>RRDY</td>
</tr>
<tr>
<td>5</td>
<td>Reserved</td>
</tr>
<tr>
<td>4</td>
<td>CRCE</td>
</tr>
<tr>
<td>3</td>
<td>Reserved</td>
</tr>
<tr>
<td>2</td>
<td>WAKEb</td>
</tr>
<tr>
<td>1</td>
<td>WEN</td>
</tr>
<tr>
<td>0</td>
<td>WIP</td>
</tr>
</tbody>
</table>

The Device Status Register can always be read even if the the ATAES132 is processing a command or writing the EEPROM. The SPI RDSR command is the preferred method for reading the STATUS in SPI interface mode. If the ATAES132 is in the Sleep or Standby power state, reading the STATUS Register forces the ATAES132 to wake-up; the STATUS Register is 0xFF until the wake-up process is complete.

Table K-4. Read Status Register Bit Definition Using the SPI RDSR Command

<table>
<thead>
<tr>
<th>Bit</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>WIP = 0b = Device is ready, waiting for a command. 1b = Write cycle or a cryptographic operation is in progress.</td>
</tr>
<tr>
<td>1</td>
<td>WEN = 0b = Device is not SPI Write enabled. 1b = Device is SPI Write enabled.</td>
</tr>
<tr>
<td>2</td>
<td>WAKEb = 0b = Device is not in the Sleep or Standby power state. 1b = Device is in the Sleep or Standby power state.</td>
</tr>
<tr>
<td>3</td>
<td>Reserved = Always 0b. This bit is reserved for future use.</td>
</tr>
<tr>
<td>4</td>
<td>CRCE = 0b = The most recent command block contained a correct Checksum (CRC). 1b = The most recent command block contained an error.</td>
</tr>
<tr>
<td>5</td>
<td>Reserved = Always 0b. This bit is reserved for future use.</td>
</tr>
<tr>
<td>6</td>
<td>RRDY = 0b = Response Memory Buffer is empty. 1b = Response Memory Buffer is ready to read.</td>
</tr>
<tr>
<td>7</td>
<td>EERR = 0b = Most recent command did not generate an error during execution. 1b = Most recent command generated an execution error.</td>
</tr>
</tbody>
</table>

Notes:
1. When the SPI RDSR command is used to read the STATUS Register during an EEPROM Write or during execution of any ATAES132 command, then status bits 0 – 7 are 1b. The reserved bits will read as 0b if the STATUS Register is read directly from memory during an EEPROM Write or during execution of an ATAES132 command.
2. STATUS Register bits 0 – 7 are 1b during wake-up and power-up. See for Appendix L, Power Management additional information.
Reading the STATUS Register does not change the contents STATUS Register or the contents of the Response Memory Buffer.

K.4 Timing Diagram

Figure K-6. SPI Synchronous Data Timing (see Section 9.4.3, SPI Interface Timing)
Appendix L. Power Management

The ATAES132 contains several features that facilitate power management. This appendix describes the various power states and features.

L.1 Power State Descriptions

The ATAES132 has three powered states and the Off state. Two low-power states are available to reduce power consumption when the system is not using the ATAES132.

L.1.1 Active State

The ATAES132 is in the Active state after it has completed the power-up process and is fully powered. The WIP Status bit is 0b when the ATAES132 is in the Active state and waiting for a command. The WIP Status bit is 1b when the ATAES132 is in the Active state and processing a command or performing an EEPROM Write. (See Appendix G.1.1, WIP Status Bit [0] for WIP status bit information)

The supply current of the ATAES132 in the Active state is several milliamps (see Section 9.3.1, Supply Characteristics for ICC specifications).

An ATAES132 in the Active state is capable of accepting a command immediately if the WIP Status bit is 0b. The I2C timing specifications for the Active state are in Section 9.4.1, Power-Up, Sleep, Standby, and Wake-Up Timing. The SPI timing specifications for the Active state are in Section 9.4.3, SPI Interface Timing.

L.1.2 Standby State

ATAES132 can enter the Standby state in two ways:

- The Host can send a Sleep command to place the ATAES132 into Standby, or
- The ATAES132 will automatically enter the Standby state at power-up if configured to do so (see Appendix L.2.1, Power-Up). The Standby state preserves the ATAES132 volatile memory contents and the security state.

All eight status bits are 1b when the ATAES132 is in the Standby state and during the wake-up process (see Appendix G, Understanding the STATUS Register for Status bit information).

The supply current of ATAES132 in the Standby state is several microamperes (see Section 9.3.1 for Isb specifications).

An ATAES132 in the Standby state is capable of reporting the device status immediately, but cannot accept a command until the wake-up process is complete. The timing specifications for exiting the Standby state are in Section 9.4.1, Power-Up, Sleep, Standby, and Wake-Up Timing.
L.1.3 Sleep State
The ATAES132 can enter the Sleep state in two ways:

- The Host can send a Sleep command to place the ATAES132 into Standby, or
- The ATAES132 will automatically enter the Sleep state at power-up if configured to do so (see Appendix L.2.1).

The Sleep state clears the ATAES132 volatile memory contents and the security state.

All eight Status bits are 1b when the ATAES132 is in the Sleep state and during the wake-up process (see Appendix G for Status bit information).

The supply current of the ATAES132 in the Standby state is less than one microampere (see Section 9.3.1 for ISB specifications).

An ATAES132 in the Sleep state is capable of reporting the device STATUS immediately but cannot accept a command until the wake-up process is complete. The timing specifications for exiting the Sleep state are in Section 9.4.1, Power-Up, Sleep, Standby, and Wake-Up Timing.

L.1.4 Off State
When the ATAES132 device is unpowered or when \( V_{CC} \) is significantly below the minimum \( V_{CC} \) voltage, the device is in the Off state. A device in the Off state cannot respond to any commands.

L.2 Power State Transitions
Power-Up is a transition from the Off state to one of the three powered states. Power-down is the transition from a powered state to the Off state. Wake-up is the transition from one of the two low-power states to the Active state.

L.2.1 Power-Up
Power-Up begins when the power supply is turned on, causing the \( V_{CC} \) voltage to rise continuously from \( V_{SS} \) to the operating voltage. Power-Up occurs in three stages.

1. **First Stage:** The voltage regulator and other analog circuitry are activated.
2. **Second Stage:** The serial interface logic is activated so that the ATAES132 can report the device status to the Host.
3. **Third Stage:** The ATAES132 enters the state specified by the ChipConfig Register.

During the power-up process, the device is unable to accept commands. In the SPI interface mode, the device is ready to receive a Read Status Register command after the Power-Up Time, \( t_{PU,STATUS} \). The Power-Up Ready Time (\( t_{PU,RDY} \)) specifies the time required to complete the power-up process. In the \( I^2C \) interface mode, the device will NAK all instructions prior to the completion of Power-Up (time \( t_{PU,RDY} \)).

The last stage of the power-up procedure is to enter the Active, Standby, or Sleep state specified by bits 6 and 7 of the ChipConfig Register. The ChipState Register is set to 0xFFFF at power-up (see Appendix L.3, Understanding the ChipState Register).

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Device goes to the Active state at power-up.</td>
</tr>
<tr>
<td>1</td>
<td>0</td>
<td>Device goes to the Standby state at power-up.</td>
</tr>
<tr>
<td>0</td>
<td>1</td>
<td>Device goes to the Sleep state at power-up.</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>
During power-up, the SPI Chip Select should follow the $V_{CC}$ voltage. It is recommended that the $\overline{CS}$ pin be connected to $V_{CC}$ with a pull-up resistor if the ATAES132 is configured in the SPI interface mode. The ATAES132 does not support hot swapping or hot plugging. Connecting or disconnecting this device to a system while power is energized can cause permanent damage to the ATAES132.

L.2.2 Power-Down

Before power-down, the device must be deselected (if configured for SPI) and placed in the Active, Standby, or Sleep state. During power-down, the SPI Chip Select should be allowed to follow the $V_{CC}$ voltage if the ATAES132 is configured in SPI interface mode.

The ATAES132 should not be powered down when the WIP status bit indicates that an EEPROM Write or cryptographic operation is in progress. If the WIP status bit is 0b, then it is safe to power-down the device.

L.2.3 Entering the Standby State

If the ATAES132 is in the Active state, the Host can send a Sleep command to place the ATAES132 in the Standby state (see Section 7.23, Sleep Command). It is not possible to transition the device directly from the Sleep state to the Standby state. The Host must wake-up the device and then send a Sleep command to place the device in standby.

The device can also be configured to enter the Standby state at power-up as described in Appendix L.2.1, Power-Up.

The ATAES132 exits Standby state only if a Wake-Up event occurs on the I/O pins. Wake-Up is discussed in Appendix L.2.5, SPI Wake-Up and L.2.6, I2C Wake-Up. The ChipState Register does not change when the ATAES132 enters or leaves the Standby state (see see Appendix L.3, Understanding the ChipState Register).

L.2.4 Entering the Sleep State

If the ATAES132 is in the Active state, the Host can send a Sleep command to place the ATAES132 in the Sleep state (see Section 7.23). It is not possible to transition the device directly from the Standby state to the Sleep state. The Host must wake-up the device and then send a Sleep command to place the device in the Sleep state.

The device can also be configured to enter the Sleep state at power-up, as described in Section L.2.1.

The ATAES132 exits Sleep mode only if a Wake-Up event occurs on the I/O pins. Wake-up is discussed in Sections L.2.5 and L.2.6. The ChipState Register changes to 0x5555 when the ATAES132 leaves the Sleep state (see Appendix Appendix L.3).

L.2.5 SPI Wake-Up

To wake-up the ATAES132 configured for SPI interface mode, the Host is required to read the Status Register using the SPI Read Status Register command. The ATAES132 will answer the SPI Read Status Register command with the device status if the Host has not violated the 100ns minimum $t_{CSS, Wup}$ setup time requirement. The ATAES132 status will indicate the device is busy (status = 0xFF) during wake-up. When wake-up is complete, the ATAES132 status changes to indicate the device is in the Active state.

The ATAES132 will accept the SPI Read Status Register command only while it is busy. All other commands will be ignored. The SPI Read Status Register command is described in Appendix K.3.5, Read Status Register Command (RDSR).
The wake-up process begins when a device in the Standby or Sleep state experiences a high-to-low transition of the CS pin. The device is ready to receive a Read Status Register command from the Host after Wake-Up Time $t_{WupSB.STATUS}$ for the Standby state, or $t_{WupSL.STATUS}$ for the Sleep state. The wake-up is complete after the Wake-Up Ready Time of $t_{WupSB.RDY}$ for the Standby state or $t_{WupSL.RDY}$ for the Sleep state; $t_{WupSB.RDY}$ and $t_{WupSL.RDY}$ begin when the CS pin high-to-low transition occurs and end when the device enters the Active state. The Wake-Up timing specifications are in Table 9-5.

### L.2.6 I²C Wake-Up

To wake-up an ATAES132 configured for I²C interface mode, the Host is required to perform ACK polling using the matching I²C Device Address. The ATAES132 will answer the ACK poll with an I²C NAK to indicate the device is busy during wake-up. The ACK poll reply will change to ACK when the device is in the Active state.

The ATAES132 will not accept any commands while it is busy. The ATAES132 will NAK the I²C Device Address if it does not match the internal I²C Device Address and will not wake-up if a nonmatching I²C Device Address is received.

The wake-up process begins when a device in the Standby or Sleep state receives an I²C start signal, followed immediately by an I²C Device Address that matches the ATAES132 I2CAddr Register. The device is ready to receive an ACK poll from the Host after Wake-Up Time $t_{WupSB.STATUS}$ for the Standby state or $t_{WupSL.STATUS}$ for the Sleep state. The wake-up is complete after the Wake-Up Ready Time of $t_{WupSB.RDY}$ for the Standby state or $t_{WupSL.RDY}$ for the Sleep state; $t_{WupSB.RDY}$ and $t_{WupSL.RDY}$ begin when a matching I2C Address is received, and end when the device enters the Active state. The wake-up timing specifications are in Table 9-5.
L.3 Understanding the ChipState Register

The INFO command (see Section 7.12, INFO Command) provides access to the ChipState Register. The ChipState Register value indicates if the device has recently experienced a power-up event or wake-up from the Sleep Power state. This information can be useful for determining how to recover from an unexpected transaction error.

Table L-2. Description of the ChipState Register Value Returned by the INFO command

<table>
<thead>
<tr>
<th>ChipState</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0x0000</td>
<td>ChipState = Active. Device has remained active since the previous Crypto command was processed.(^{(1)})</td>
</tr>
<tr>
<td>0x5555</td>
<td>ChipState = Wake-up from sleep. Device has experienced a wake-up from the Sleep Power state since the previous Crypto command was processed.(^{(1)})</td>
</tr>
<tr>
<td>0xFFFF</td>
<td>ChipState = Power-up. Device has experienced a power up event since the previous Crypto command was processed.(^{(1)})</td>
</tr>
</tbody>
</table>

Note: 1. The following subsections describe the events that cause ChipState to change values and events that do not change ChipState.

L.3.1 ChipState = Power-Up

The following events cause the ChipState Register to be set to the Power-Up state (0xFFFF). The events in this table cause the device to be initialized and placed in the power state specified in the ChipConfig Register (see Appendix L.2.1, Power-Up).

Table L-3. Description of Events Causing the ChipState Register to be Set to 0xFFFF

<table>
<thead>
<tr>
<th>Event</th>
<th>Event description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Power-up</td>
<td>Power-up of the device (Appendix L.2.1, Power-Up).</td>
</tr>
<tr>
<td>Power Interruption</td>
<td>Power interruption or brownout resulting in device reset.</td>
</tr>
</tbody>
</table>

L.3.2 ChipState = Wake-Up from Sleep

The following events cause the ChipState Register to be set to the wake-up from Sleep state (0x5555). The events in this table cause the security registers to be cleared, the logic reinitialized, and the device returned to the Active Power state (ready to receive a command).

Table L-4. Description of Events Causing the ChipState Register to be Set to 0x5555

<table>
<thead>
<tr>
<th>Event</th>
<th>Event Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wake-up from Sleep</td>
<td>Wake-up from the Sleep power state. (Appendix L.1.3, Sleep State)</td>
</tr>
<tr>
<td>Reset Command</td>
<td>Device receives a valid Reset command block. (Section 7.22, Reset Command)</td>
</tr>
<tr>
<td>Tamper</td>
<td>Device reset initiated by the tamper sensors. (Section 3.1.2, Hardware Security Features)</td>
</tr>
</tbody>
</table>
### L.3.3 Events that Do Not Change ChipState

The following events cause *no change* in the ChipState Register value. These events do not modify the security state of the ATAES132; therefore, do not cause the ChipState to change.

**Table L-5. Description of Events Causing No Change in the ChipState Register**

<table>
<thead>
<tr>
<th>Event</th>
<th>Event Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Wake-Up from Standby</td>
<td>Wake-Up from the Standby Power state. (Appendix L.1.2, Standby State)</td>
</tr>
<tr>
<td>Reading STATUS</td>
<td>Reading the STATUS Register with SPI RDSR or standard Read commands. (Appendix G, Understanding the STATUS Register)</td>
</tr>
<tr>
<td>Writing IO Address Reset</td>
<td>Writing the IO Address Reset Register. (Appendix D.3, IO Address Reset Register)</td>
</tr>
<tr>
<td>Reading a Response</td>
<td>Reading the Response Memory Buffer. (Appendix D.2, Response Memory Buffer)</td>
</tr>
<tr>
<td>Command CRC Error</td>
<td>Device receives any command block which results in a CRCE error. (1) (Appendix G.1.4, CRCE Status Bit [4])</td>
</tr>
<tr>
<td>Command Invalid</td>
<td>Device receives a command block containing an undefined/invalid opcode. (Section 6.2, Command Summary).</td>
</tr>
<tr>
<td>ACK Polling</td>
<td>I²C Acknowledge Polling. (Appendix J.3.7, Acknowledge Polling)</td>
</tr>
<tr>
<td>I²C Read</td>
<td>I²C Standard Read (READ, RREAD, SREAD instructions) (Appendix J.3, I²C Instruction Set).</td>
</tr>
<tr>
<td>Invalid I²C Write</td>
<td>I²C standard Write beginning at any address from 0x1000 to 0xEF FF or above 0xF300, except address 0xFE00 (BWRITE, PWRITE instructions) (1) (Appendix J.3).</td>
</tr>
<tr>
<td>I²C SRESET</td>
<td>I²C SRESET instruction (Appendix J.3.6, Software Reset (SRESET)).</td>
</tr>
<tr>
<td>SPI Read</td>
<td>SPI standard read [READ instruction] (Appendix K.3.3, Read Memory Command (READ)).</td>
</tr>
<tr>
<td>Invalid SPI Write</td>
<td>SPI standard write beginning at any address from 0x1000 to 0xEE FF or above 0xF300, except address 0xFE00 (WREN, WRITE, WRDI instructions) (1) (Appendix K.3, SPI Instruction Set).</td>
</tr>
<tr>
<td>INFO command</td>
<td>Device receives a valid INFO command block (Section 7.12, INFO Command).</td>
</tr>
</tbody>
</table>

**Notes:**
1. A CRCE error results from a command block with a short count, bad checksum, or buffer overrun.
2. Writing the Command Memory Buffer (address 0xFE00) may or may not change ChipState, depending on which command is written to the buffer.
### L.3.4 ChipState = Active

The following events cause the ChipState Register to be set to the Active state (0x0000). The events in this table may result in a change in the security state of the device.

#### Table L-6. Description of Events Causing the ChipState Register to be Set to 0x0000

<table>
<thead>
<tr>
<th>Event</th>
<th>Event Description</th>
<th>Section</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auth Command</td>
<td>Device receives a valid Auth command block.</td>
<td>7.1</td>
</tr>
<tr>
<td>AuthCheck Command</td>
<td>Device receives a valid AuthCheck command block.</td>
<td>7.2</td>
</tr>
<tr>
<td>AuthCompute Command</td>
<td>Device receives a valid AuthCompute command block.</td>
<td>7.3</td>
</tr>
<tr>
<td>BlockRead Command</td>
<td>Device receives a valid BlockRead command block.</td>
<td>7.4</td>
</tr>
<tr>
<td>Counter Command</td>
<td>Device receives a valid Counter command block.</td>
<td>7.5</td>
</tr>
<tr>
<td>Crunch Command</td>
<td>Device receives a valid Crunch command block.</td>
<td>7.6</td>
</tr>
<tr>
<td>DecRead</td>
<td>Device receives a valid DecRead command block.</td>
<td>7.7</td>
</tr>
<tr>
<td>Decrypt Command</td>
<td>Device receives a valid Decrypt command block.</td>
<td>7.8</td>
</tr>
<tr>
<td>EncRead Command</td>
<td>Device receives a valid EncRead command block.</td>
<td>7.9</td>
</tr>
<tr>
<td>Encrypt Command</td>
<td>Device receives a valid Encrypt command block.</td>
<td>7.10</td>
</tr>
<tr>
<td>EncWrite Command</td>
<td>Device receives a valid EncWrite command block.</td>
<td>7.11</td>
</tr>
<tr>
<td>KeyCreate Command</td>
<td>Device receives a valid KeyCreate command block.</td>
<td>7.13</td>
</tr>
<tr>
<td>KeyImport Command</td>
<td>Device receives a valid KeyImport command block.</td>
<td>7.14</td>
</tr>
<tr>
<td>KeyLoad Command</td>
<td>Device receives a valid KeyLoad command block.</td>
<td>7.15</td>
</tr>
<tr>
<td>KeyTransfer Command</td>
<td>Device receives a valid KeyTransfer command block.</td>
<td>7.16</td>
</tr>
<tr>
<td>Legacy Command</td>
<td>Device receives a valid Legacy command block.</td>
<td>7.17</td>
</tr>
<tr>
<td>Lock Command</td>
<td>Device receives a valid Lock command block.</td>
<td>7.18</td>
</tr>
<tr>
<td>Nonce Command</td>
<td>Device receives a valid Nonce command block.</td>
<td>7.19</td>
</tr>
<tr>
<td>NonceCompute Command</td>
<td>Device receives a valid NonceCompute command block.</td>
<td>7.20</td>
</tr>
<tr>
<td>Random Command</td>
<td>Device receives a valid Random Command block.</td>
<td>7.21</td>
</tr>
<tr>
<td>Sleep Command</td>
<td>Device receives a valid Sleep command block.</td>
<td>7.23</td>
</tr>
<tr>
<td>WriteCompute Command</td>
<td>Device receives a valid WriteCompute command block.</td>
<td>7.24</td>
</tr>
<tr>
<td>I²C Write</td>
<td>I²C standard Write beginning at any user zone address, any Configuration Memory address, or any Key Memory address (BWRITE, PWRITE instructions).</td>
<td>J.3</td>
</tr>
<tr>
<td>SPI Write</td>
<td>SPI standard Write beginning at any user zone address, any Configuration Memory address, or any Key Memory address (WREN, WRITE, WRDI instructions).</td>
<td>K.3</td>
</tr>
</tbody>
</table>
Appendix M. Block Checksum

An Atmel CRC-16 Checksum is used to verify the integrity of blocks communicated to and from the ATAES132. The Host sends ATAES132 extended commands to the device in a block of at least four bytes. The ATAES132 responses are returned to the Host in a block of at least four bytes. The command and response blocks are constructed in the following manner:

<table>
<thead>
<tr>
<th>Byte #</th>
<th>Name</th>
<th>Meaning</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Count</td>
<td>Number of bytes to be transferred to the device in the block, including count, packet, and checksum. This byte will always have a value of N.</td>
</tr>
<tr>
<td>1 to (N-3)</td>
<td>Packet</td>
<td>Command, parameters and data, or response. Data are transmitted in the byte order shown in command definitions in Section 7, Command Definitions.</td>
</tr>
<tr>
<td>N-2, N-1</td>
<td>Checksum</td>
<td>Atmel CRC-16 verification of the Count and packet bytes.</td>
</tr>
</tbody>
</table>

The Atmel CRC-16 polynomial is 0x8005. The initial register value should be 0x0000. After the last bit of the Count and packet has been transmitted, the internal CRC Register should have a value that matches that in the block. The first Checksum byte transmitted (N-2) is the most-significant byte of the CRC value, and the last byte of the block is the least-significant byte of the CRC.
### M.1 Checksum Function

```c
/** \This function calculates a 16-bit CRC.
 * \param[in] count number of bytes in data buffer
 * \param[in] data pointer to data
 * \param[out] crc pointer to calculated CRC (high byte at crc[0])
 */
void CalculateCrc(uint8_t length, uint8_t *data, uint8_t *crc)
{
    uint8_t counter;
    uint8_t crcLow = 0, crcHigh = 0, crcCarry;
    uint8_t polyLow = 0x05, polyHigh = 0x80;
    uint8_t shiftRegister;
    uint8_t dataBit, crcBit;

    for (counter = 0; counter < length; counter++) {
        for (shiftRegister = 0x80; shiftRegister > 0x00; shiftRegister >>= 1) {
            dataBit = (data[counter] & shiftRegister) ? 1 : 0;
            crcBit = crcHigh >> 7;

            // Shift CRC to the left by 1.
            crcCarry = crcLow >> 7;
            crcLow <<= 1;
            crcHigh <<= 1;
            crcHigh |= crcCarry;

            if ((dataBit ^ crcBit) != 0) {
                crcLow ^= polyLow;
                crcHigh ^= polyHigh;
            }
        }
    }

crc[0] = crcHigh;
crc[1] = crcLow;
}
```

### M.2 Checksum Examples

DATA = 09 02 02 00 00 00 00 CRC = 0xF960
Appendix N. ATAES132 Command Response Time

The typical and maximum time required for the ATAES132 to process an extended command is shown in Table N-1. The response time is the time from sending the last bit of the last byte of the command block to the Command Memory Buffer until the STATUS Register (or I2C ACK) indicates the response block is available. The typical response time is the average time required for an error-free command to be processed on a typical device at room temperature. The maximum response time is the worst-case time for the command to be processed over the specified temperature range (with or without an error condition, whichever results in the worst response time).

Table N-1. ATAES132 Extended Commands Typical and Maximum Response Times (1)

<table>
<thead>
<tr>
<th>Command Description</th>
<th>Typical ms (2)</th>
<th>Maximum ms (3)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Auth, Reset (Mode [0:1] = 00b)</td>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>Auth, Inbound-Only (Mode [5:7] = 000b)</td>
<td>1.7</td>
<td>2.4</td>
</tr>
<tr>
<td>Auth, Inbound-Only (Mode [5:7] not 000b)</td>
<td>2.0</td>
<td>2.8</td>
</tr>
<tr>
<td>Auth, Inbound-Only (Mode [5:7] not 000b), with Key Usage</td>
<td>5.3</td>
<td>21.0</td>
</tr>
<tr>
<td>Auth, Outbound-Only (Mode [5:7] = 000b)</td>
<td>1.7</td>
<td>2.4</td>
</tr>
<tr>
<td>Auth, Outbound-Only (Mode [5:7] not 000b)</td>
<td>2.0</td>
<td>2.8</td>
</tr>
<tr>
<td>Auth, Outbound-Only (Mode [5:7] not 000b), with Key Usage</td>
<td>5.3</td>
<td>21.0</td>
</tr>
<tr>
<td>Auth, Mutual (Mode [5:7] = 000b)</td>
<td>2.6</td>
<td>3.6</td>
</tr>
<tr>
<td>Auth, Mutual (Mode [5:7] not 000b)</td>
<td>3.1</td>
<td>4.3</td>
</tr>
<tr>
<td>Auth, Mutual (Mode [5:7] not 000b), with Key Usage</td>
<td>6.4</td>
<td>22.6</td>
</tr>
<tr>
<td>AuthCheck</td>
<td>1.9</td>
<td>2.7</td>
</tr>
<tr>
<td>AuthChec, with Key Usage.</td>
<td>5.2</td>
<td>20.9</td>
</tr>
<tr>
<td>AuthCompute</td>
<td>2.0</td>
<td>2.7</td>
</tr>
<tr>
<td>AuthCompute, with Key Usage.</td>
<td>5.3</td>
<td>20.9</td>
</tr>
<tr>
<td>BlockRead, 32 bytes</td>
<td>0.9</td>
<td>1.3</td>
</tr>
<tr>
<td>Counter, Read, without MAC</td>
<td>0.6</td>
<td>0.8</td>
</tr>
<tr>
<td>Counter, Read, with OutMAC (Mode [5:7] = 000b)</td>
<td>1.8</td>
<td>2.5</td>
</tr>
<tr>
<td>Counter, Read, with OutMAC (Mode [5:7] not 000b)</td>
<td>2.1</td>
<td>2.9</td>
</tr>
<tr>
<td>Counter, Read, with OutMAC (Mode [5:7] not 000b), with Key Usage</td>
<td>5.4</td>
<td>21.1</td>
</tr>
<tr>
<td>Counter, Increment, without MAC</td>
<td>3.9</td>
<td>4.4</td>
</tr>
<tr>
<td>Counter, Increment, with InMAC (Mode [5:7] = 000b)</td>
<td>5.1</td>
<td>6.2</td>
</tr>
<tr>
<td>Counter, Increment, with InMAC (Mode [5:7] not 000b)</td>
<td>5.4</td>
<td>6.5</td>
</tr>
<tr>
<td>Counter, Increment, with InMAC (Mode [5:7] not 000b), with Key Usage</td>
<td>8.7</td>
<td>24.8</td>
</tr>
<tr>
<td>Crunch, with Count 0x0001</td>
<td>0.9</td>
<td>1.2</td>
</tr>
<tr>
<td>DecRead</td>
<td>2.5</td>
<td>3.5</td>
</tr>
<tr>
<td>Command Description</td>
<td>Typical (ms)</td>
<td>Maximum (ms)</td>
</tr>
<tr>
<td>-------------------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>DecRead, with Key Usage. <em>(5)</em></td>
<td>5.9</td>
<td>21.8</td>
</tr>
<tr>
<td>Decrypt, 1 to 16 bytes (Mode [5:7] = 000b)</td>
<td>2.4</td>
<td>3.4</td>
</tr>
<tr>
<td>Decrypt, 1 to 16 bytes (Mode [5:7] not 000b)</td>
<td>2.7</td>
<td>3.7</td>
</tr>
<tr>
<td>Decrypt, 1 to 16 bytes (Mode [5:7] not 000b), with Key Usage. <em>(5)</em></td>
<td>6.0</td>
<td>21.9</td>
</tr>
<tr>
<td>Decrypt, 17 to 32 bytes (Mode [5:7] = 000b)</td>
<td>3.2</td>
<td>4.3</td>
</tr>
<tr>
<td>Decrypt, 17 to 32 bytes (Mode [5:7] not 000b)</td>
<td>3.4</td>
<td>4.7</td>
</tr>
<tr>
<td>Decrypt, 17 to 32 bytes (Mode [5:7] not 000b), with Key Usage. <em>(5)</em></td>
<td>6.7</td>
<td>22.9</td>
</tr>
<tr>
<td>EncRead, 1 to 16 bytes (Mode [5:7] = 000b)</td>
<td>2.5</td>
<td>3.5</td>
</tr>
<tr>
<td>EncRead, 1 to 16 bytes (Mode [5:7] not 000b)</td>
<td>2.8</td>
<td>3.9</td>
</tr>
<tr>
<td>EncRead, 1 to 16 bytes (Mode [5:7] not 000b), with Key Usage. <em>(5)</em></td>
<td>6.1</td>
<td>22.1</td>
</tr>
<tr>
<td>EncRead, 17 to 32 bytes (Mode [5:7] = 000b)</td>
<td>3.2</td>
<td>4.5</td>
</tr>
<tr>
<td>EncRead, 17 to 32 bytes (Mode [5:7] not 000b)</td>
<td>3.5</td>
<td>4.8</td>
</tr>
<tr>
<td>EncRead, 17 to 32 bytes (Mode [5:7] not 000b), with Key Usage. <em>(5)</em></td>
<td>6.8</td>
<td>23.1</td>
</tr>
<tr>
<td>EncRead, Configuration Memory Signature Generation Mode</td>
<td>9.1</td>
<td>12.7</td>
</tr>
<tr>
<td>EncRead, Key Memory Signature Generation Mode</td>
<td>13.9</td>
<td>18.4</td>
</tr>
<tr>
<td>Encrypt, 1 to 16 bytes (Mode [5:7] = 000b)</td>
<td>2.4</td>
<td>3.4</td>
</tr>
<tr>
<td>Encrypt, 1 to 16 bytes (Mode [5:7] not 000b)</td>
<td>2.7</td>
<td>3.7</td>
</tr>
<tr>
<td>Encrypt, 1 to 16 bytes (Mode [5:7] not 000b), with Key Usage. <em>(6)</em></td>
<td>6.0</td>
<td>21.9</td>
</tr>
<tr>
<td>Encrypt, 17 to 32 bytes (Mode [5:7] = 000b)</td>
<td>3.0</td>
<td>4.1</td>
</tr>
<tr>
<td>Encrypt, 17 to 32 bytes (Mode [5:7] not 000b)</td>
<td>3.2</td>
<td>4.5</td>
</tr>
<tr>
<td>Encrypt, 17 to 32 bytes (Mode [5:7] not 000b), with Key Usage. <em>(5)</em></td>
<td>6.5</td>
<td>22.7</td>
</tr>
<tr>
<td>EncWrite, 1 to 16 bytes (Mode [5:7] = 000b)</td>
<td>9.1</td>
<td>10.8</td>
</tr>
<tr>
<td>EncWrite, 1 to 16 bytes (Mode [5:7] not 000b)</td>
<td>9.4</td>
<td>11.1</td>
</tr>
<tr>
<td>EncWrite, 1 to 16 bytes (Mode [5:7] not 000b), with Key Usage. <em>(5)</em></td>
<td>12.4</td>
<td>29.0</td>
</tr>
<tr>
<td>EncWrite, 17 to 32 bytes (Mode [5:7] = 000b)</td>
<td>9.9</td>
<td>11.9</td>
</tr>
<tr>
<td>EncWrite, 17 to 32 bytes (Mode [5:7] not 000b)</td>
<td>10.2</td>
<td>12.2</td>
</tr>
<tr>
<td>EncWrite, 17 to 32 bytes (Mode [5:7] not 000b), with Key Usage. <em>(5)</em></td>
<td>13.2</td>
<td>30.1</td>
</tr>
<tr>
<td>EncWrite a Key (Mode [5:7] = 000b)</td>
<td>15.8</td>
<td>18.1</td>
</tr>
<tr>
<td>EncWrite a Key (Mode [5:7] not 000b)</td>
<td>16.1</td>
<td>18.5</td>
</tr>
<tr>
<td>EncWrite a Key (Mode [5:7] not 000b), with Key Usage. <em>(5)</em></td>
<td>19.4</td>
<td>36.7</td>
</tr>
<tr>
<td>INFO</td>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>KeyCreate, without RNG Seed Update. (Mode [5:7] = 000b)</td>
<td>17.0</td>
<td>19.9</td>
</tr>
<tr>
<td>Command Description</td>
<td>Typical (ms)</td>
<td>Maximum (ms)</td>
</tr>
<tr>
<td>----------------------------------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>KeyCreate, without RNG Seed Update. (Mode [5:7] not 000b)</td>
<td>17.3</td>
<td>20.2</td>
</tr>
<tr>
<td>KeyCreate, without RNG Seed Update. (Mode [5:7] not 000b), with Key Usage. (5)</td>
<td>20.6</td>
<td>38.5</td>
</tr>
<tr>
<td>KeyCreate, with RNG Seed Update. (Mode [5:7] = 000b)</td>
<td>32.4</td>
<td>37.4</td>
</tr>
<tr>
<td>KeyCreate, with RNG Seed Update. (Mode [5:7] not 000b)</td>
<td>32.9</td>
<td>38.2</td>
</tr>
<tr>
<td>KeyCreate, with RNG Seed Update. (Mode [5:7] not 000b), with Key Usage. (5)</td>
<td>35.2</td>
<td>54.9</td>
</tr>
<tr>
<td>KeyCreate, VolatileKey with RNG Seed Update. (Mode [5:7] = 000b)</td>
<td>18.8</td>
<td>22.4</td>
</tr>
<tr>
<td>KeyCreate, VolatileKey with RNG Seed Update. (Mode [5:7] not 000b)</td>
<td>19.4</td>
<td>23.1</td>
</tr>
<tr>
<td>KeyCreate, VolatileKey with RNG Seed Update. (Mode [5:7] not 000b), with Key Usage. (5)</td>
<td>22.7</td>
<td>41.3</td>
</tr>
<tr>
<td>KeyImport (Mode [5:7] = 000b)</td>
<td>15.8</td>
<td>18.2</td>
</tr>
<tr>
<td>KeyImport (Mode [5:7] not 000b)</td>
<td>16.1</td>
<td>18.5</td>
</tr>
<tr>
<td>KeyImport (Mode [5:7] not 000b), with Key Usage. (5)</td>
<td>19.4</td>
<td>36.7</td>
</tr>
<tr>
<td>KeyLoad (Mode [5:7] = 000b)</td>
<td>15.8</td>
<td>18.2</td>
</tr>
<tr>
<td>KeyLoad (Mode [5:7] not 000b)</td>
<td>16.1</td>
<td>18.5</td>
</tr>
<tr>
<td>KeyLoad (Mode [5:7] not 000b), with Key Usage. (5)</td>
<td>19.4</td>
<td>36.7</td>
</tr>
<tr>
<td>KeyTransfer</td>
<td>14.2</td>
<td>15.8</td>
</tr>
<tr>
<td>Legacy</td>
<td>1.2</td>
<td>1.7</td>
</tr>
<tr>
<td>Legacy, with Key Usage. (5)</td>
<td>4.5</td>
<td>19.9</td>
</tr>
<tr>
<td>Lock SmallZone, Key Memory, Configuration Memory, with Checksum.</td>
<td>16.8</td>
<td>20.6</td>
</tr>
<tr>
<td>Lock User Zone, without MAC</td>
<td>3.8</td>
<td>4.4</td>
</tr>
<tr>
<td>Lock User Zone, with MAC (Mode [5:7] = 000b)</td>
<td>5.1</td>
<td>6.1</td>
</tr>
<tr>
<td>Lock User Zone, with MAC (Mode [5:7] not 000b)</td>
<td>5.3</td>
<td>6.5</td>
</tr>
<tr>
<td>Lock User Zone, with MAC (Mode [5:7] not 000b), with Key Usage. (5)</td>
<td>8.7</td>
<td>24.7</td>
</tr>
<tr>
<td>Nonce, Inbound</td>
<td>0.5</td>
<td>0.7</td>
</tr>
<tr>
<td>Nonce, Random, without RNG Seed Update.</td>
<td>2.1</td>
<td>2.9</td>
</tr>
<tr>
<td>Nonce, Random, with RNG Seed Update.</td>
<td>16.8</td>
<td>19.5</td>
</tr>
<tr>
<td>NonceCompute</td>
<td>0.9</td>
<td>1.3</td>
</tr>
<tr>
<td>Random, without RNG Seed Update.</td>
<td>1.7</td>
<td>2.4</td>
</tr>
<tr>
<td>Random, with RNG Seed Update.</td>
<td>16.3</td>
<td>18.8</td>
</tr>
<tr>
<td>Reset. (4)</td>
<td>1.3</td>
<td>1.7</td>
</tr>
<tr>
<td>Sleep, enter Standby state. (4)</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>Sleep, enter Sleep state. (4)</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>WriteCompute, 1 to 16 bytes</td>
<td>2.6</td>
<td>3.7</td>
</tr>
<tr>
<td>Command Description</td>
<td>Typical (ms)</td>
<td>Maximum (ms)</td>
</tr>
<tr>
<td>----------------------------------------------------------</td>
<td>--------------</td>
<td>--------------</td>
</tr>
<tr>
<td>WriteCompute, 1 to 16 bytes, with Key Usage. (5)</td>
<td>5.9</td>
<td>21.8</td>
</tr>
<tr>
<td>WriteCompute, 17 to 32 bytes</td>
<td>3.2</td>
<td>4.4</td>
</tr>
<tr>
<td>WriteCompute, 17 to 32 bytes</td>
<td></td>
<td></td>
</tr>
<tr>
<td>WriteCompute, 17 to 32 bytes, with Key Usage. (5)</td>
<td>6.5</td>
<td>22.3</td>
</tr>
</tbody>
</table>

Notes:
1. The values in this table are based on characterization and/or simulation. These parameters are not tested.
2. The typical response time is the time required for 60% of devices to place a packet in the Response Memory Buffer and change the WIP status bit to 0b after successful execution of the command at room temperature. If an error occurs, the response will be available in a shorter amount of time.
3. The maximum response time is the time required for 95% of devices to place a packet in the Response Memory Buffer and change the WIP Status bit to 0b after successful execution of the command at the worst case temperature.
   Note: 5% of the devices may be slower than this number. The Host is expected to read the STATUS Register to determine when a response is available (see Appendix G, Understanding the STATUS Register).
4. The Reset command and the Sleep command do not generate a response. The response times are the time required for the operation to be completed.
5. These times are with the Key Usage limits enabled in the KeyConfig Register. All other times are with the Key Usage limits disabled in the KeyConfig Register.
Appendix O. Default Configuration

The ATAES132 memory map is shown in Table O-1 with the default memory values. Reserved memory cannot be written or read.

Table O-1. ATAES132 Memory Map Showing the Default Memory Contents

<table>
<thead>
<tr>
<th>Byte Address</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0000_{h}-0FFF_{h}</td>
<td>User Memory (Default = All bytes FF_{h})</td>
</tr>
<tr>
<td>1000_{h}-EFFF_{h}</td>
<td>Reserved</td>
</tr>
<tr>
<td>F000_{h}-F1FF_{h}</td>
<td>Configuration Memory (see Appendix O.1, Configuration Memory Contents for default values)</td>
</tr>
<tr>
<td>F200_{h}-F2FF_{h}</td>
<td>Key Memory (see Appendix O.2, Key Memory Contents for default values)</td>
</tr>
<tr>
<td>F300_{h}-FDFF_{h}</td>
<td>Reserved</td>
</tr>
<tr>
<td>FE00_{h}</td>
<td>Command / Response Memory Buffer</td>
</tr>
<tr>
<td>FE01_{h}-FFFD_{h}</td>
<td>Reserved</td>
</tr>
<tr>
<td>FFE0_{h}</td>
<td>I/O Address Reset</td>
</tr>
<tr>
<td>FFE1_{h}-FFEF_{h}</td>
<td>Reserved</td>
</tr>
<tr>
<td>FFF0_{h}</td>
<td>STATUS Register</td>
</tr>
<tr>
<td>FFF1_{h}-FFFF_{h}</td>
<td>Reserved</td>
</tr>
</tbody>
</table>
# O.1 Configuration Memory Contents

The default contents of the Configuration Memory after completion of production test are shown in Table O-2. This configuration enables most functions, and is expected to be changed by the customer during personalization. See Appendix E, Configuration Memory Map.

Table O-2. Default Configuration Memory contents (All Register Values Shown are Hexadecimal Numbers)

<table>
<thead>
<tr>
<th>Address</th>
<th>$0_{h} / 8_{h}$</th>
<th>$1_{h} / 9_{h}$</th>
<th>$2_{h} / A_{h}$</th>
<th>$3_{h} / B_{h}$</th>
<th>$4_{h} / C_{h}$</th>
<th>$5_{h} / D_{h}$</th>
<th>$6_{h} / E_{h}$</th>
<th>$7_{h} / F_{h}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>F000h-F007h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Unique Die Serial Number</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F008h-F00Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Atmel Proprietary Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F010h-F017h</td>
<td>00</td>
<td>1F</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Atmel Proprietary Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F018h-F01Fh</td>
<td>20</td>
<td>20</td>
<td>0A</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Atmel Proprietary Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F020h-F027h</td>
<td>55</td>
<td>55</td>
<td>55</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Atmel Proprietary Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F028h-F02Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Atmel Proprietary Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F030h-F037h</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Atmel Proprietary Data</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F038h-F03Fh</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>F040h-F047h</td>
<td>i2CAddr</td>
<td>C3</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F048h-F04Fh</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F050h-F057h</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F058h-F05Fh</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F060h-F067h</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F068h-F06Fh</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F070h-F077h</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F078h-F07Fh</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F080h-F087h</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>F088h-F08Fh</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F090h-F097h</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F098h-F09Fh</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F0A0h-F0A7h</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F0A8h-F0AFh</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F0B0h-F0B7h</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F0B8h-F0BFh</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F0C0h-F0C7h</td>
<td>00</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F0C8h-F0CFh</td>
<td>00</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F0D0h-F0D7h</td>
<td>00</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F0D8h-F0DFh</td>
<td>00</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F0E0h-F0EFh</td>
<td>00</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F0F0h-F0FFh</td>
<td>00</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F0F8h-F0FFh</td>
<td>00</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F100h-F107h</td>
<td>FF</td>
<td>FF</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>F108h-F10Fh</td>
<td>FF</td>
<td>FF</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>F110h-F117h</td>
<td>FF</td>
<td>FF</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>Address</td>
<td>0h / 8h</td>
<td>1h / 9h</td>
<td>2h / Ah</td>
<td>3h / Bh</td>
<td>4h / Ch</td>
<td>5h / Dh</td>
<td>6h / Eh</td>
<td>7h / Fh</td>
</tr>
<tr>
<td>------------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
<td>---------</td>
</tr>
<tr>
<td>F118h-F11Fh</td>
<td>FF</td>
<td>FF</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>F120h-F127h</td>
<td>FF</td>
<td>FF</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>F128h-F12Fh</td>
<td>FF</td>
<td>FF</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>F130h-F137h</td>
<td>FF</td>
<td>FF</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>F138h-F13Fh</td>
<td>FF</td>
<td>FF</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>F140h-F147h</td>
<td>FF</td>
<td>FF</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>F148h-F14Fh</td>
<td>FF</td>
<td>FF</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>F150h-F157h</td>
<td>FF</td>
<td>FF</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>F158h-F15Fh</td>
<td>FF</td>
<td>FF</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>F160h-F167h</td>
<td>FF</td>
<td>FF</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>F168h-F16Fh</td>
<td>FF</td>
<td>FF</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>F170h-F177h</td>
<td>FF</td>
<td>FF</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>F178h-F17Fh</td>
<td>FF</td>
<td>FF</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
<td>00</td>
</tr>
<tr>
<td>F180h-F187h</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F188h-F18Fh</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F190h-F197h</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F198h-F19Fh</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F1A0h-F1A7h</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F1A8h-F1AFh</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F1B0h-F1B7h</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F1B8h-F1BFh</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F1C0h-F1C7h</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F1C8h-F1CFh</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F1D0h-F1D7h</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F1D8h-F1DFh</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F1E0h-F1Ef</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F1E8h-F1EFh</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F1F0h-F1F7h</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
<tr>
<td>F1F8h-F1FFh</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
<td>FF</td>
</tr>
</tbody>
</table>

Notes:
1. Orange Registers = Locked at the factory and cannot be changed by the customer.
2. Blue Registers = Lock registers can be changed only by using the Lock command (see Section7.9, EncRead Command).
3. Green Registers = Configuration registers can be written by the customer prior to locking (by setting LockConfig to 0x00 using the Lock command).
4. Yellow Registers = The SmallZone Register can be written by the customer prior to locking (by setting LockSmall to 0x00 using the Lock command). SmallZone is locked separately from the remainder of the Configuration Memory.

The default value of the I2CAddr Register is 0x01 for devices configured for I^2^C interface mode. The default value of I^2^CAddr is 0x00 for devices configured for SPI interface mode. See Appendix Q, Ordering Information for ordering codes.
**O.2 Key Memory Contents**

The Key Memory contains pseudorandom values after completion of production test, except for Key 00 which contains the Transport Key. Device personalization can be performed without knowledge of the Transport Key; however, secure personalization can be performed only if the Transport Key value has been obtained from Atmel.
Appendix P. Serial Memory Backward Compatibility

The ATAES132 secure Serial EEPROM architecture was developed to allow security to be retrofitted into systems using standard Atmel Serial EEPROM. The ATAES132 package pinouts, the interface protocol, and the command set are all compatible with standard I²C and SPI EEPROM, but are not identical.

This section describes the differences that must be considered when the ATAES132 is inserted into systems using I²C or SPI Serial EEPROM.

P.1 I²C Serial EEPROM Compatibility

This section describes differences between the Atmel AT24C32C standard 32Kb I²C Serial EEPROM and the ATAES132 secure Serial EEPROM configured for I²C communication mode.

P.1.1 Package Pins

On AT24C32C, pins 1, 2, and 3 are used to set I²C Device Address bits A0, A1, and A2. The AT24C32C pin 7 is the Write Protect (WP) input.

On the ATAES132, pins 1, 2, 3, and 7 are not used in I²C communication mode. These pins should be tied to VCC or VSS. The state of these four pins has no impact on the functionality of the ATAES132 in the I²C communication mode. See Appendix J.2, Pin Descriptions.

P.1.2 I²C Device Address

The AT24C32C I²C Device Address is 1010A2A1A0b, with A0, A1, and A2 determined by the state of pins 1, 2, and 3. A maximum of eight AT24C32C devices are permitted on the I²C interface.

On the ATAES132, the I²C Device Address is determined by the contents of the I2CAddr Register (see Appendix J.1.3, I²C Device Address). The ATAES132 I²C Device Address can be any set to any value, allowing up to 127 devices on the I²C interface.

P.1.3 Write Protect

The AT24C32C Write Protect (WP) input pin inhibits all EEPROM Write operations when the WP pin is high. If WP is low, then EEPROM Write operations are allowed.

On the ATAES132, the User Memory Write permissions are controlled by the ZoneConfig Registers (see Appendix E.2.20, ZoneConfig Registers). The User Memory is divided into 16 user zones that are independently controlled by 16 ZoneConfig Registers; different Write permissions can be assigned to different sections of the memory. By default, all User Memory has open Write access.

P.1.4 Page Write Operations

If the Host attempts to write data across the physical (32 byte) EEPROM page boundary, the AT24C32C wraps to the beginning of the EEPROM page where the Page Write operation begins and performs the EEPROM Write after receiving a Stop condition. If the Host attempts to write more than 32 bytes in a Page Write operation, then the AT24C32C wraps the data at the page boundary and performs the EEPROM Write after receiving a Stop condition. Partial Page Writes are supported by the AT24C32C.

The ATAES132 does not allow Write operations to cross physical (32 byte) EEPROM page boundaries (see Appendix B.2, EEPROM Page Boundary) and does not allow a Write operation if more than 32 data bytes are received from the Host. In both cases, the EEPROM contents remain unchanged, the data is discarded, and an error bit is set in the STATUS Register (see Appendix J.3.3, Page Write (PWRITE)). Partial Page Writes are supported by the ATAES132.
**P.1.5 Read Operations**

Reading beyond the end of physical memory on the AT24C32C causes the internal data address register to roll-over to address zero. The Read operation continues from address zero.

If an ATAES132 Read operation begins at a valid User Memory address but continues past the end of User Memory, the Read operation will not wrap to the beginning of User Memory. Reading beyond the end of User Memory causes 0xFF to be returned to the Host in reply to the Read, the internal data address register stops incrementing, and an error bit is set in the STATUS Register (see Appendix G.2.5, Read User Memory).

**P.1.6 Read Protect**

The AT24C32C and other standard I²C EEPROMs do not have a Read inhibit function.

On the ATAES132, the User Memory Read permissions are controlled by the ZoneConfig Registers (see Appendix E.2.20, ZoneConfig Registers). The User Memory is divided into 16 user zones that are independently controlled by 16 ZoneConfig Registers; different Read permissions can be assigned to different sections of the memory. If Read access is prohibited, then 0xFF will be returned to the Host in reply to a Read command (see Section 5.1, Read). By default all User Memory has open Read access.

**P.1.7 Standby Mode**

Standard I²C EEPROMs automatically enter low-power standby mode upon completion of any internal operation.

The ATAES132 has three powered states:

- Active State and Two Low-Power States
- Standby State
- Sleep State

The ATAES132 will remain in the Active state between operations unless the Host sends a Sleep command to activate the Standby state or the Sleep state. The ATAES132 can also be configured to automatically enter a Low-Power state at power-up. See Appendix L, Power Management for details on the power management features.

**P.1.8 Operating Voltage**

- The AT24C32C operating range is 1.8V minimum to 5.5V maximum.
- The ATAES132 operating range is 2.5V minimum to 5.5V maximum. See Section 9.3, DC Characteristics.

**P.2 SPI Serial EEPROM Compatibility**

This section describes differences between the AT25320B standard Atmel 32Kb SPI Serial EEPROM and the ATAES132 secure Serial EEPROM configured for SPI communication mode.

**P.2.1 Package Pins**

On the AT25320B, pin 3 is the WP input and pin 7 is the HOLD input.

On the ATAES132, pins 3 and 7 are not used in SPI communication mode; these pins can be tied to VCC or VSS. The state of these two pins have no impact on the functionality of the ATAES132 in the SPI communication mode. See Appendix K.2, SPI Communication Mode Pin Descriptions for the pin descriptions.
P.2.2 Write Protect (WP)

The AT25320B WP input pin inhibits all EEPROM Write operations when the WP pin is low. If WP is high, then EEPROM Write operations are allowed. The Write protect pin can be disabled by writing the WPEN bit in the STATUS Register to 0.

On the ATAES132, the User Memory Write permissions are controlled by the ZoneConfig Registers (see Appendix E.2.20, ZoneConfig Registers). The User Memory is divided into 16 user zones that are independently controlled by 16 ZoneConfig Registers; different Write permissions can be assigned to different sections of the memory. By default, all User Memory has open Write access.

P.2.3 Hold

The AT25320B HOLD input pin allows the Host to pause communication with the memory temporarily (by bringing HOLD low) and then resume the communication sequence (by bringing HOLD high). The sequence continues exactly from the point where it was paused as if there was no interruption.

The ATAES132 does not have a Hold function. If communications are interrupted, the sequence must be restarted beginning with a high-to-low transition on the CS input.

P.2.4 Page Write Operations

If the Host attempts to write data across the physical (32-byte) EEPROM page boundary, the AT25320B wraps to the beginning of the EEPROM page where the Page Write operation begins and performs the EEPROM Write after receiving a low-to-high transition on the CS input. If the Host attempts to write more than 32 bytes in a Page Write operation, then the AT25320B wraps the data at the page boundary and performs the EEPROM write after receiving a Stop condition. Partial Page Writes are supported by the AT25320B.

The ATAES132 does not allow Write operations to cross physical (32 byte) EEPROM page boundaries (see Appendix B.2, EEPROM Page Boundary and does not allow a Write operation if more than 32 data bytes are received from the Host. In both cases, the EEPROM contents remain unchanged, the data is discarded, and an error bit is set in the STATUS Register (see Appendix J.3.3, Page Write (PWRITE)). Partial Page Writes are supported by the ATAES132.

P.2.5 Read Operations

Reading beyond the end of physical memory on AT25320B causes the internal data address register to roll-over to address zero. The Read operation continues from address zero.

If an ATAES132 Read operation begins at a valid User Memory address but continues past the end of User Memory, the Read operation will not wrap to the beginning of User Memory. Reading beyond the end of User Memory causes 0xFF to be returned to the Host in reply to the Read, the internal data address register stops incrementing, and an error bit is set in the STATUS Register.

P.2.6 Read Protect

The Atmel AT25320B and other standard SPI EEPROMs do not have a Read inhibit function.

On the ATAES132, the User Memory Read permissions are controlled by the ZoneConfig registers (see Appendix E.2.20). The User Memory is divided into 16 user zones that are independently controlled by 16 ZoneConfig registers; different Read permissions can be assigned to different sections of the memory. If Read access is prohibited, then 0xFF will be returned to the Host in reply to a Read command (see Section 5.1, Read). By default, all User Memory has open Read access.
P.2.7 STATUS Register

The AT25320B STATUS Register definition is shown in Table P-1. The default state of all STATUS bits is 0b. The WPEN bit controls the Write Protect pin. Block Write protection is controlled by the BP0 and BP1 bits. If WEN = 1b, then the device is Write Enabled. If WIP = 0b, the device is ready to accept a command; WIP = 1b indicates a write cycle is in progress. The reserved bits are 0b, except when an internal write cycle is in progress. All bits of the STATUS Register are 1b when an internal write cycle is in progress.

Table P-1. AT25320B STATUS Register Definition

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>WPEN</td>
<td>Reserved</td>
<td>Reserved</td>
<td>Reserved</td>
<td>BP1</td>
<td>BP0</td>
<td>WEN</td>
<td>WIP</td>
</tr>
</tbody>
</table>

The ATAES132 STATUS Register definition is shown in Table P-2 and described in Appendix G, Understanding the STATUS Register. The default state of all STATUS bits is 0b. The WEN, WIP, and reserved bits are similar to those of standard SPI Serial EEPROM: If WEN = 1b, then the device is Write Enabled. If WIP = 0b, the device is ready to accept a command; WIP = 1b indicates a write cycle or a cryptographic operation is in progress. The reserved bits are 0b except when an internal write cycle or a cryptographic operation is in progress. All bits of the STATUS Register are 1b when an internal write cycle or a cryptographic operation is in progress.

Table P-2. ATAES132 STATUS Register definition

<table>
<thead>
<tr>
<th>Bit 7</th>
<th>Bit 6</th>
<th>Bit 5</th>
<th>Bit 4</th>
<th>Bit 3</th>
<th>Bit 2</th>
<th>Bit 1</th>
<th>Bit 0</th>
</tr>
</thead>
<tbody>
<tr>
<td>EERR</td>
<td>RRDY</td>
<td>Reserved</td>
<td>CRCE</td>
<td>Reserved</td>
<td>WAKEb</td>
<td>WEN</td>
<td>WIP</td>
</tr>
</tbody>
</table>

ATAES132 reports errors to the Host using the EERR and CRCE bits. The RRDY bit indicates if the Response Memory Buffer is empty (0b), or ready to read (1b). The WAKEb bit indicates if the device is in the sleep or standby power state. See Appendix G.1, Device Status Register (STATUS) Definition for detailed descriptions of each STATUS bit.

P.2.8 Write Status Register Command (WRSR)

The AT25320B STATUS Register contains three bits that control the Block Write Protect function and the Write Protect pin. These bits can be changed by sending a Write Status Register (WRSR) command to the memory. The ATAES132 does not support the Write Status Register (WRSR) command. The WRSR command will be ignored if it is received.

P.2.9 Block Write Protect

The AT25320B STATUS Register contains two block protect bits (BP0 and BP1) that control the Block Write Protect function. By writing the STATUS Register, the user can set the Block Protect bits to inhibit writes in ¼, ½, or the full Memory Array.

On the ATAES132, the User Memory Write permissions are controlled by the ZoneConfig registers (see Appendix E.2.20, ZoneConfig Registers). The User Memory is divided into 16 user zones that are independently controlled by 16 ZoneConfig Registers; different Write permissions can be assigned to different sections of the memory. By default, all User Memory has open Write access.
P.2.10 Standby Mode

Standard SPI EEPROMs automatically enter low-power Standby mode upon completion of any internal operation. The ATAES132 has three powered states: the Active state and two Low-Power states, the Standby state and the Sleep state. The ATAES132 will remain in the Active state between operations unless the Host sends a Sleep command to activate the Standby state or the Sleep state. The ATAES132 can also be configured to automatically enter a Low-Power state at power-up. See Appendix L, Power Management for details on the power management features.

P.2.11 Operating Voltage

The AT25320B operating voltage range is 1.8V minimum to 5.5V maximum.
The ATAES132 operating voltage range is 2.5V minimum to 5.5V maximum. See Section 9.3, DC Characteristics.

P.2.12 Maximum Operating Frequency

The AT25320B maximum SCK frequency is 10MHz when $V_{CC}$ is 2.7V to 5.5V. The maximum SCK frequency is 20MHz when $V_{CC}$ is 4.5V to 5.5V.
The ATAES132 maximum SCK frequency is 10MHz when $V_{CC}$ is 2.5V to 5.5V. See Section 9.4, AC Characteristics for AC specifications.
Appendix Q. Ordering Information

The ATAES132 production ordering codes are listed in Appendix Q.1, Atmel Ordering Codes. To increase security, ATAES132 packages are not marked with the ordering code. The ATAES132 standard packages are marked with a trace code which is unique for each manufacturing lot. Contact Atmel for additional information.

Q.1 Atmel Ordering Codes

<table>
<thead>
<tr>
<th>Atmel Ordering Code</th>
<th>Interface Configuration</th>
<th>Conditioning</th>
<th>Package</th>
<th>Lead Finish</th>
<th>Temperature Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>ATAES132-SH-EQ</td>
<td>SPI</td>
<td>Bulk$^{(1)}$</td>
<td>8S1</td>
<td>NiPdAu Lead-free/Halogen-free</td>
<td>Industrial Temperature (-40°C to 85°C)</td>
</tr>
<tr>
<td>ATAES132-SH-ER</td>
<td>I²C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATAES132-SH-EQ-T</td>
<td>SPI</td>
<td>Tape and Reel$^{(2)}$</td>
<td>8MA2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATAES132-SH-ER-T</td>
<td>I²C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATAES132-MAH-EQ-T</td>
<td>SPI</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>ATAES132-MAH-ER-T</td>
<td>I²C</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Notes: 1. Bulk
   - SOIC = 100 per tube
2. Tape and Reel (-T)
   - SOIC = 4K per reel
   - UDFN = 5K per reel

<table>
<thead>
<tr>
<th>Package Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>8S1</td>
</tr>
<tr>
<td>8MA2</td>
</tr>
</tbody>
</table>
Q.2 Mechanical Information

Q.2.1 8S1 — 8-lead JEDEC SOIC

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

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MIN</th>
<th>NOM</th>
<th>MAX</th>
<th>NOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1.35</td>
<td>–</td>
<td>1.75</td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td>0.10</td>
<td>–</td>
<td>0.25</td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>0.31</td>
<td>–</td>
<td>0.51</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0.17</td>
<td>–</td>
<td>0.25</td>
<td></td>
</tr>
<tr>
<td>D</td>
<td>4.80</td>
<td>–</td>
<td>5.05</td>
<td></td>
</tr>
<tr>
<td>E1</td>
<td>3.81</td>
<td>–</td>
<td>3.99</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>5.79</td>
<td>–</td>
<td>6.20</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>1.27</td>
<td>–</td>
<td></td>
<td>BSC</td>
</tr>
<tr>
<td>L</td>
<td>0.40</td>
<td>–</td>
<td>1.27</td>
<td></td>
</tr>
<tr>
<td>Ø</td>
<td>0°</td>
<td>–</td>
<td>8°</td>
<td></td>
</tr>
</tbody>
</table>

Package Drawing Contact: packagedrawings@atmel.com

6/22/11

TITLE
8S1, 8-lead (0.150” Wide Body), Plastic Gull Wing Small Outline (JEDEC SOIC)

G
8S1
G

ATAES132 [Datasheet]
**Q.2.2 8MA2 — 8-pad UDFN/USON**

**COMMON DIMENSIONS**

(Unit of Measure = mm)

<table>
<thead>
<tr>
<th>SYMBOL</th>
<th>MIN</th>
<th>NOM</th>
<th>MAX</th>
<th>NOTE</th>
</tr>
</thead>
<tbody>
<tr>
<td>D</td>
<td>1.90</td>
<td>2.00</td>
<td>2.10</td>
<td></td>
</tr>
<tr>
<td>E</td>
<td>2.90</td>
<td>3.00</td>
<td>3.10</td>
<td></td>
</tr>
<tr>
<td>D2</td>
<td>1.40</td>
<td>1.50</td>
<td>1.60</td>
<td></td>
</tr>
<tr>
<td>E2</td>
<td>1.20</td>
<td>1.30</td>
<td>1.40</td>
<td></td>
</tr>
<tr>
<td>A</td>
<td>0.50</td>
<td>0.55</td>
<td>0.60</td>
<td></td>
</tr>
<tr>
<td>A1</td>
<td>0.0</td>
<td>0.02</td>
<td>0.05</td>
<td></td>
</tr>
<tr>
<td>A2</td>
<td>–</td>
<td>–</td>
<td>0.55</td>
<td></td>
</tr>
<tr>
<td>C</td>
<td>0.152</td>
<td></td>
<td></td>
<td>REF</td>
</tr>
<tr>
<td>L</td>
<td>0.30</td>
<td>0.35</td>
<td>0.40</td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>0.18</td>
<td>0.25</td>
<td>0.30</td>
<td>3</td>
</tr>
</tbody>
</table>

Notes:
1. This drawing is for general information only. Refer to JEDEC Drawing MO-229, for proper dimensions, tolerances, datums, etc.
2. The terminal #1 ID is a laser-marked feature.
3. Dimension b applies to metallized terminal and is measured between 0.15 mm and 0.30 mm 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.

---

**Atmel**

Package Drawing Contact: packagedrawings@atmel.com

---

**ATAES132 [Datasheet]**
Atmel-8780C-CryptoAuth-ATAES132-Datasheet_102013
Appendix R. Errata

R.1 KeyCreate Command Executed with Usage Counter

If the KeyCreate command is executed with Mode bit 2 set to 1b and a Key with a Usage Counter attached to it is used, the Usage Counter will not be incremented.
## Appendix S. Revision History

<table>
<thead>
<tr>
<th>Doc. Rev.</th>
<th>Date</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>8760C</td>
<td>10/2013</td>
<td>Remove TempCal and TempOffset registers. Update Key Management section and Update KeyCompute to KeyCreate command. Replace ExportOK VolUsage Field to reserved and ExportAuth with ChildAuth. Remove KeyCompute Command, KeyCompute: KeyImport MAC, KeyCompute: KeyLoad MAC, and KeyImport Command: KeyCompute MAC sections. Update disclaimer page.</td>
</tr>
<tr>
<td>8760A</td>
<td>05/2011</td>
<td>Initial document release.</td>
</tr>
</tbody>
</table>