ECC608-TMNGTLS CryptoAuthentication™ Data Sheet

ECC608-TMNGTLS

www.microchip.com Product Pages: ECC608-TMNGTLS

Introduction

The ECC608-TMNGTLS is a Trust Manager pre-provisioned variant of the ATECC608. The ECC608-TMNGTLS device will work in combination with the keySTREAM™ Software-as-a-Service (SaaS) powered by Kudelski IoT. The device comes pre-provisioned with a set of cryptographic keys to connect to keySTREAM SaaS. When deployed in the marketplace, the IoT device containing the ECC608-TMNGTLS will connect to the keySTREAM SaaS, which will give ownership of the IoT device to the intended owner by provisioning the device "in-field" with its custom PKI. The solution is truly zero touch as there is no need to physically intervene in a secret exchange. Your cryptographic keys are now managed remotely and dynamically while being protected within the physical boundary of the secure authentication IC. The solution ensures security practices for the lifecycle management of your product related to key management after the IoT device is deployed.

This data sheet provides the slot and key configuration information for the ECC608-TMNGTLS device. This information defines the access policies of each of the data zone slots. Limited command and I/O operating information is provided in this document. Specific sections discussing Microchip's hardware and software tools that can aid in developing user applications have been included. Guided use cases can be leveraged in Microchip's Trust Platform Design Suite (TPDS). The required secrets and private keys are provisioned in-field through the keySTREAM SaaS.

Features

- Fully Specified Configuration Zone
- I^2C Interface with One-Time Changeable I^2C Address
- JIL High Rating – Validated to JIL Application of Attack Potential to Smartcards and Similar Devices, Version 3.1
- Internal High-Quality NIST SP 800-90A/B/C True Random Number Generator (TRNG) NIST CMVP ESV Certified
- Predefined Slot Access Policies to Work with Kudelski keySTREAM SaaS:
  - ECC P-256 Device Identity Key
  - ECC P-256 Attestation Private Key
  - Customer specific identity information
- Slots for In-Field Provisioning of:
  - Device and Signer compressed certificate slots
- 1.8V to 5.5V I/O Levels, 2.0V to 5.5V Supply Voltage
- Standard Industrial Temperature Range: -40℃ to +85℃
- 30 nA nominal Sleep Current
- Available in 8-Pad UDFN and 8-Pin SOIC packages with a fixed 10-unit reel size for prototyping and a 2k Minimum-Order-Quantity (MOQ) for production

Use Cases

- Secure IoT TLS 1.2 and 1.3 Connections including custom Root CA setup and associated PKI

- Dynamic Certificate management including rotation, revocation, renewal
- Secure Boot
- End-to-End Data Protection
- Private Key Rotation for security agility

Table of Contents

Introduction....1

Features....1

Use Cases....1

1. Pin Configuration and Pinouts....6

2. The Trust Manager Experience....7

3. EEPROM Memory and Data Zone Access Policies....8
   - 3.1. ECC608-TMNGTLS Configuration Zone....9
   - 3.2. Data Zone and Access Policies....10
     - 3.2.1. Data Zone Data Types....10
       - 3.2.1.1. Private Keys....10
       - 3.2.1.2. Public Keys....10
       - 3.2.1.3. Certificate Storage....10
       - 3.2.1.4. Secure Boot....11
       - 3.2.1.5. Secret Key....12
       - 3.2.1.6. I/O Protection Key....12
       - 3.2.1.7. General Data Storage....12
     - 3.2.2. Slot Configuration Terminology....12
     - 3.2.3. ECC608-TMNGTLS Slot Configuration Summary....14
     - 3.2.4. ECC608-TMNGTLS Detailed Slot Access Policies....14
   - 3.3. ECC608-TMNGTLS EEPROM One Time Programmable (OTP) Zone....18

4. Static RAM (SRAM) Memory....19

5. General Command Information....20
   - 5.1. I/O Transactions....20
   - 5.2. Command Packets....20
   - 5.3. Status/Error Codes....21
   - 5.4. Address Encoding....22
     - 5.4.1. Configuration Zone Addressing....22
     - 5.4.2. OTP Zone Addressing....22
     - 5.4.3. Data Zone Addressing....22
   - 5.5. Formatting of Keys, Signatures and Certificates....23
     - 5.5.1. ECC Key Formatting....23
       - 5.5.1.1. Public Key Formats....24
     - 5.5.2. Signature Format....25
     - 5.5.3. Certificate Storage....25

6. Device Commands....27
   - 6.1. General Device Commands....28
     - 6.1.1. `Counter` Command....28
     - 6.1.2. `Info` Command....28
     - 6.1.3. `Lock` Command....29
     - 6.1.4. `Nonce` Command....29
     - 6.1.5. `Random` Command....29

6.1.6. `Read` Command....29
6.1.7. `SelfTest` Command....29
6.1.8. `SHA` Command....29
6.1.9. `UpdateExtra` Command....29
6.1.10. `Write` Command....30
6.2. Asymmetric Cryptography Commands....30
6.2.1. `ECDH` Command....30
6.2.2. `GenKey` Command....30
6.2.3. `SecureBoot` Command....30
6.2.4. `Sign` Command....30
6.2.5. `Verify` Command....30
6.3. Symmetric Cryptography Commands....31
6.3.1. `AES` Command....31
6.3.2. `CheckMac` Command....31
6.3.3. `GenDig` Command....31
6.3.4. `KDF` Command....32
6.3.5. `MAC` Command....32

7. Application Information....33
7.1. Use Cases....33
7.2. Development Tools....34
7.2.1. Trust Platform Design Suite....35
7.2.2. Hardware Tools....35
7.2.3. CryptoAuthLib....36

8. I^2C Interface....37
8.1. I/O Conditions....37
8.1.1. Device is Asleep....37
8.1.2. Device is Awake....38
8.2. I^2C Transmission to ECC608-TMNGTLS....39
8.2.1. Word Address Values....40
8.2.2. I^2C Synchronization....41
8.3. I^2C Transmission from the ECC608-TMNGTLS....41
8.4. Sleep Sequence....42
8.5. Idle Sequence....42

9. Electrical Characteristics....43
9.1. Absolute Maximum Ratings....43
9.2. Reliability....43
9.3. AC Parameters: All I/O Interfaces....43
9.3.1. AC Parameters: I^2C Interface....45
9.4. DC Parameters: All I/O Interfaces....46
9.4.1. V_{IH} and V_{IL} Specifications....46

10. Package Drawings....48
10.1. Package Marking Information....48
10.2. 8-Pad UDFN....49
10.3. 8-Lead SOIC....52

11. Product Identification System....55

12. Revision History....56

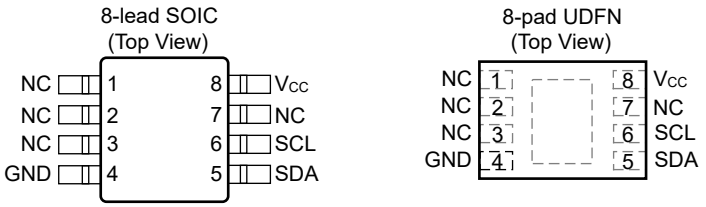
Microchip Information....57
Trademarks....57
Legal Notice....57
Microchip Devices Code Protection Feature....57

1. Pin Configuration and Pinouts

Table 1-1. Pin Configuration

Pin	Function
NC	No Connect
GND	Ground
SDA	I^2C Serial Data
SCL	I^2C Serial Clock Input
V_{CC}	Power Supply

Figure 1-1. UDFN and SOIC Pinout



Note: The UDFN backside paddle is recommended to be connected to GND.

Related Links

8-Pad UDFN

8-Lead SOIC

2. The Trust Manager Experience

The ECC608-TMNGTLS secure authentication IC is focused on making hardware security for IoT products a self-served experience while maintaining a high level of security throughout the entire lifecycle of a project.

What keySTREAM SaaS Provides

The keySTREAM SaaS service provides the necessary HSM infrastructure to set up a root CA and associated PKI, as well as provision it in-field, along with TLS cryptographic keys, data and certificates into the secure boundary of the ECC608-TMNGTLS. These cryptographic credentials are, now, actively managed for the whole product life cycle. The solution allows for:

- The management of certificates to ensure secure authentication to any cloud platforms. The certificates can be updated or rotated to ensure a certificate expiration date will not disconnect an IoT product. The end result lowers device management cost and complexity.
- The Infrastructure Agnostic SaaS (IaaS) works with AWS® and Microsoft Azure® out of the box.
- Cost-effective hosting of the cryptographic keys in keySTREAM HSMs along with associated availability and maintenance

The keySTREAM SaaS offering provides a space-efficient embedded library (keySTREAM Trusted Agent KTA) that is capable of fitting in a memory-constrained MCU all the way to MPU products. This allows security to be implemented across a wide complexity range of devices without paying a significant memory penalty. The tools are tied into a telecom-grade, cloud-hosted platform designed to scale to the wide range of needs for IoT products.

Trust Manager Flow

Getting started with the ECC608-TMNGTLS was implemented to be a relatively simple process.

1. Sign up with a Kudelski IoT account to gain access to keySTREAM Security Management Services. Provide the unique information associated with your company and your project to ensure that your product will have a unique identity that cannot be cloned. Following the autoclaim process in keySTREAM SaaS, make sure to record the email address of the purchaser used in the Microchip ordering system.
2. Order ECC608-TMNGTLS devices. Microchip provides 10 unit sample packs to make the cost of entry to evaluate devices low. These devices are available off-the-shelf and do not require any manual provisioning prior to purchasing. The keySTREAM SaaS is capable of validating a specific device to see if it was provisioned for keySTREAM SaaS remote management by Microchip
3. Use the keySTREAM SaaS to create a custom ROOT CA and associated PKI.
4. Use the keySTREAM SaaS services to claim the devices purchased through MicrochipDirect. In-field provisioning comes later, when the embedded system connects to Kudelski keySTREAM SaaS. The keySTREAM SaaS services allow for the creation of a custom PKI in a hosted and managed HSM, ability to enter unique customer information, creation of certificates and the ability to create I/O protection keys for secure communication.
5. Deploy the product in the field.
6. Use the in-field management capability to rotate keys and/or update or rotate certificates.

3. EEPROM Memory and Data Zone Access Policies

The EEPROM memory contains a total of 1,400 bytes and is divided into the following zones:

Table 3-1. ECC608-TMNGTLS EEPROM Zones

Zone	Description	Nomenclature
Configuration	Zone of 128 bytes (1,024 bits) EEPROM that contains: • Device configuration • Slot access policy information • Counter values • Device serial number • Lock information The LockConfig byte is already set. Nothing can be directly written to this zone. The zone can always be read.	Config[a:b] = A range of bytes within a field of the Configuration zone
Data	Zone of 1,208 bytes (9.7 Kb) split into 16 general purpose read-only or read/write memory slots. The access policy information defined by the Configuration zone bytes determines how each slot can be accessed. The access policy for each data slot in the ECC608-TMNGTLS device is set and the slot access policies defined by the Configuration zone are in full effect.	Slot[YY] = The entire contents stored in Slot YY of the Data zone
One-Time-Programmable (OTP)	Zone of 64 bytes (512 bits) arranged into two blocks of 32 bytes each. For the ECC608-TMNGTLS, the zone is preloaded with a predefined value. This zone cannot be modified but can be read at any time. See ECC608-TMNGTLS EEPROM One Time Programmable (OTP) Zone for more information.	OTP[bb] = A byte within the OTP zone, while OTP[aa:bb] indicates a range of bytes

Table 3-2. Document Terms

Terms discussed within this document will have the following meanings:

Term	Meaning
Block	A single 256-bit (32-byte) area of a particular memory zone. The industry standard SHA-256 documentation also uses the term block to indicate a 512-bit section of the message input. Within this document, this convention is used only when describing hash input messages.
KeyID	KeyID is equivalent to the slot number for those slots designated to hold key values. Key 1 (sometimes referred to as key[1]) is stored in Slot[1] and so on. While all 16 slots can potentially hold keys, those slots that are configured to permit clear text reads are not normally used as private or secret keys by the crypto commands.
mode[b]	Indicates bit, b, of the parameter mode.
SRAM	Contains input and output buffers, as well as internal state storage locations. This memory is not directly accessible by the user.
Word	A single 4-byte word of data read from or written to a block. The word is the smallest unit of data access.
LSB/MSB	Least Significant Byte/Most Significant Byte.
LSb/MSb	Least Significant Bit/Most Significant Bit.

Related Links

ECC608-TMNGTLS Configuration Zone
Data Zone and Access Policies
ECC608-TMNGTLS EEPROM One Time Programmable (OTP) Zone

3.1 ECC608-TMNGTLS Configuration Zone

The ECC608-TMNGTLS configuration is fixed and cannot be modified by the customer. Relevant information about how the device is configured is shown below or in the slot information. The keys are provisioned in-field by the keySTREAM SaaS.

Device Configuration Information

- The serial number for each device is unique and stored in Bytes[0:3, 8:12]. Bytes[0:1] are 0x01 0x23 and Byte[8] is 0x01. All other bytes are unique.
- The default 7-bit I^2C address is 0x38. The I^2C 8-bit address byte used for writes/reads is 0x70 and 0x71, respectively. The I^2C address can be overwritten using the `UpdateExtra` command.

> **Important:** The default I^2C address of the ECC608-TMNGTLS differs from that of the standard ECC608 Device.

- All data slots, except the Secure Boot Public Key (Slot #15), the Secure Boot Digest (Slot #13) and the I/O Protection (Slot #7), are managed by Kudelski.
- AES128 operations are enabled.
- ChipMode Features
  - Clock divider is set for maximum execution time speed.
  - The Watchdog Timer (WDT) is set to a maximum time-out of 1.3s.
  - The I/O levels are set to a fixed reference level; therefore, the host processor can operate at a lower voltage than the ECC608-TMNGTLS device.
  - Setting an alternate I^2C address is enabled.
- SecureBoot Features
  - FullStore digest mode is enabled.
  - Secure Boot ECC P-256 public key is stored in Slot 15 and loaded at manufacturing time.
  - Secure Boot digest is stored in Slot 13.
  - A random nonce is not required but is recommended.
  - Secure Boot persistent latch is disabled.
  - I/O protection key to encrypt the output of the command
- Chip Options Features
  - Use of the I/O protection key is enabled with the key stored in Slot 7.
  - Output of the KDF function will be in the clear but can be encrypted based on the mode of the `KDF` command.
  - KDF AES mode is enabled.
  - Output of the ECDH master secret will be in the clear but can be encrypted based on the mode of the `ECDH` command.
  - The Health Test Failure bit is cleared after any time that a command fails as a result of a health test failure. If the failure symptom is transient, the command will pass when run a second time.
  - Power on self tests are disabled. Self tests must be explicitly run if so desired.
  - The Health Test Failure bit is cleared after any time that a command fails as a result of a health test failure. If the failure symptom is transient, the command will pass when run a second time.
- Monotonic counters are available for use by the system and are not attached to any keys.

- The Configuration Zone is permanently locked prohibiting the update of the configuration.

Related Links
ECC608-TMNGTLS Slot Configuration Summary

3.2 Data Zone and Access Policies

The following sections describe the detailed access policy information associated with each slot. The actual access policy information is stored within the Slot and Key configuration sections in the EEPROM Configuration zone. Each Data zone slot has two Slot Configuration bytes and two Key Configuration bytes associated with it. Together, these four bytes create the access policies for each slot. The actual type of data stored within the slot is determined by the access policies for that slot.

3.2.1 Data Zone Data Types

The following section provides more details on the various types of data capable of being stored in the ECC608-TMNGTLS data slots.

3.2.1.1 Private Keys

ECC P-256 private keys are the fundamental building blocks of ECC Security for the ECC608-TMNGTLS. These keys are private and unique to each device and cannot be read. ECC private keys are randomly generated by the secure element's TRNG and are securely held in slots configured as ECC private keys.

Primary Private Key

This is the primary authentication key and is stored in Slot 0. Each device has its own unique private key. The key can be modified unless Slot 0 is locked.

This key is enabled for two primary elliptic curve functions:

- ECDSA Sign for authentication
- ECDH for key agreement. If encryption of the ECDH output is required, the I/O protection key needs to be set up first. See I/O Protection Key for setup details.

This private key is the foundation for the generation of the corresponding public key and the X.509 Certificates.

Key Attestation

The private key in Slot 1 is configured as an internal sign-only key, which means it can only sign messages generated internally by the `GenKey` or `GenDig` commands and cannot be used to sign arbitrary external messages. This feature allows the internal sign key to be used to attest to what keys are in the device and their configuration/status to any system that knows (and trusts) the internal sign public key.

3.2.1.2 Public Keys

Public keys are associated with ECC private keys. Every ECC private key will have its own unique public key. Public keys associated with private keys stored on the device can be retrieved by using the `GenKey` command. Public keys from an ECC private key stored off chip can be stored in the device in a slot configured for a public key.

For the ECC608-TMNGTLS public keys have been stored in Slots 11, 14 and 15.

3.2.1.3 Certificate Storage

The ECC608-TMNGTLS storage is centered around securely holding keys. X.509 certificates tend to be larger than what will fit into a single ECC608-TMNGTLS device slot; therefore, a compressed format is used. This technique may be better called a partial certificate as it stores dynamic certificate information on the device and imposes some limitations. Dynamic information is certificate content that can be expected to change from device to device (e.g., public key, validity dates, etc.). Firmware is expected to have a certificate definition with a template of the full X.509

Certificate containing static information (data that are the same for all certificates) and instructions on how to rebuild the full certificate from the dynamic information in the compressed certificate.

The following application note documents the compressed certificate format: ATECC Compressed Certificate Definition.

The CryptoAuthLib library also contains APIs for working with compressed certificates.

Device Certificate

The device certificate consists of information associated with the actual end unit. For the ECC608-TMNGTLS, the compressed device certificate is stored in Slot #10.

Signer Certificate

The signer certificate consists of the information associated with the signer certificate authority used to sign the device certificate. For the ECC608-TMNGTLS, the compressed signer certificate is stored in Slot #12. The signer public key is also required to rebuild the full signer certificate.

Signer Public Key

The signer public key is the public key needed to verify the signer and the information that is associated with the signer compressed certificate. For the ECC608-TMNGTLS, it is stored in Slot #11.

The following table shows all the slots associated with certificates in the ECC608-TMNGTLS:

Slot	Description
0	Primary private key. The public key can be generated at any time using the `GenKey` command in Mode = `0x00`.
10	Device certificate. This is stored here in a compressed format.
11	Signer public key.
12	Signer certificate. This is stored in a compressed format.

For the ECC608-TMNGTLS, these slots can be overwritten unless the individual slots are locked.

Related Links

ECC608-TMNGTLS Configuration Zone
ECC608-TMNGTLS Detailed Slot Access Policies
Certificate Storage

3.2.1.4 Secure Boot

The `SecureBoot` command is enabled for the ECC608-TMNGTLS. This allows the system to cryptographically validate its firmware via a boot loader before performing a full boot. This functionality can also be used to validate new firmware images before they are loaded.

The secure boot feature requires establishing an ECC P-256 firmware signing key before it can be used. The private key will be held by the firmware developers for signing the firmware image. The public key will be written to the secure boot public key slot and the slot will be locked to make it permanent during manufacturing at the contract manufacturer site.

To implement the SecureBoot, several data slots are required.

Secure Boot Digest

The Secure Boot Digest is a 32-byte SHA-256 digest calculated over the firmware application code. This digest needs to be updated every time the firmware is updated. For the ECC608-TMNGTLS, the digest is stored in Slot 13.

Secure Boot Public Key

The Secure Boot public key is used to perform a verify function to validate the Secure Boot Digest and signature. The Secure Boot public key is stored in Slot 15.

I/O Protection Key

The I/O protection key is stored in Slot 7 and can be used to protect the `SecureBoot` command output.

Related Links

ECC608-TMNGTLS Configuration Zone
ECC608-TMNGTLS Detailed Slot Access Policies
SecureBoot Command

3.2.1.5 Secret Key

Slot 5 can be used to store a 32-byte Secret Key. This key can be used with the ECC608-TMNGTLS's symmetric key commands (`GenDig`, `MAC`, `CheckMac`, `KDF`, `SHA/HMAC`, `AES`).

This key must be written with an encrypted write using the encryption key stored in slot 6.

3.2.1.6 I/O Protection Key

The `Verify`, `ECDH`, `SecureBoot` and `KDF` commands can optionally use the I/O protection feature to encrypt some parameters and validate (via MAC) some responses. This is to help protect against man-in-the-middle attacks on the physical I^2C bus. During Secure Boot, the I/O protection key can also be useful in preventing replay attacks. However, before this feature can be used, the MCU and ECC608-TMNGTLS need to generate and save a unique I/O protection key, essentially pairing the MCU and ECC608-TMNGTLS devices to each other. The pairing process must happen at the customer production facility.

I/O protection key generation:

1. MCU uses a random command to generate a random 32-byte I/O protection key.
2. MCU saves the I/O protection key in its internal Flash.
3. MCU writes the I/O protection key to the I/O protection key slot.
4. MCU slot locks that slot to make the I/O protection key permanent.

As a pairing check, the MCU could use the `MAC` command to issue a challenge to the I/O protection key and verify that the I/O protection key stored in Flash matches the one in the ECC608-TMNGTLS.

3.2.1.7 General Data Storage

Slot 15 can be used as general purpose data storage provided Secure Boot is not implemented and this slot does not contain the Secure Boot Public Key. This slot may be used to store any data that are allowed to be publicly accessible and can always be read and written in the clear.

3.2.2 Slot Configuration Terminology

The following section provides a set of terms used to discuss configuration options. The terms are arranged alphabetically.

Term	Description
AES Key	Slot can be used as a key source for `AES` commands. The AES key is 128 bits in width for the ECC608-TMNGTLS.
Always Write	Slot can be written in the clear with the `Write` command.
Clear Read	Slot is considered public (non-secret) and its contents can be read in the clear with the `Read` command.
ECDH	Elliptic Curve Diffie Hellman. Private key can be used with the `ECDH` command.
Ext Sign	Private key can be used to sign external (arbitrary) messages.
Int Sign	Private key can be used to sign internal messages generated by the `GenKey` or `GenDig` commands. Used to attest to the device's internal keys and configuration.
Lockable	Slot can be locked at some point in the future. Once locked, the slot contents cannot be changed (read/use only).

Term	Description
No Read	Slot is considered secret and its contents cannot be read with the `Read` command. Private keys and symmetric secrets must always be configured as No Read.
No Write	Slot cannot be changed with the `Write` command.
Permanent	Private key is permanent/unchangeable. It is internally generated during factory provisioning.
Updatable	Private key can be overwritten later with a new, random, internally-generated private key. Its initial value is internally generated during factory provisioning.

3.2.3 ECC608-TMNGTLS Slot Configuration Summary

The ECC608-TMNGTLS has 16 slots that are configured for specific purposes as part of the Trust Manager solution. The slots are divided into several types and are specified in the Slot Type column in the table below. The following slot types are defined:

- K = keySTREAM SaaS slots for use by Kudelski only
- P = Application slots provisioned through keySTREAM SaaS. This information is used by the customer application but is under the control of the keySTREAM SaaS.
- C = Customer Provisioned Slot. This information is known only to the customer and is updated in the customer's production line.
- R = Reserved. Not in use at the time this data sheet was created but may be used for future applications.

Slot	Slot Type	Key Name	Description
0	P	Device Identity Key	Primary Private key used to sign device compressed certificate stored in Slot 10.
1	K	Attestation Key	Private Key used by Kudelski to attest to the device authenticity. Only Kudelski knows the corresponding Public Key.
2	K	Seal Identity ID	Unique profile User ID (UID) stored in this slot.
3	K	Asymmetric Key	Managed and reserved by Kudelski.
4	K	Symmetric Key	Managed and reserved by Kudelski.
5	P	Symmetric Key for in-field provisioning	Slot to hold customer symmetric key. This slot can ONLY be updated by keySTREAM™ SaaS. The slot is set as encrypted write and WriteKey is in Slot 6.
6	R	Encrypt Write Key	Encrypted WriteKey for Slot 5 and Slot 14, provisioned at MCHP facility. The Parent Key is only known to Kudelski. The stored key is diversified.
7	C	IO Protection Key	Slot to hold customer I/O protection key. This slot is updated by the customer in their production line.
8	K	keySTREAM SaaS Onboarding Data	Kudelski-specific data used for multiple keySTREAM SaaS operations.
9	R	Symmetric Key	Managed and reserved by Kudleski.
10	P	Device Compressed Certificate	Compressed Device Certificate.
11	P	Signer Public Key	Public Key associated with the Kudelski signer.
12	P	Signer compressed Certificate	Kudelski Compressed Signer Certificate.
13	P	Secure Boot Digest	Slot to hold the Secureboot Digest (Stored digest mode). Can only be updated internally using Secure boot commands.
14	P	Public Key for in-field provisioning.	Slot to hold the customer-specific Public Key. This slot can only be updated by keySTREAM SaaS. The slot is set as encrypted write and WriteKey is in slot 6.
15	P	Secure Boot public key or C-Data	Slot to hold Customer SecureBoot public key. The slot is expected to be provisioned at the customer's production line.

Related Links

ECC608-TMNGTLS Detailed Slot Access Policies

3.2.4 ECC608-TMNGTLS Detailed Slot Access Policies

The ECC608-TMNGTLS slot access policies for each data slot are completely configured. Each slot has a designated purpose. In most cases, all slots are required for proper operation and security for a given use case. The slots are divided into K-slots, which are primarily used by Kudelski as part of the keySTREAM SaaS operational environment. The C-slots are for specific use by the customer.

Slot Locking Options

Slot locking options are called out for each individual slot and will be one of two types.

Slot Lockable	A slot that has the slot lock option set allows for the end user to lock the slot at some point in the future after the initial manufacturing phase. This can be used to allow for a key or data to be set during a subsequent manufacturing step outside of Microchip or by the end user. The slot can be locked using the `Lock` command. Once the slot is locked, no future modifications to the data in the slot are possible.
Permanent Lock	A permanently locked slot cannot be updated once it leaves the Microchip manufacturing facilities. The correct data or key must be provided to Microchip prior to the provisioning of these devices.

Detailed Slot Configurations

The following tables provide a more detailed description of the slot configuration and key configuration settings for each configured slot on the device. Relevant information for the key and slot configuration are provided as part of the Description of Enabled Features. The specific key type is shown in () after the key name. Key Type Descriptions can be found in the ECC608-TMNGTLS Slot Configuration Summary.

Table 3-3. Slot 0: Device Identity Key (P)

Slot	Configuration Value	Description of Enabled Features
0	Key:	• Contains the P256 NIST ECC private key. • The private key can be generated. • The Public Key can be generated from the private key. • Random nonce is required before commands are run. • Slot can be individually locked.
	Slot:	• Slot is secret. • Can be used to sign external messages. • Can be used to generate a session key with `ECDH` command.

Table 3-4. Slot 1: Attestation Key (K)

Slot	Configuration Value	Description of Enabled Features
1	Key:	• Contains P256 NIST ECC private key. • Key is written at time of provisioning. • The corresponding public key can never be generated.
	Slot:	• Slot is secret. • Can sign messages internally generated by the `GenDig` or `GenKey` command.

Table 3-5. Slot 2: Seal Identity ID (K)

Slot	Configuration Value	Description of Enabled Features
2	Key:	• Contains data used to identify the user and will contain the device profile UID. • Slot can be individually locked.
	Slot:	• Clear text reads are always permitted. • Clear text writes are always permitted.

Table 3-6. Slot 3: Asymmetric Key (K)
Note: This slot is only for use only by Kudelski IoT.

Slot	Configuration Value	Description of Enabled Features
3	Key:	• Contains P256 NIST ECC private key. • The corresponding public key can always be generated. • Random nonce is required. • Slot can be individually locked.
	Slot:	• GenKey can be used to generate a new ECC private key in this slot prior to locking. • Slot is secret. • Can sign external messages. • Can be used to generate a session key with ECDH command.

Table 3-7. Slot 4: Symmetric Key (K)
Note: This slot is only for use only by Kudelski IoT.

Slot	Configuration Value	Description of Enabled Features
4	Key:	• Slot will contain an AES session key. • Slot holds the output of the ECDH command using the key stored in Slot 3. • Slot is not individually lockable.
	Slot:	• Slot is secret and cannot be read. • Slot can be directly written.

Table 3-8. Slot 5: Symmetric Key for In-field Provisioning (R)

Slot	Configuration Value	Description of Enabled Features
5	Key:	• Contains an AES Key. • A random nonce is always required.
	Slot:	• Contents are secret and can never be read. • Encrypted data can be written to this slot using the key in Slot 6. • Encrypted write requires a MAC.

Table 3-9. Slot 6: Encrypt Write Key (R)

Slot	Configuration Value	Description of Enabled Features
6	Key:	• Contains a key for encrypting data. • A random nonce is required when this key is used. • This slot is permanently locked.
	Slot:	• Reads are not permitted from this slot. • Writes are never permitted to this slot.

Table 3-10. Slot 7: IO Protection Key (C)

Slot	Configuration Value	Description of Enabled Features
7	Key:	• Contains a key for encrypting I/O data. • A random nonce is required prior to using this key. • Slot can be individually locked.
	Slot:	• This slot cannot be read. • Clear text writes are permitted to this slot.

⚠ CAUTION In general, the I/O protection key stored in Slot 7 must be left to be Slot Lockable. In most cases, the I/O protection key is often unique to each device. If, for some use case, the I/O protection key is the same for all devices, a Permanent Lock Option can be selected.

Table 3-11. Slot 8: keySTREAM™ SaaS Onboarding Data (K)

Slot	Configuration Value	Description of Enabled Features
8	Key:	• Data slot reserved for use by Kudelski to onboard customers. • Slot is lockable.
	Slot:	• Clear text reads are permitted from this slot. • Clear Text Writes are allowed to this slot if not locked.

Table 3-12. Slot 9: Symmetric Key

Slot	Configuration Value	Description of Enabled Features
9	Key:	• Slot can store up to four AES 128-bit symmetric keys.
	Slot:	• Slot is secret and Keys can not be read. • Clear text writes are allowed to this slot.

Table 3-13. Slot 10: Device Compressed Certificate (P)

Slot	Configuration Value	Description of Enabled Features
10	Key:	• Slot defined to store other data. • Slot can be individually locked.
	Slot:	• Data can always be read in the clear. • Data can be written in the clear unless slot is locked.

Table 3-14. Slot 11: Signer Public Key (P)

Slot	Configuration Value	Description of Enabled Features
11	Key:	• P256 NIST ECC Public key associated with Signer Certificate stored in this slot. • Slot can be locked.
	Slot:	• Data can always be read in the clear. • Data can be written in the clear unless the slot is locked.

Table 3-15. Slot 12: Signer Compressed Certificate (P)

Slot	Configuration Value	Description of Enabled Features
12	Key:	• Slot defined to store other data. • Slot can be individually locked.
	Slot:	• Data can always be read in the clear. • Data can be written in the clear unless the slot is locked.

Table 3-16. Slot 13: Secure Boot Digest (P)

Slot	Configuration Value	Description of Enabled Features
13	Key:	• Slot is defined to hold the Secure Boot Digest. • Slot cannot be individually locked.
	Slot:	• Data cannot be read from this slot. • Data cannot be directly written to the slot. The Digest can be stored using the Secure Boot command.

Table 3-17. Slot 14: Public Key for in-field programming. (R)

Slot	Configuration Value	Description of Enabled Features
14	Key:	• P256 NIST ECC customer Public key associated for use in in-field programming. • Slot is unlocked at time of provisioning.
	Slot:	• Data can always be read in the clear. • Data can be written to the slot using an encrypted write using the key in Slot 6. • Encrypted write must include a MAC.

Table 3-18. Slot 15: Secure Boot Public key or C-Data (P)

Slot	Configuration Value	Description of Enabled Features
15	Key:	• P256 NIST ECC customer Public key for validating secure boot operation with the `Verify` command. • Slot can be individually locked.
	Slot:	• Data can always be read in the clear. • Data can be written in the clear unless the slot is locked.

Related Links

Data Zone Data Types

The following section provides more details on the various types of data capable of being stored in the ECC608-TMNGTLS data slots.

3.3 ECC608-TMNGTLS EEPROM One Time Programmable (OTP) Zone

The OTP zone of 64 bytes (512 bits) is part of the EEPROM array and is used for read-only storage. It is organized as two blocks of 32 bytes each. For the ECC608-TMNGTLS, the OTP zone is shipped pre-locked and contains the following data:

```
02 25 70 13 37 B9 CF F0 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
```

The data byte values written into the OTP zone are always available for reading using either 4 or 32-byte reads but can never be modified.

NOTICE The bytes in the OTP zone may change over time. These values are recommended to not be used in any cryptographic calculations.

4. Static RAM (SRAM) Memory

The device also includes an SRAM array that is used to store the input command or output result, nonces, intermediate computation values, ephemeral keys, the SHA context, etc. The contents of the SRAM can never be read directly; only used internally by the secure element. The entire contents of this memory are invalidated whenever the device goes into Sleep mode or the power is removed.

The SRAM Array consists of the following buffers:

TempKey

TempKey is the primary storage register in the SRAM array that can be used to store various intermediate values generated by various commands. TempKey is 64 bytes long and is divided into an upper and lower half. The contents of this register can never be read from the device (although the device itself can read and use the contents internally).

Message Digest Buffer

The Message Digest Buffer is a 64-byte register that is used to convey the input message digest to the `Verify` and `Sign` commands when the TempKey register is needed to retain different information. The `SHA` command can write a digest directly to this register to simplify external host programming.

Alternate Key Buffer

The Alternate Key Buffer is a 32-byte register that can be used by the `KDF` command to store keys when the TempKey register is needed to retain different information. It can be written to a fixed input value by the `Nonce` command or to a secret value by the `KDF` command.

SHA Context Buffer

The SHA context buffer allows for the generation of a digest to be interrupted to do other functions or to generate other digests. The `SHA` command uses a standard three-phase flow: Initialize, Update and Finalize. In many situations, the Update phase is run many times. Internal SRAM memory is used to store the intermediate state, aka SHA context, between these phases.

5. General Command Information

The following sections provide some general information on the basic I/O transactions, command structure, error codes, memory addressing and formatting of keys and signatures that are used in the ECC608-TMNGTLS.

5.1 I/O Transactions

The ECC608-TMNGTLS uses the I^2C protocol to communicate with a host microcontroller. Security commands are sent to the device and responses are received from the device within a transaction that is constructed in the following way:

Table 5-1. I/O Transaction Format

Byte	Name	Meaning
0	Count	Number of bytes to be transferred to (or from) the device in the group, including the count byte, packet bytes and checksum bytes. Therefore, the count byte must always have a value of (N+1), where N is equal to the number of bytes in the packet plus the two checksum bytes. For a group with one count byte, 50 packet bytes and two checksum bytes, the count byte must be set to 53. The maximum size group (and value of count) is 155 bytes and the minimum size group is four bytes. Values outside this range will cause the device to return an I/O error.
1 to (N-2)	Packet	Command, parameters and data or response. See Command Packets for general command packet information or Device Commands for specific parameters for each command.
N-1, N	Checksum	CRC-16 verification of the count and packet bytes. The CRC polynomial is 0x8005. Prior to the start of the CRC calculation, the CRC register is initialized to zero. After the last bit of the count and packet is transmitted, the internal CRC register must have a value that matches the checksum bytes in the block. The first CRC byte transmitted (N-1) is the LSB of the CRC value, so the last byte of the group is the MSB of the CRC.

The ECC608-TMNGTLS is designed to have the count value in the input group consistent with the size requirements that are specified in the command parameters. If the count value is inconsistent with the command opcode and/or parameters within the packet, the ECC608-TMNGTLS responds in different ways depending upon the specific command. The response may either include an error indication or some input bytes may be silently ignored.

5.2 Command Packets

The command packet is broken down as shown in Table 5-2:

Table 5-2. Command Packets

Byte	Name	Meaning
0	Opcode	The command code. See Device Commands.
1	Param1	The first parameter; always present.
2-3	Param2	The second parameter; always present.
0-155	Data	Optional remaining input data.

After the ECC608-TMNGTLS receives all the bytes in a group, the device transitions to the Busy state and attempts to execute the command. Neither status nor results can be read from the device when it is busy. During this time, the I/O interface of the device ignores all transitions on the I^2C SDA input signal.

5.3 Status/Error Codes

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

- Count
- Packet
- Two byte CRC

After the device receives the first byte of an input command group, the system cannot read anything from the device until the system sends all the bytes to the device.

After the wake and execution of a command, there will be error, status or result bytes in the device's output register that can be retrieved by the system. For a four bytes length of that group, the codes returned are detailed in Table 5-3. Some commands return more than four bytes when they execute successfully. The resulting packet description is listed in Device Commands.

CRC errors are always returned before any other type of error. They indicate that an I/O error occurred and that the command may be resent to the device. No particular precedence is enforced among the remaining errors if more than one occurs.

Table 5-3. Status/Error Codes in Four Byte Groups

State Description	Error/ Status	Description
Successful Command Execution	0x00	Command executed successfully.
CheckMac or Verify Miscompare	0x01	The CheckMac or Verify command was properly sent to the device, but the input response did not match the expected value.
Parse Error	0x03	Command was properly received but the length, command opcode or parameters are illegal regardless of the state (volatile and/or EEPROM configuration) of the ECC608-TMNGTLS. Changes in the value of the command bits must be made before it is re-attempted.
ECC Fault	0x05	A computation error occurred during ECC processing that caused the result to be invalid. Retrying the command may result in a successful execution.
Self Test Error	0x07	There was a Self Test error and the chip is in Failure mode waiting for the failure to be cleared.
Health Test Error	0x08	There was a random number generator Health Test error and the chip fails subsequent commands requiring a random number until it is cleared.
Execution Error	0x0F	Command was properly received but could not be executed by the device in its current state. Changes in the device state or the value of the command bits must be made before it is re-attempted.
After Wake, Prior to First Command	0x11	Indication that the ECC608-TMNGTLS has received a proper Wake token.
Watchdog About to Expire	0xEE	There is insufficient time to execute the given command before the WDT expires. The system must reset the WDT by entering the Idle or Sleep modes.
CRC or other Communications Error	0xFF	Command was not properly received by the ECC608-TMNGTLS and must be retransmitted by the I/O driver in the system. No attempt was made to parse or execute the command.

5.4 Address Encoding

The following subsections provide detailed information on how to address the various memory zones of the ECC608-TMNGTLS device.

5.4.1 Configuration Zone Addressing

The Configuration zone can be accessed either 4 or 32 bytes at a time. Individual bytes cannot be accessed. The Configuration zone address is a 2-byte (16-bit value). Only the lowest five bits of the address word are used in addressing of the Configuration zone. For the ECC608-TMNGTLS device, these addresses can only be used with the `Read` command.

Table 5-4. Address Format

Byte 1: Addr[15:8]	Byte 0: Addr[7:0]		
Unused	Unused	Block	Offset
Addr[15:8]	Addr[7:5]	Addr[4:3]	Addr[2:0]

Table 5-5. Configuration Zone Addresses

Block # (Addr[4:3])	Offset Value (Addr[2:0])							
	000	001	010	011	100	101	110	111
00	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
01	[32:35]	[36:39]	[40:43]	[44:47]	[48:51]	[52:55]	[56:59]	[60:63]
10	[64:67]	[68:71]	[72:75]	[76:79]	[80:83]	[84:87]	[88:91]	[92:95]
11	[96:99]	[100:103]	[104:107]	[108:111]	[112:115]	[116:119]	[120:123]	[124:127]

5.4.2 OTP Zone Addressing

The OTP zone can be accessed either 4 or 32 bytes at a time. The zone has a total of 64 bytes. Individual bytes cannot be accessed. The OTP zone address is a 2-byte (16-bit value). Only the lowest four bits are used in addressing.

For the ECC608-TMNGTLS device, these addresses can only be used with the `Read` command.

Table 5-6. Address Format

Byte 1: Addr[15:8]	Byte 0: Addr[7:0]		
Unused	Unused	Block	Offset
Addr[15:8]	Addr[7:4]	Addr[3]	Addr[2:0]

Table 5-7. OTP Zone Byte Addresses

Block # (Addr[3)	Block Offset Value (Addr[2:0])							
	000	001	010	011	100	101	110	111
0	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
1	[32:35]	[36:39]	[40:43]	[44:47]	[48:51]	[52:55]	[56:59]	[60:63]

5.4.3 Data Zone Addressing

Read/Write access to the Data zone is much more complex than the Configuration and OTP zones. There are 16 slots and the size of the slots vary. Each slot's access policies individually control whether or not a slot has the ability to be read or written.

For the ECC608-TMNGTLS:

- Data Slots 2, 4, 7-12 and 15 are configured to be written as clear text.
- Data Slots 5 and 14 can be written with encrypted text.
- Data Slots 2, 8, 10-12 and 14-15 can be read as clear text.
- Any slots not specified cannot be read or written.

Table 5-8. Address Format by Data Slot Size

Data Zone	Byte 1: Addr[15:8]		Byte 0: Addr[7:0]		
	Unused	Block	Unused	Slot	Offset
Data Slots[7:0]	Addr[15:9]	Addr[8]	Addr[7]	Addr[6:3]	Addr[2:0]
Data Slot[8]	Addr[15:12]	Addr[11:8]	Addr[7]	Addr[6:3]	Addr[2:0]
Data Slots[15:9]	Addr[15:10]	Addr[9:8]	Addr[7]	Addr[6:3]	Addr[2:0]

Data Slots[7:0]

To fully access one of these slots, require two 32-byte accesses or nine 4-byte accesses.

Table 5-9. Data Zone Addresses Slots 0-7

Slot# (Addr[6:3])	Block # (Addr[8])	Block Offset Value (Addr[2:0])							
		000	001	010	011	100	101	110	111
0x0 to 0x7	00	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
	01	[32:35]	Not Valid	Not Valid	Not Valid	Not Valid	Not Valid	Not Valid	Not Valid

Data Slot[8]

To fully access this slot, require thirteen 32-byte accesses or 104 4-byte accesses or a combination of the two methods.

Table 5-10. Data Zone Addressing Slot 8

Slot# (Addr[6:3])	Block # (Addr[8])	Block Offset Value (Addr[2:0])							
		000	001	010	011	100	101	110	111
0x8	0x0	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
	0x1	[32:35]	[36:39]	[40:43]	[44:47]	[48:51]	[52:55]	[56:59]	[60:63]
	...	...	...	...	...	...	...	...	...
	0xC	[384:387]	[388:391]	[392:395]	[396:399]	[400:403]	[404:407]	[408:411]	[412:415]

Data Slots[15:9]

To fully access these slots, require three 32-byte accesses or eighteen 4-byte accesses or a combination of the two methods.

Table 5-11. Data Zone Addressing Slots 9-15

Slot# (Addr[6:3])	Block # (Addr[8])	Block Offset Value (Addr[2:0])							
		000	001	010	011	100	101	110	111
0x9 to 0xF	00	[0:3]	[4:7]	[8:11]	[12:15]	[16:19]	[20:23]	[24:27]	[28:31]
	01	[32:35]	[36:39]	[40:43]	[44:47]	[48:51]	[52:55]	[56:59]	[60:63]
	10	[64:67]	[68:71]	Not Valid	Not Valid	Not Valid	Not Valid	Not Valid	Not Valid

5.5 Formatting of Keys, Signatures and Certificates

The following sections provide detailed formatting information for ECC keys, Signatures and Compressed certificates.

5.5.1 ECC Key Formatting

The format for public and private keys depends on the command and key length. In general, the Most Significant Bytes (MSB) appear first on the bus and at the lowest address in memory. In the remainder of this section below, the bytes on the left side of the page are the MSBs. Microchip recommends all pad bytes be set to zero for consistency.

- ECC private keys appear to the user only as the input parameter to the `PrivWrite` command. This parameter is always 36 bytes in length and the first four bytes (32 bits) are all pad bits.

ECC public keys appear as the input or output parameters to several commands, and they can also be stored in EEPROM. They are composed of an X value first on the bus or in memory, followed by a Y value. They are formatted differently depending on the situation as noted below:

- **The public key is an output of the `GenKey` command or an input to the `Verify` command**: 32 bytes of X, then 32 bytes of Y. (36 bytes) There are no pad bytes.
- **`Write` command**:
  Public keys can be written directly to the EEPROM using the `Write` command and are always 72 bytes long, formatted as follows: 4-pad bytes, 32 bytes of X, four pad bytes, then 32 bytes of Y.
- **`GenKey` command**:
  SHA Message: Public keys can be hashed and placed in TempKey by the `GenKey` command. The SHA message contains various bytes that are independent of the size of the key. These are followed by 25 bytes of pad, followed by 32 bytes of X, then by 32 bytes of Y.
- **`Verify` command**:
  SHA Message: When used to validate a stored public key, the `Verify` command expects an input signature created over a SHA-256 digest of a key stored in memory. Such an inner SHA calculation is always performed over 72 bytes formatted as they are stored in EEPROM as 4-pad bytes, 32 bytes of X, 4-pad bytes, then 32 bytes of Y.

When a public key is configured to be validated by the `Verify` command, the Most Significant four bits of the first byte in memory are used internally by the device to save the validation state. They are always set to the invalid state (`0xA`) by the `Write` command, and then may be set to the Valid state (`0x5`) by the `Verify` command.

The lowest levels of the I/O protocols are described below. Above the I/O protocol level, the exact same bytes are transferred to and from the device to implement the commands. Error codes are documented in the following sections.

5.5.1.1 Public Key Formats

The ECC608-TMNGTLS works with the P-256 elliptic curve public keys in two formats. The following example illustrates those two formats in detail.

For the following examples, we'll use a sample public key, with the X and Y integers expressed as fixed-width big-endian unsigned integers:

```
X: b2be345ad7899383a9aab4fb968b1c7835cb2cd42c7e97c26f85df8e201f3be8
Y: a82983f0a11d6ff31d66ce9932466f0f2cca21ef96bec9ce235b3d87b0f8fa9e
```

Command Public Key Format

Any command that returns a public key (`GenKey`) or accepts a public key as a parameter (`Verify` and `ECDH`) will format the public key as the X and Y big-endian unsigned integers concatenated together for a total of 64 bytes.

For example:

```
b2be345ad7899383a9aab4fb968b1c7835cb2cd42c7e97c26f85df8e201f3be8
a82983f0a11d6ff31d66ce9932466f0f2cca21ef96bec9ce235b3d87b0f8fa9e
```

Stored Public Key Format

When storing a public key in a slot for use with the `Verify` command, , the X and Y integers will be padded out to 36 bytes and concatenated together for a total of 72 bytes.

For example:

```
00000000b2be345ad7899383a9aab4fb968b1c7835cb2cd42c7e97c26f85df8e201f3be8
00000000a82983f0a11d6ff31d66ce9932466f0f2cca21ef96bec9ce235b3d87b0f8fa9e
```

Note: Only Slots 8-15 are large enough to hold a public key.

Stored Validated Public Key Format

A validated or invalidated public key format is the same as a stored public key format with the exception of the four Most Significant bits of the LSB. If a key is validated, the Least Significant Nibble will be 0x5 and 0xA if invalidated. These values can be changed by the `Verify` command in Validate or Invalidate mode. When written, the key will be initially invalidated.

Validated Public Key Example:

```
50000000b2be345ad7899383a9aab4fb968b1c7835cb2cd42c7e97c26f85df8e201f3be8
00000000a82983f0a11d6ff31d66ce9932466f0f2cca21ef96bec9ce235b3d87b0f8fa9e
```

Invalidated Public Key Example:

```
A0000000b2be345ad7899383a9aab4fb968b1c7835cb2cd42c7e97c26f85df8e201f3be8
00000000a82983f0a11d6ff31d66ce9932466f0f2cca21ef96bec9ce235b3d87b0f8fa9e
```

Note: Only Slots 8-15 are large enough to hold a public key.

5.5.2 Signature Format

The ECDSA signature that is generated and output by the `Sign` command or input to the `Verify` command is always 64 bytes in length. The signature is divided into R and S components. Both components are 32 bytes in length and R always appears before S on the bus. Each portion of the signature appears MSB first on the bus, meaning the MSB of the signature is in the lowest memory location.

Example R/S Signature

Any command that returns a signature (`Sign`) or accepts a signature as a parameter (`Verify`) will format the signature as the R and S big-endian unsigned integers concatenated together for a total of 64 bytes.

For example:

```
R: 7337887F8C39DF79FD8BF88DDFBFB9DB15D7B1AD68196AE3FB0CE5BFA2842DF3
S: 72868A43A42831E950E1DA9F73B29F5C0ED8A96B2889E3CBBE8E61EA6C67F673
```

5.5.3 Certificate Storage

The amount of storage required for a full X.509 Certificate within the device can rapidly use up multiple EEPROM memory slots. Depending on the actual application, it may or may not be desirable to use these slots for certificate storage. Due to these memory limitations, Microchip has defined an encoding that allows for a full X.509 Certificate to be reconstructed from a minimal amount of information.

The host system would actually be responsible for reconstructing the full X.509 Certificate but how to do this will be determined by the data stored in the encoded certificate. Data that are common to all devices for a given system can readily be stored in the host system. Other data can readily be calculated or extracted from data that are already stored in the device. Table 5-12 indicates the type of data that are stored in an X.509 Certificate and how it can be encoded to fit into a single 72-byte slot.

Table 5-12. Certificate Storage

X.509 Certificate		Encoded Certificate		
X.509 Element	Size (Bytes)	Encoded Certificate Element	Device Cert (Bits)	Signer Cert (Bits)
Serial Number	8-20	Serial number source	4	4
Issue Date	13	Compressed format	19	19
Expiry Date	13	# of years before expiration	5	5

.....continued

X.509 Certificate		Encoded Certificate		
X.509 Element	Size (Bytes)	Encoded Certificate Element	Device Cert (Bits)	Signer Cert (Bits)
Signer ID[2]	4	ID of the specific signer used to sign the certificate (device cert) or of the signer itself (signer cert)	16	16
AuthorityKeyIdentifier	20	SHA1 HASH of the authority public key	0	0
SubjectKeyIdentifier	20	SHA1 HASH of the subject public key	0	0
Signature R	32	Stored in device	256	256
Signature S	32	Stored in device	256	256
Public Key X[1]	32	Calculated from the private key or stored in the device[1]	0	256
Public Key Y[1]	32	Calculated from the private key or stored in the device[1]	0	256
n/a	0	Cert format	4	4
n/a	0	Template ID	4	4
n/a	0	Chain ID	4	4
n/a	0	Reserved/User Defined	8	8
Total	(206-218 bytes)	—	576 bits (72 bytes)	1088 bits (136 bytes)

Notes:

1. For the device certificate, the device public key can be regenerated from the private key. For the signer certificate, the public key is typically stored in a separate slot.
2. For the device certificate, the ID of the signer used to sign the certificate is stored. For the signer certificate, the actual ID of the signer is stored so that the device can identify it.

Slot 8 contains a total of 416 bytes. Depending on the size of the serial number stored in the cert, it may or may not be possible to store two complete certificates. Often within devices where a chain of trust has been created, the device certificate, the signer certificate and the signer public key must be stored within the device.

For more information, see the Compressed Certificate Definition Application Note.

6. Device Commands

The following section details all of the commands broken out by Command mode that are allowed in the ECC608-TMNGTLS. The commands are broken into three categories:

1. **General Device Commands**
   These commands fall into two categories:
   - General device access commands that are used to send data to the device or retrieve data but typically do not perform any cryptographic functions.
   - General cryptographic commands that can be used by the device or the system but typically do not operate on specific data slots.
2. **Asymmetric Cryptography Commands**
   These commands perform asymmetric cryptographic operations, such as key generation, message signing and message verification that utilize an ECC public or private key. These commands are limited to use on ECC data zone slots.
3. **Symmetric Cryptography Commands**
   These commands perform a symmetric cryptographic function, such as generating a digest or MAC, key derivation or AES encryption and decryption.

Input Parameters for all Commands

Multibyte input parameters are shown as big-endian (MSB first) values in the input parameters tables unless otherwise specified. Note that the ECC608-TMNGTLS device actually expects the data to be sent little-endian (LSB first).

Table 6-1. Commands, Descriptions and Command Categories

Command	Description	Command Category
AES	Execute the AES-ECB Encrypt or Decrypt functions. Calculate a Galois Field Multiply.	Symmetric Cryptography Command
CheckMac	Verify a MAC calculated on another CryptoAuthentication™ device.	Symmetric Cryptography Command
Counter	Read or increment one of the monotonic counters.	General Device Commands
ECDH	Generate an ECDH pre-master secret using stored private key and input public key.	Asymmetric Cryptography Command
GenDig	Generate a data digest from a random or input seed and a stored value.	Symmetric Cryptography Command
GenKey	Generate an ECC public key. Optionally generate an ECC private key.	Asymmetric Cryptography Command
Info	Return device state information.	General Device Commands
KDF	Implement the PRF or HKDF key derivation functions	Symmetric Cryptography Command
Lock	Prevent further modifications to a zone or slot of the device.	General Device Commands
MAC	Calculate digest (response) from key and other internal data using SHA-256.	Symmetric Cryptography Command
Nonce	Generate a 32-byte random number and an internally stored nonce.	General Device Commands
Random	Generate a random number.	General Device Commands
Read	Read 4 or 32 bytes from the device, with or without authentication and encryption.	General Device Commands
SecureBoot	Validate code signature or code digest on power-up.	Asymmetric Cryptography Command

.....continued		
Command	Description	Command Category
SelfTest	Test the various internal cryptographic computation elements.	General Device Commands
Sign	ECDSA signature calculation.	Asymmetric Cryptography Command
SHA	Compute a SHA-256 or HMAC digest for general purpose used by the system.	General Device Commands
UpdateExtra	Update bytes 84 or 85 within the Configuration zone after the Configuration zone is locked.	General Device Commands
Verify	ECDSA verify calculation.	Asymmetric Cryptography Command
Write	Write 4 or 32 bytes to the device, with or without authentication and encryption.	General Device Commands

Related Links
General Device Commands
Asymmetric Cryptography Commands
Symmetric Cryptography Commands

6.1 General Device Commands

The following table provides a summary of the general device commands:

Table 6-2. General Device Commands

Command Name	Description
Counter	Increments and reads the monotonic counters
Info	Used to read revision and status information from the device
Lock	Used to lock the individual lockable slots in the device
Nonce	Used to generate or pass a number used once into the device
Random	Used to generate a 32-byte random number used by the system
Read	Used to read various zones of the device
SelfTest	Tests the various internal cryptographic computation elements
SHA	Computes a SHA-256 or HMAC digest for general purpose use by the system
UpdateExtra	Updates bytes 84 or 85 within the Configuration zone after the Configuration zone is locked
Write	Used to write 4 or 32 bytes to the device, with or without authentication and encryption

6.1.1 `Counter` Command

The `Counter` command reads the binary count value from one of the two monotonic counters located on the device within the Configuration zone. The maximum value that the counter may have is 2,097,151. Any attempt to count beyond this value will result in an error code. The counter is designed to never lose counts even if the power is interrupted during the counting operation. In some power loss conditions, the counter may increment by a value of more than one.

For the ECC608-TMNGTLS, the counters are not attached to any keys but may still be used by the system. Each count is set to its default value and can count to the maximum value.

6.1.2 `Info` Command

The `Info` command is used to read the status and state of the device. This information is useful in determining errors or to operate various commands.

6.1.3 Lock Command

For the ECC608-TMNGTLS, the Configuration zone is already locked and the access policies of the Data zone are set. However, several of the data slots can still be updated through the use of other commands. If so desired, some of these slots can be permanently locked from future updates by using the Slot Locking mode of the Lock command.

6.1.4 Nonce Command

The Nonce command generates a nonce (number used once) for use by a subsequent command by combining a random number (which can be generated internally or externally) with an input value from the system. The resulting nonce is stored internally in three possible buffers: TempKey, Message Digest Buffer and Alternate Key Buffer. Instead of generating a nonce, a value may be passed to the device if so desired.

6.1.5 Random Command

The Random command generates a random number to be used by the system. Random numbers are generated via the internal NIST 800-90 A/B/C random number generator. The output of the command is always a 32-byte number placed on the bus. The number cannot be stored in any data slot or SRAM location.

6.1.6 Read Command

The Read command can be used to access any of the EEPROM zones of the ECC608-TMNGTLS device. Data zone access is limited based on the access policies set for each of the slots. Encrypted reads are possible only on the Data zone slots if specific access policies are set.

6.1.7 SelfTest Command

The SelfTest command performs a test of one or more of the cryptographic engines within the ECC608-TMNGTLS chip. Some or all of the algorithms will be tested depending on the Input mode parameter.

For the ECC608-TMNGTLS device, the SelfTest command has been disabled from running automatically after a Power-up or Wake event. However, the command may be executed by the system, if so desired. There is no requirement to run this test.

If any self test fails, whether called automatically on power-up, wake or via this command, the chip will enter a Failure state, where chip operation is limited. The stored Failure state is always cleared upon a wake or power cycle.

6.1.8 SHA Command

The SHA command computes a SHA-256 or HMAC/SHA digest for general purpose use by the host system. The SHA computation is performed in a special section of internal ECC608-TMNGTLS memory (Context Buffer) that is not read nor written by any other commands. Any arbitrary command can be interspersed between the various phases of the SHA command without problems. This SHA context is invalidated on power-up and wake. In most cases, if an error occurs during the execution of the SHA command, the context is retained without change.

6.1.9 UpdateExtra Command

The UpdateExtra command is used to update the UpdateExtra and UpdateExtraAdd bytes, bytes 84 and 85 respectively, in the Configuration zone. These bytes can only be updated by this command. These bytes are one-time updatable bytes and can only be updated if the current value is 0x00. Trying to update this byte if the value is not 0x00 will result in an error.

For the ECC608-TMNGTLS device, the UpdateExtraAdd byte (byte 85) is configured to be an alternate I^2C address.

6.1.10 `Write` Command

For the ECC608-TMNGTLS, the Configuration zone and OTP zone are locked and no updates to these zones are possible. Limited write capability exists on the Data zone based on access policies of each slot.

6.2 Asymmetric Cryptography Commands

The Asymmetric Cryptography command set is made up of those commands that are specifically used to generate or use ECC keys. Keys are typically stored in Data zone slots, but, for some commands, could also be in the SRAM array.

Table 6-3. Asymmetric Cryptography Commands

Command Name	Description
`ECDH`	Generates an ECDH pre-master secret using the stored private key and input public key
`GenKey`	Generates an ECC private key or optionally generates an ECC public key from the stored private key
`SecureBoot`	Validates code signature or code digest on power-up
`Sign`	Signs an internal or external message digest using an ECC private key with an ECDSA signature calculation
`Verify`	Verifies an internal or external message digest using an ECC public key with an ECDSA verify calculation

6.2.1 `ECDH` Command

The `ECDH` command is used to generate a shared secret between two devices. By passing an ECC public key from another device and combining it with the ECC private key stored in a slot or with an ephemeral key stored in TempKey and doing the reverse on the other device, both devices will generate the same shared pre-master secret. This can, then, be further combined with other common data in both sides to generate a shared session key between the devices. The `KDF` command is often used with TLS sessions to further diversify the shared secret.

6.2.2 `GenKey` Command

The `GenKey` command is used to generate ECC private keys, ECC public keys from private keys or generate a public key digest. This command is only applicable for those slots designated to be ECC private or public keys. Running this command on a non-ECC slot will result in an error.

6.2.3 `SecureBoot` Command

The `SecureBoot` command provides support for the secure boot of an external MCU or MPU. The general approach is that the boot code within the system will use the ECC608-TMNGTLS to assist in validating the application code that is to be subsequently executed. The ECC608-TMNGTLS device is configured to operate in the SecureBoot, Stored Digest mode. The digest will be stored in Slot 13 and the public key required to verify the SecureBoot is stored in Slot 15.

In lieu of a return code, a MAC can optionally be generated from a nonce written to TempKey, the I/O protection secret and various other data, dependent upon the mode of the command, to prevent tampering with the wire between the host and the ECC608-TMNGTLS.

6.2.4 `Sign` Command

The `Sign` command generates a signature using the ECDSA algorithm. The ECC private key in the slot specified by KeyID is used to generate the signature. Multiple modes of the device are available depending on what is being signed.

6.2.5 `Verify` Command

The `Verify` command takes an ECDSA [R,S] signature and verifies that it is correctly generated given an input message digest and public key. In all cases, the signature is an input to the command. The public key can be either stored on the device or provided as an input.

An optional MAC can be returned from the `Verify` command to defeat any man-in-the-middle attacks. If the verify calculation shows that the signature is correctly generated from the input

digest, a MAC will be computed based on an input nonce stored in TempKey and the value of the I/O protection secret, which is stored in both the ECC608-TMNGTLS and the host MCU. MAC outputs can only be generated in External and Stored modes. The I/O protection function must be enabled for MAC computation.

AES keys stored in Slot 4 and 9 must be validated before use.

6.3 Symmetric Cryptography Commands

The Symmetric Cryptography command set is made of up of those commands associated with the generation or use of Symmetric Keys. Keys are typically stored in Data zone slots, but for some commands could also be in the SRAM memory locations.

Table 6-4. Symmetric Cryptography Commands

Command Name	Description
AES	Execute the AES-ECB Encrypt or Decrypt functions. Calculate a Galois Field Multiply.
CheckMac	Verify a MAC calculated on another CryptoAuthentication™ device.
GenDig	Generate a data digest from a random or input seed and a stored value.
KDF	Implement the PRF or HKDF key derivation functions
MAC	Calculate digest (response) from key and other internal data using SHA-256.

6.3.1 AES Command

The AES Command can be used to encrypt and/or decrypt a 16-byte block of data utilizing an AES key. Note that the key is stored in a 16 Byte (128 bit) location with a given slot or within the first 16 bytes of TempKey. Multiple keys may be stored in a given slot and accessed in successive 16-byte boundaries, starting with 0-15 up to the size of the slot, but not exceeding four keys in any slot. For the ECC608-TMNGTLS, up to two AES keys may be stored in Slot 5.

In addition to AES encryption and decryption, the AES command may be used to generate a Galois Field Multiply (GFM) in support of other cryptographic operations.

6.3.2 CheckMac Command

The CheckMac command calculates a MAC response that would have been generated on a different CryptoAuthentication device[1] and then compares the result with the input value. The command returns a boolean result to indicate the success or failure of the comparison.

If a value in TempKey is used as an input to the CheckMac, then a Nonce and/or GenDig command must be run prior to the CheckMac command.

6.3.3 GenDig Command

The GenDig command uses a SHA-256 hash to combine a stored or input value with the contents of TempKey, which must be validated prior to the execution of this command. The stored value can come from one of the data slots, the Configuration zone, either of the OTP pages or the monotonic counters. The specific mode of the device determines which data are to be included in the GenDig calculation.

In some cases, it is required to run the GenDig prior to the execution of some commands. The command can be run multiple times to include more data in the digest prior to executing a given command. The resulting digest is retained in TempKey and can be used in one of four ways:

1. It can be included as part of the message used by the MAC, Sign or CheckMac commands. Because the MAC response output incorporates both the data used in the GenDig calculation and the secret key from the MAC command, it serves to authenticate the data stored in the Data and/or OTP zones.

[1] Devices that generate a compatible MAC response include, ATECC608A/B/C, ATECC508A, ATSHA204A and SHA104. This includes Trust variants associated with the products.

2. A subsequent `Read` or `Write` command can use the digest to provide authentication and/or confidentiality for the data, in which case, it is known as a data protection digest.
3. The command can be used for secure personalization by using a value from the transport key array. The resulting data protection digest would, then, be used by write.
4. The input value, typically a nonce from a remote device, is combined with the current TempKey value to create a shared nonce in which both devices can attest to the inclusion of the RNG.

6.3.4 KDF Command

For the ECC608-TMNGTLS, the `KDF` command implements several key derivation functions including PRF for TLS 1.2, HKDF for TLS 1.3 and AES. Support for this function is available in CryptoAuthLib. See CryptoAuthLib for more information.

For additional information on the `KDF` command contact Microchip Sales for more information.

6.3.5 MAC Command

The Message Authentication Code (`MAC`) command is used to generate a SHA-256 digest of a message, which consists of a key stored in the device, a challenge and other information on the device. The output of this command is the digest of this message.

The normal flow to use this command is as follows:

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

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

7. Application Information

The ECC608-TMNGTLS is a member of Microchip's CryptoAuthentication™ Trust Platform family of products. The Trust Manager products are self-service cryptographic key management devices. These devices are simple to implement and allow even low-volume customers to implement secure authentication into their end system while leveraging Microchip's and Kudelski IoT expertise and provisioning infrastructures.

The ECC608-TMNGTLS device was developed to take the guesswork out of adding security to an IoT- connected product. The product is pre-configured to readily connect to any cloud infrastructure through DTLS connections.

When the actual security device is combined with the Kudelski IoT keySTREAM SaaS service, a complete trust solution for the lifecycle of the product exists. Multiple use case are supported by the solution.

7.1 Use Cases

The ECC608-TMNGTLS was defined to specifically address the IoT market. Through use of the Kudelski keySTREAM SaaS, a wide array of cloud networks and specific authentication services can be accessed. The inherent flexibility built in to the system allows for a custom PKI to be established in-field, with no factory intervention from Microchip. This allows for the ability to modify and rotate certificates, update and maintain security policies for a variety of applications and use cases within those applications. A brief description of some of the use cases that this device addresses is provided below. These use cases can be implemented separately or in combination with each other. To prototype and implement these use cases, Microchip provides both hardware and software tools.

Secure TLS Connection

The ECC608-TMNGTLS allows the creation of a secure authentication in the TLS protocol. The device will store the birth private key generated by Microchip within our certified secure provisioning facilities. This key will issue an ECC-P256 signature to the keySTREAM SaaS, which will authenticate the ECC608 TrustManager. The SaaS will, then, create a custom PKI and provision the new custom certificates in the device. This is the in-field provisioning of the custom PKI. The private keys associated to the root and intermediate certificates are protected in the Kudelski IoT HSMs, which are set up during the customer account setup. This, then, allows for connection to a wide array of cloud providers. Through the various modes of the Key Derivation Function (KDF), appropriate keys can be generated to support TLS1.2, TLS1.3 and earlier secure connection Internet protocols.

Certificate Management

Through use of the custom PKI that was set up in keySTREAM SaaS when the account was set up, the SaaS can be used to manage and monitor the certificate expiration date, certificate revocation and certificate renewal.

After the initial account setup, the expiration date of the certificates can be set and stored in the ECC608-TMNGTLS. When the certificate expiration date is reached, it can be can be configured to renew automatically as specified by the Kudelski IoT policies so as to prevent loss of service due to an expired certificate.

If a security breach or some other incident occurs that warrants the revocation of a certificate, through use of the KeySTREAM SaaS, the ECC608-TMNGTLS device certificates can be revoked. Revocation of the certificate will prevent the device from connecting to the cloud. If so desired, to restore secure authentication to the device, the certificate can be renewed by issuing a new certificate to replace the current certificate stored in the device. Revocation and renewal achieve certificate rotation.

Private Key Rotation

The keySTREAM SaaS is able control the rotation of private keys inside the ECC608-TMNGTLS. If the end product private key is determined to no longer be trustworthy, a new private key, stored in Slot

0 can be regenerated through use of keySTREAM SaaS. The risk is a denial of service on Slot 0, where keySTREAM SaaS will deny access to the SaaS for that particular device.

Secure Boot

Protecting the firmware image of a microcontroller or microprocessor is a concern for many vendors. By providing a mechanism to verify that the code being run is authentic and was not modified, the overall integrity of the system is maintained. The ECC608-TMNGTLS was configured to allow Secure Boot by storing the code digest of the system within a data slot of the device. Upon initial execution of the code, the system can regenerate the digest over the system firmware and compare it with the digest stored in the ECC608-TMNGTLS, verifying that the firmware was not tampered with.

IP and Data Protection

Protecting Intellectual Property (IP) can be crucial to maintaining a company's competitive edge. IP protection describes the way of protecting the firmware or hardware developed by the customer from being copied. Firmware IP protection can be done with just a software-based approach, but the key information inside the firmware still remains quite vulnerable to attacks.

The ECC608-TMNGTLS device offers hardware-based secure key storage to ensure that a product with the firmware runs. The devices can perform both the Symmetric authentication and Asymmetric authentication where the keys are securely stored in the secure element, thereby, reducing a hacker's ability to extract and modify the keys. The Slot 15 public key can be used to verify either the hash of a firmware image or the signature of a signed image, respectively, during any point of run-time operations.

End-to-End Data Protection

Through the use of symmetric keys, secure data communication can be implemented. Through the use of the ECDH function, symmetric keys can be generated to allow for encryption and decryption of communication between devices and endpoints. Data can be fully protected from cloud to device and device to cloud.

General Data Storage

Data Slot 8 is used to store specific customer/application onboarding information needed for multiple keySTREAM SaaS operations. This slot is not available for general use by the customer. Sometimes there is a need to store a small amount of additional information for a given system. Data Slot 15 can be used to store additional customer-specific or application-specific information. The ECC608-TMNGTLS can be used for this purpose by utilizing the data slot that allows read and write access. This eliminates the need to add an additional EEPROM memory device to only store data.

Restriction: If data Slot 15 is used for general data storage, the Secure Boot feature cannot be implemented.

7.2 Development Tools

The ECC608-TMNGTLS is supported with multiple hardware and software tools and backend services that provide a path to rapidly develop applications. Initial development can start by using a family of easy-to-use Trust Platform Design Suite tools. These tools provide a graphical way to implement your use case and end with the C code necessary to implement your application.

If your application differs from what the predefined Trust Platform Design Suite tools can provide, then through use of the CryptoAuthLib or the Python® version of CryptoAuthLib and CryptoAuthTools, an application can be developed. CryptoAuthLib is also the backbone of the code that is output from the Trust Platform Design Suite tools.

Full verification of your application can be implemented via hardware tools along with samples of the ECC608-TMNGTLS device. The access policies of the device are already set, therefore, the focus revolves just around developing the system level code.

Once the application is complete, the ECC608-TMNGTLS devices can be ordered through Microchip Direct.

7.2.1 Trust Platform Design Suite

To simplify the implementation process, Microchip developed a web-based Trust Platform Design Suite of tools that will allow developers to go from concept to production via a guided flow. The tools allow you to develop and construct the transaction diagrams and code necessary to implement a particular application within the constraints of the configuration and defined access policies of the ECC608-TMNGTLS.

Note: More information on these tools can be found on Microchip's Trust Platform information page.

7.2.2 Hardware Tools

There are multiple hardware tools that can help in developing with the ECC608-TMNGTLS. Check the Microchip website for the availability of additional tools that are not mentioned here.

EA06V72A/EV10E69A – CryptoAuthentication Trust Manager Platform

The EA06V72A/EV10E69A is a compact development system consisting of an ATSAMD21 microcontroller, the ECC608-TMNGTLS Trust Manager device, a USB Hub, a mikroBUS™ connector and an on-board debugger. The kit is intended for use with the Trust Platform Design Suite (TPDS) of tools used to implement various use cases for the ECC608-TMNGTLS device. This is the kit that is recommended to be used with the Kudelski keySTREAM SaaS tools and Microchip's Trust Platform Development Suite of tools.

The EA06V72A is a board for early adopters of the ECC608-TMNGTLS but will be replaced long term with the EV10E69A production board. The kits will operate identically as far as using TPDS with the Trust Manager development flow and Kudelski keySTREAM SaaS tools. The EV10E69A includes several additional user LEDs and a user switch that is not included on the EA06V72A.

DT100104 ATECC608 Trust Board

Revision 4 of the DT100104 ATECC608 Trust Board contains a single ECC608-TMNGTLS device along with other ATECC608C Trust devices. The trust manager device is configured the same as the ECC608-TMNGTLS prototype device that can be ordered from Microchip Direct or is on the Trust Manager Platform board. The board has a mikroBUS format and can readily be connected to the CryptoAuth Trust Manager Platform or CryptoAuth Trust Platform (DM320118). The CryptoAuth Trust Platform in combination with the DT100104(Rev 4) board, with appropriate DIP Switch settings, will operate identically to the CryptoAuth Trust Manager Platform board.

Secure UDFN/SOIC Click Boards

To work with either SOIC or UDFN samples of the ECC608-TMNGTLS, along with the Trust Manager Platform Development Board, it is recommended that the user acquires specially designed mikroBUS socket boards that work with the Trust Manager Platform board. These socket boards are provided through mikroElektronika and are designed to install in the mikroBUS header of the Trust Manager Platform board. These socket boards allow for the ability to program up one or more devices and then use them on prototype or production boards. Samples of the ECC608-TMNGTLS must be acquired separately to be used with these kits.

- Secure SOIC Click Board
- Secure UDFN Click Board

7.2.3 CryptoAuthLib

CryptoAuthLib is a software library that supports Microchip's family of CryptoAuthentication devices. Microchip recommends working with this library when developing with the ECC608-TMNGTLS. The library implements the API calls necessary to execute the commands detailed in this data sheet.

The library was implemented to readily work with many of Microchip's microcontrollers but can easily be extended through a Hardware Abstraction Layer (HAL) to other microcontrollers, including those made by other vendors.

For more details on these tools, check the information on:

- CryptoAuthLib – Web Link
- CryptoAuthLib – GitHub

API Calls

Each of the commands in the data sheet have one or more API calls that are associated with them. Typically, there is a base API call of the command where all input parameters can be specified. The parameter shown in the commands and subsections can be used with this command. There are also mode variants of each of the API calls. The table below shows examples of commands and base API calls. For the most accurate API information, refer to the GitHub information.

Table 7-1. Example Commands to CryptoAuthLib API Calls

Device Command	API Call	Comments
Info	atcab_info_base()	
Write	atcab_write()	
Read	atcab_read_zone()	
SHA	atcab_sha_base()	
Sign	atcab_sign_base()	
Random	atcab_random()	
Verify	atcab_verify()	

8. I²C Interface

The I²C interface uses the SDA and SCL pins to indicate various I/O states to the ECC608-TMNGTLS. This interface is designed to be compatible with other Microchip CryptoAuthentication and Serial EEPROM devices at the protocol level. Exact commands and device I²C Addresses may differ.

The SDA pin is normally pulled high with an external pull-up resistor because the ECC608-TMNGTLS client includes only an open-drain driver on its output pin. The bus host may either be open-drain or totem pole. In the latter case, it must be tri-stated when the ECC608-TMNGTLS is driving results on the bus. The SCL pin is an input and must be driven both high and low at all times by an external device or resistor.

For the ECC608-TMNGTLS, the default 7-bit I²C address is defined to be 0x38. With this value, the I²C address write/read byte values will be 0x70 and 0x71, respectively. If so desired, the I²C address value can be overwritten one time using the `UpdateExtra` command.

Remember: The I²C standard uses the terminology “Master” and “Slave”. The equivalent Microchip terminology used in this document is “Host” and “Client”, respectively.

Related Links
AC Parameters: I2C Interface

8.1 I/O Conditions

The device responds to the following I/O conditions:

8.1.1 Device is Asleep

When the device is asleep, it ignores all but the Wake condition.

- Wake – Upon the rising edge of SDA, after SDA is held low for a period ≥ t_{WLO}, the device exits the Low-Power mode. After a delay of t_{WHI}, it will be ready to receive I²C commands.
- The device ignores any levels or transitions on the SCL pin when the device is idle or asleep and during t_{WLO}. At some point during t_{WHI}, the SCL pin is enabled and the conditions listed in Device is Awake are honored.

The Wake condition requires that either the system processor manually drive the SDA pin low for t_{WLO}, or a data byte of `0x00` be transmitted at a clock rate sufficiently slow so that SDA is low for a minimum period of t_{WLO}. When the device is awake, the normal processor I²C hardware and/or software can be used for device communications. This includes the I/O sequences required to put the device back into Low-Power (i.e., Sleep) mode.

Tip: A simple way to generate a wake pulse is to send a byte of `0x00` at 100 kHz. Subsequent commands can be run at a higher frequency.

In the I²C mode, the device will ignore a wake sequence that is sent when the device is already awake.

Multiple Devices on the Bus

When there are multiple devices on the bus and the I²C interface is run at speeds of less than ~300 kHz[2], the transmission of certain data patterns will cause the ECC608-TMNGTLS devices on

[2] The actual frequency for a given device will vary with process and environmental factors. This value is considered safe under all conditions.

the bus to wake up. The lower the frequency, the higher the probability that the device wakes up. Because subsequent device addresses transmitted along the bus only match the desired devices, the ECC608-TMNGTLS will not respond but will be awake. It is recommended that after communicating with another device at slow frequencies, a sleep or idle sequence be issued to place the ECC608-TMNGTLS back into a known state.

Important: t_{WLO} is the minimum time that the system must provide to ensure that the ECC608-TMNGTLS will wake under all manufacturing and environmental conditions. In actuality, the device may wake up with a lesser pulse width.

If multiple devices are on the I^2C bus, it is recommended that the ECC608-TMNGTLS device be woken up after a power-up sequence and prior to communicating with any device on the bus. This is to ensure that the ECC608-TMNGTLS has properly initialized.

Related Links

Device is Awake

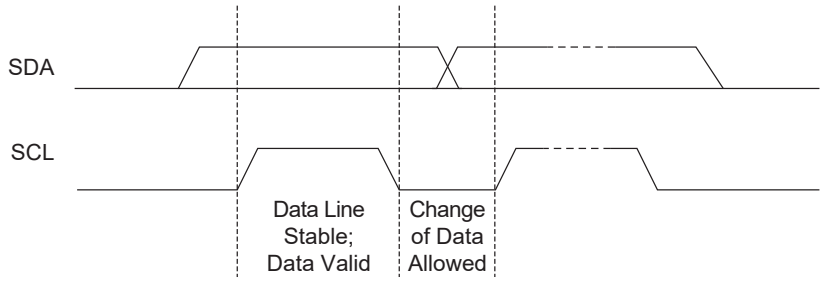
AC Parameters: All I/O Interfaces

8.1.2 Device is Awake

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

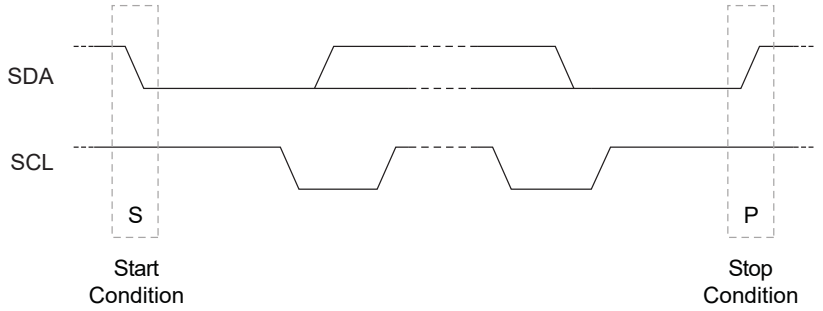
- **DATA Zero**: If SDA is low and stable while SCL goes from low to high to low, then a zero bit is being transferred on the bus. SDA can change while SCL is low.
- **DATA One**: If SDA is high and stable while SCL goes from low to high to low, then a one bit is being transferred on the bus. SDA can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

Figure 8-1. Data Bit Transfer on I^2C Interface



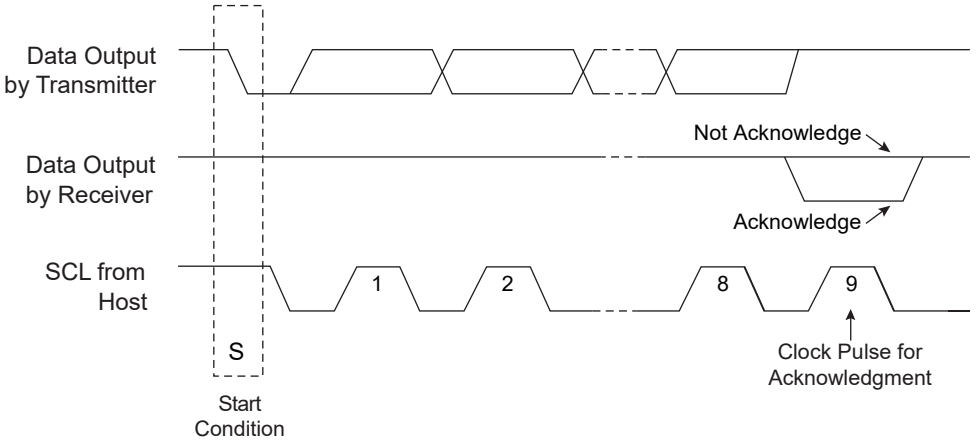
- **Start Condition**: A high-to-low transition of SDA with SCL high is a Start condition which must precede all commands.
- **Stop Condition**: A low-to-high transition of SDA with SCL high is a Stop condition. After this condition is received by the device, the current I/O transaction ends. On input, if the device has sufficient bytes to execute a command, the device transitions to the busy state and begins execution. The Stop condition must always be sent at the end of any packet sent to the device.

Figure 8-2. Start and Stop Conditions on I²C Interface



- **Acknowledge (ACK)**: On the ninth clock cycle after every address or data byte is transferred, the receiver will pull the SDA pin low to acknowledge proper reception of the byte.
- **Not Acknowledge (NACK)**: Alternatively, on the ninth clock cycle after every address or data byte is transferred, the receiver can leave the SDA pin high to indicate that there was a problem with the reception of the byte or that this byte completes the group transfer.

Figure 8-3. NACK and ACK Conditions on I²C Interface



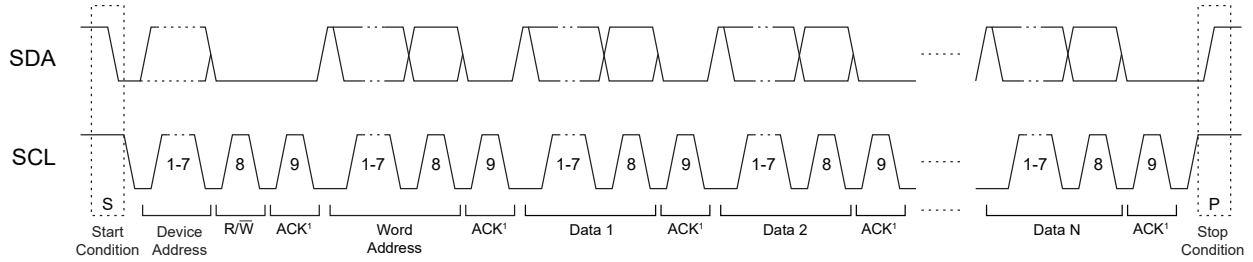
Multiple ECC608-TMNGTLS devices can easily share the same I²C interface signals if the I2C_Address byte in the Configuration zone is programmed differently for each device on the bus. Because all seven of the bits of the device address are programmable, ECC608-TMNGTLS can also share the I²C interface with any I²C device, including any Serial EEPROM.

8.2 I²C Transmission to ECC608-TMNGTLS

The transmission of data from the system to the ECC608-TMNGTLS is summarized in the table below. The order of transmission is as follows:

- Start Condition
- Device Address Byte
- Word Address Byte
- Optional Data Bytes (1 through N)
- Stop Condition

Figure 8-4. Normal I²C Transmission to ECC608-TMNGTLS



SDA is driven low by the ECC608-TMNGTLS ACK periods.

The following tables label the bytes of the I/O transaction. The column labeled "I²C Name" provides the name of the byte as described in the AT24C16 data sheet.

Table 8-1. I²C Transmission to ECC608-TMNGTLS

Name	I²C Name	Description
Device Address	Device Address	This byte selects a particular device on the I²C interface. The ECC608-TMNGTLS is selected if bits 1 through 7 of this byte match bits 1 through 7 of the I2C_Address byte in the Configuration zone. Bit 0 of this byte is the standard I²C R/W bit and must be zero to indicate a write operation (the bytes following the device address travel from the host to the client).
Word Address	Word Address	This byte must have a value of 0x03 for normal operation.
Command	Data1, N	The command group, consisting of the count, command packet and the 2-byte CRC. The CRC is calculated over the size and packet bytes.

Because the device treats the command input buffer as a FIFO, the input group can be sent to the device in one or many I²C command groups. The first byte sent to the device is the count, so after the device receives that number of bytes, it will ignore any subsequently received bytes until the execution is finished.

The system must send a Stop condition after the last command byte to ensure that ECC608-TMNGTLS will start the computation of the command. Failure to send a Stop condition may eventually result in a loss of synchronization; see I2C Synchronization for recovery procedures.

8.2.1 Word Address Values

During an I²C write packet, the ECC608-TMNGTLS interprets the second byte sent as the word address, which indicates the packet function as it is described in the table below:

Table 8-2. Word Address Values

Name	Value	Description
Reset	0x00	Resets the address counter. The next I²C read or write transaction will start with the beginning of the I/O buffer.
Sleep (Low-power)	0x01	The ECC608-TMNGTLS goes into the low-power Sleep mode and ignores all subsequent I/O transitions until the next Wake flag. The entire volatile state of the device is reset.
Idle	0x02	The ECC608-TMNGTLS goes into Idle mode and ignores all subsequent I/O transitions until the next Wake flag. The contents of TempKey, MessageDigestBuffer and Alternate Key registers are retained.
Command	0x03	Writes subsequent bytes to sequential addresses in the input command buffer that follow previous writes. This is the normal operation.
Reserved	0x04 – 0xFF	These addresses must not be sent to the device.

8.2.2 I²C Synchronization

It is possible for the system to lose synchronization with the I/O port on the ECC608-TMNGTLS, perhaps due to a system Reset, I/O noise or other conditions. Under this circumstance, the ECC608-TMNGTLS may not respond as expected, may be asleep or may be transmitting data during an interval when the system is expecting to send data. To resynchronize, the following procedure can be followed:

1. To ensure an I/O channel Reset, the system must send the standard I²C software Reset sequence as follows:
   - A Start bit condition
   - Nine cycles of SCL, with SDA held high by the system pull-up resistor
   - Another Start bit condition
   - A Stop bit condition

   It may, then, be possible to send a read sequence, and if synchronization completes properly, the ECC608-TMNGTLS will ACK the device address. The device may return data or may leave the bus floating (which the system will interpret as a data value of 0xFF) during the data periods.

   If the device does ACK the device address, the system must reset the internal address counter to force the ECC608-TMNGTLS to ignore any partial input command that may have been sent. This can be accomplished by sending a write sequence to word address 0x00 (Reset), followed by a Stop condition.
2. If the device does not respond to the device address with an ACK, then it may be asleep. In this case, the system must send a complete Wake token and wait t_{WHI} after the rising edge. The system may, then, send another read sequence, and if synchronization is complete, the device will ACK the device address.
3. If the device still does not respond to the device address with an ACK, then it may be busy executing a command. The system must wait the longest t_{EXEC} (max.), then send the read sequence, which will be acknowledged by the device.

Related Links

Device is Asleep
Device is Awake

8.3 I²C Transmission from the ECC608-TMNGTLS

When the ECC608-TMNGTLS is awake and not busy, the host can retrieve the current output buffer contents from the device using an I²C read. If valid command results are available, the size of the group returned is determined by the particular command that is run. Otherwise, the size of the group (and the first byte returned) will always be four: count, status/error and 2-byte CRC.

Table 8-3. I²C Transmission from the ECC608-TMNGTLS

Name	I²C Name	Direction	Description
Device Address	Device Address	To Client	This byte selects a particular device on the I²C interface and the ECC608-TMNGTLS will be selected if bits 1 through 7 of this byte match bits 1 through 7 of the I2C_Address byte in the Configuration zone. Bit 0 of this byte is the standard I²C R/W pin and must be one to indicate that the bytes following the device address travel from the client to the host (read).
Data	Data1, N	To Host	The output group, consisting of the count, status/error byte or the output packet followed by the 2-byte CRC.

The status, error or command outputs can be read repeatedly by the host. Each time a `Read` command is sent to the ECC608-TMNGTLS along the I²C interface, the device transmits the next sequential byte in the output buffer. See the following section for details on how the device handles the address counter.

If the ECC608-TMNGTLS is busy, idle or asleep, it will NACK the device address on a read sequence. If a partial command is sent to the device and a read sequence `[Start + DeviceAddress(R/W == R)]` is sent to the device, the ECC608-TMNGTLS will NACK the device address to indicate that no data are available to be read.

8.4 Sleep Sequence

Upon completion of the use of the ECC608-TMNGTLS by the system, it is recommended that the system issue a sleep sequence to put the device into Low-Power mode. This sequence consists of the proper device address followed by the value of 0x01 as the word address followed by a Stop condition. This transition to the Low-Power state causes a complete reset of the device's internal command engine and input/output buffer. It can be sent to the device at any time when it is awake and not busy.

Related Links

Device is Asleep
Device is Awake

8.5 Idle Sequence

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

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

9. Electrical Characteristics

9.1 Absolute Maximum Ratings

Operating Temperature	-40°C to +85°C
Storage Temperature	-65°C to +150°C
Maximum Operating Voltage	6.0V
DC Output Current	5.0 mA
Voltage on any pin -0.5V to (V_{CC} + 0.5V)	-0.5V to (V_{CC} + 0.5V)
ESD Ratings:	
Human Body Model(HBM) ESD	>4 kV
Charge Device Model(CDM) ESD	>1 kV

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

9.2 Reliability

The ECC608-TMNGTLS is fabricated with Microchip’s high reliability CMOS EEPROM manufacturing technology.

Table 9-1. EEPROM Reliability

Parameter	Min.	Typ.	Max.	Units
Write Endurance at +85°C (Each Byte)	400,000	—	—	Write Cycles
Data Retention at +70°C	15	—	—	Years
Data Retention at +55°C	45	—	—	Years
Read Endurance	Unlimited			Read Cycles

9.3 AC Parameters: All I/O Interfaces

Figure 9-1. Wake Timing: All Interfaces

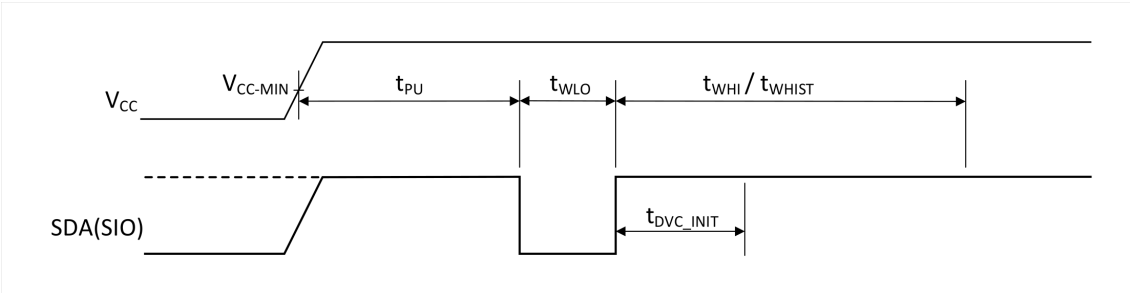


Figure 9-2. Input Noise Suppression: All Interfaces

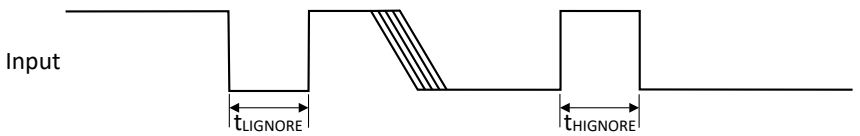


Table 9-2. AC Parameters: All I/O Interfaces

Parameter	Sym.	Direction	Min.	Typ.	Max.	Units	Conditions
Power-Up Delay	t_{PU}	To Crypto Device	100	—	—	µs	Minimum time between $V_{CC} > V_{CC}$ min prior to start of t_{WLO}.
Wake Low Duration	t_{WLO}	To Crypto Device	60	—	—	µs	—
Initialization Time(1)	t_{DVC_INIT}	Internal	—	—	400	µs	Maximum time from rising edge of wake pulse until device is initialized.(2)
Wake High Delay to Data Comm	t_{WHI}	To Crypto Device	1500	—	—	µs	SDA is recommended to be stable high for this entire duration unless polling is implemented. SelfTest is not enabled at power-up.
Wake High Delay when SelfTest is Enabled	t_{WHIST}	To Crypto Device	20	—	—	ms	SDA is recommended to be stable high for this entire duration unless polling is implemented.
High-Side Glitch Filter at Active(1)	$t_{HIGNORE\_A}$	To Crypto Device	45	—	—	ns	Pulses shorter than this in width will be ignored by the device, regardless of its state when active.
Low-Side Glitch Filter at Active(1)	$t_{LIGNORE\_A}$	To Crypto Device	45	—	—	ns	Pulses shorter than this in width will be ignored by the device, regardless of its state when active.
Low-Side Glitch Filter at Sleep(1)	$t_{LIGNORE\_S}$	To Crypto Device	15	—	—	µs	Pulses shorter than this in width will be ignored by the device when in Sleep mode.
Watchdog Time-out	$t_{WATCHDOG}$	To Crypto Device	0.7	1.3	1.7	s	Time from wake until device is forced into Sleep mode if Config.ChipMode[2] is 0.

Notes:

1. These parameters are characterized, but not production tested.
2. No communications, other than the wake pulse, on the I^2C bus is recommended until after t_{DVC_INIT} time has passed.

9.3.1 AC Parameters: I²C Interface

Figure 9-3. I²C Synchronous Data Timing

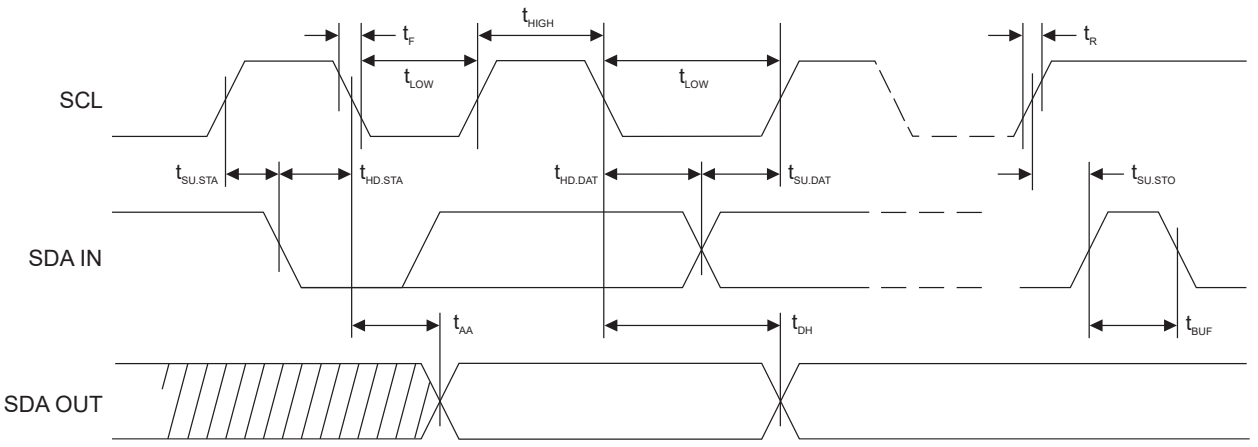


Table 9-3. AC Characteristics of I²C Interface**(2)**

Unless otherwise specified, applicable over recommended operating range from T_A = -40°C to +85°C, V_{CC} = +2.0V to +5.5V, C_L = 1 TTL Gate and 100 pF.

Parameter	Sym.	Min.	Max.	Units
SCL Clock Frequency	f_{SCL}	0	1	MHz
SCL High Time	t_{HIGH}	400	—	ns
SCL Low Time	t_{LOW}	400	—	ns
Start Setup Time	$t_{SU.STA}$	250	—	ns
Start Hold Time	$t_{HD.STA}$	250	—	ns
Stop Setup Time	$t_{SU.STO}$	250	—	ns
Data In Setup Time	$t_{SU.DAT}$	100	—	ns
Data In Hold Time	$t_{HD.DAT}$	0	—	ns
Input Rise Time[1]	t_R	—	300	ns
Input Fall Time[1]	t_F	—	100	ns
Clock Low to Data Out Valid	t_{AA}	50	550	ns
Data Out Hold Time	t_{DH}	50	—	ns
SMBus Time-Out Delay	$t_{TIMEOUT}$	25	35	ms
Time bus must be free before a new transmission can start[1]	t_{BUF}	500	—	ns

Notes:

1. Values are based on characterization and are not tested.
2. AC measurement conditions:
   - R_L (connects between SDA and V_{CC}): 1.2 kΩ (for V_{CC} = +2.0V to +5.0V)
   - Input pulse voltages: 0.3V_{CC} to 0.7V_{CC}
   - Input rise and fall times: ≤ 50 ns
   - Input and output timing reference voltage: 0.5V_{CC}

9.4 DC Parameters: All I/O Interfaces

Table 9-4. DC Parameters on All I/O Interfaces

Parameter	Sym.	Min.	Typ.	Max.	Units	Conditions
Ambient Operating Temperature	T_A	-40	—	+85	°C	Standard Industrial Temperature Range
Power Supply Voltage	V_{CC}	2.0	—	5.5	V	—
Active Power Supply Current	I_{CC}	—	2	3	mA	Waiting for I/O during I/O transfers or execution of non-ECC commands. Independent of Clock Divider value.
		—	—	14	mA	During ECC command execution. Clock divider = 0x0
Idle Power Supply Current	I_{IDLE}	—	800	—	µA	When device is in Idle mode, V_{SDA} and V_{SCL} < 0.4V or > V_{CC} – 0.4
Sleep Current	I_{SLEEP}	—	30	150	nA	When device is in Sleep mode, V_{CC} ≤ 3.6V, V_{SDA} and V_{SCL} < 0.4V or > V_{CC} – 0.4, T_A ≤ +55°C
		—	—	2	µA	When device is in Sleep mode. Over full V_{CC} range and -40°C to 85°C temperature range.
Output Low Voltage	V_{OL}	—	—	0.4	V	When device is in Active mode, V_{CC} = 2.5 to 5.5V
Output Low Current	I_{OL}	—	—	4	mA	When device is in Active mode, V_{CC} = 2.5 to 5.5V, V_{OL} = 0.4V
Theta JA	Θ_{JA}	—	166	—	°C/W	SOIC (SS)
		—	173	—	°C/W	UDFN (MA)

9.4.1 V_{IH} and V_{IL} Specifications

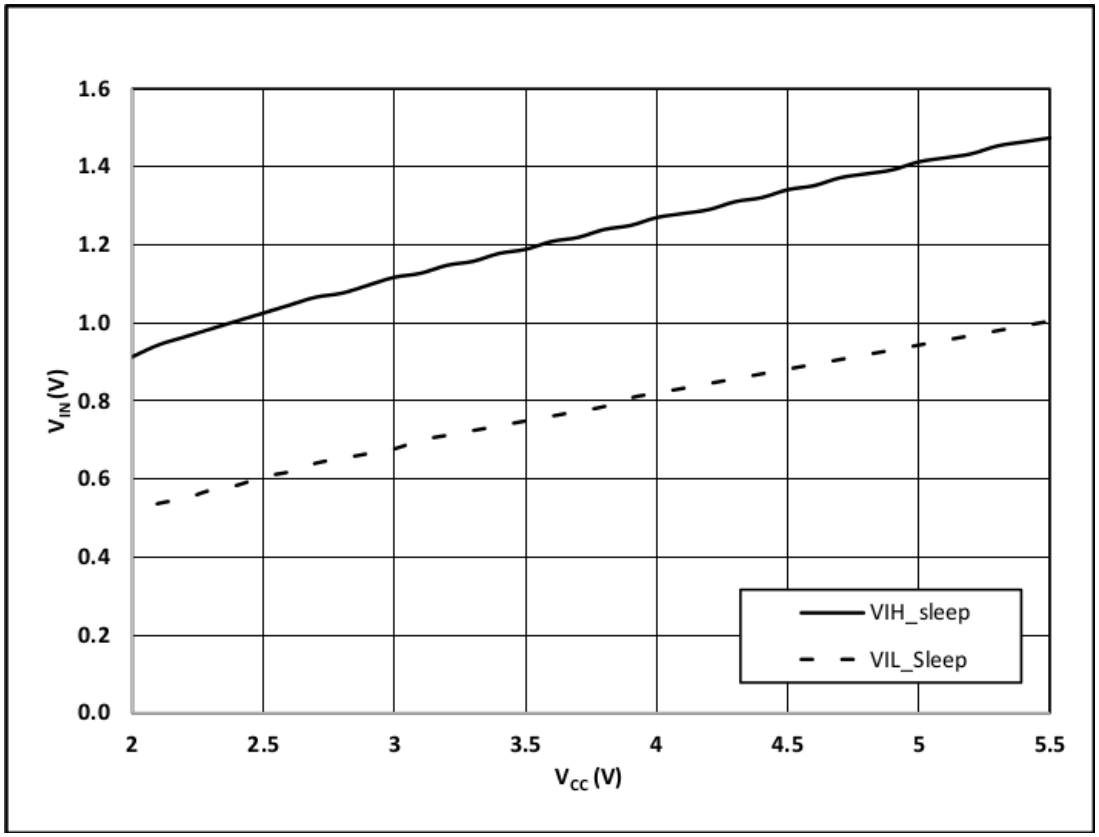
The input levels of the device will vary dependent on the mode and voltage of the device. The input voltage thresholds when in Sleep or Idle mode are dependent on the V_{CC} level as shown in Figure 9-4. When in Sleep or Idle mode the TTLenable bit has no effect.

The active input levels of the ECC608-TMNGTLS are fixed and do not vary with the V_{CC} level. The input levels transmitted to the device must comply with the table below.

Table 9-5. V_{IL}, V_{IH} on All I/O Interfaces (TTLenable = 0)

Parameter	Sym.	Min.	Typ.	Max.	Units	Conditions
Input Low Voltage	V_{IL}	-0.5	—	0.5	V	When device is active and TTLenable bit in Configuration memory is zero; otherwise, see above.
Input High Voltage	V_{IH}	1.5	—	V_{CC} + 0.5	V	When device is active and TTLenable bit in Configuration memory is zero; otherwise, see above.

Figure 9-4. V_{IH} and V_{IL} in Sleep and Idle Mode



10. Package Drawings

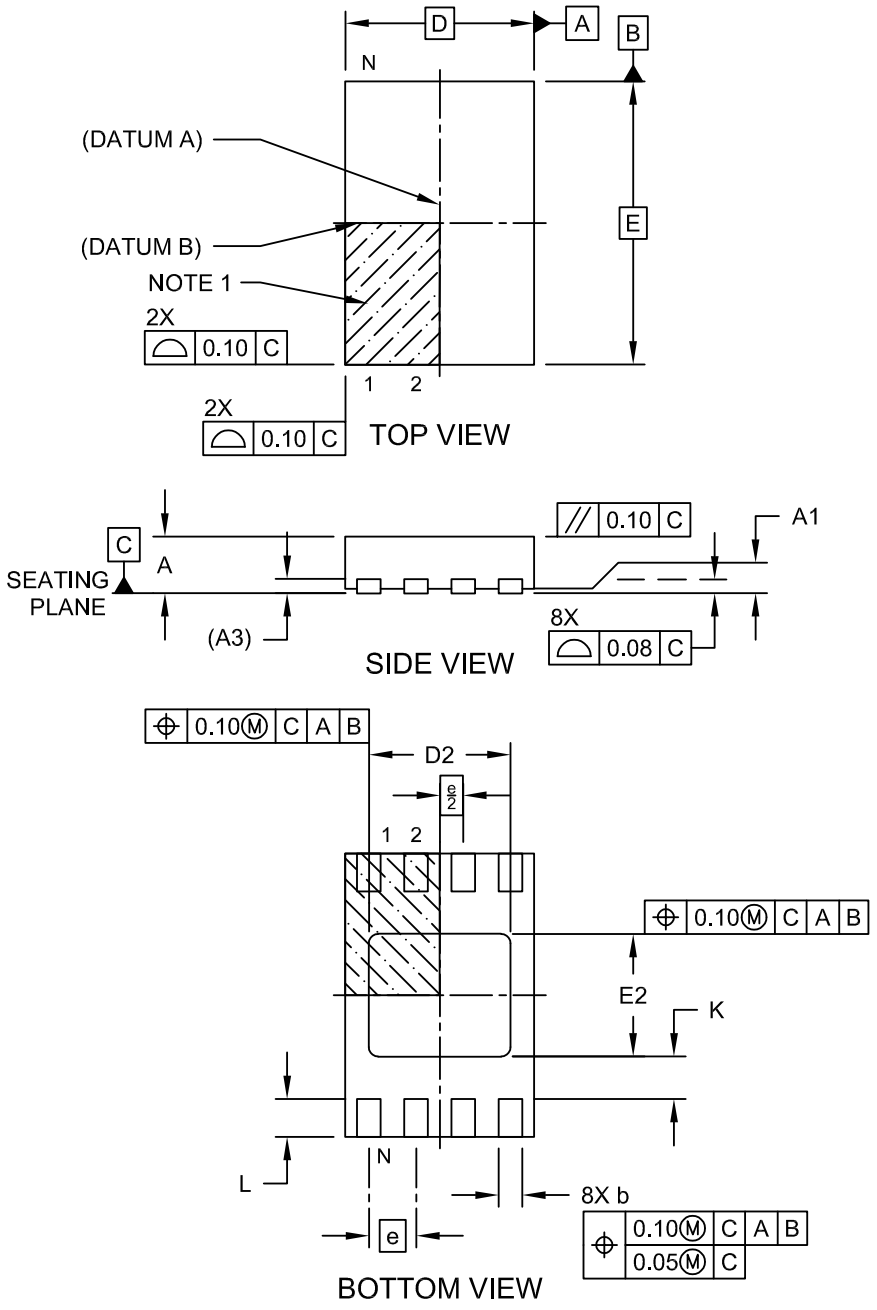
10.1 Package Marking Information

As part of Microchip's overall security features, the part marking for all crypto devices is intentionally vague. The marking on the top of the package does not provide any information as to the actual device type or the manufacturer of the device. The alphanumeric code on the package provides manufacturing information and will vary with assembly lot. It is recommended that the packaging mark not be used as part of any incoming inspection procedure.

10.2 8-Pad UDFN

8-Lead Ultra Thin Plastic Dual Flat, No Lead Package (Q4B) - 2x3 mm Body [UDFN]
Atmel Legacy Global Package Code YNZ

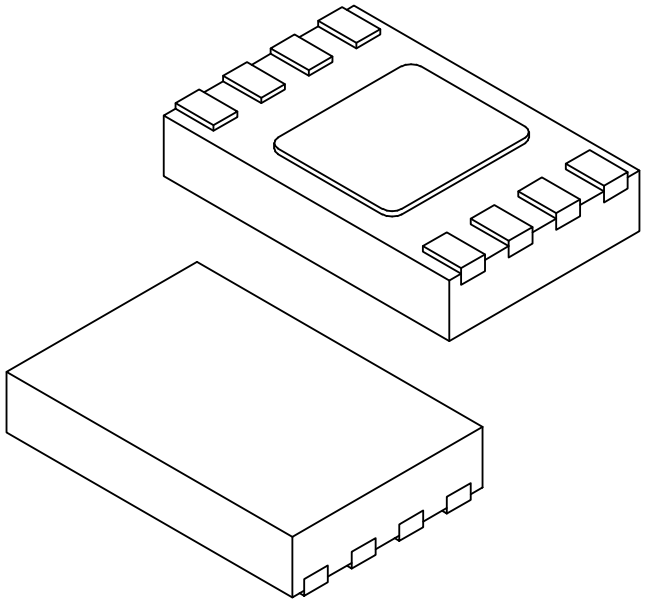
Note:	For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing C04-21355-Q4B Rev C Sheet 1 of 2

8-Lead Ultra Thin Plastic Dual Flat, No Lead Package (Q4B) - 2x3 mm Body [UDFN]
Atmel Legacy Global Package Code YNZ

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Terminals	N		8	
Pitch	e		0.50 BSC	
Overall Height	A	0.50	0.55	0.60
Standoff	A1	0.00	0.02	0.05
Terminal Thickness	A3		0.152 REF	
Overall Length	D		2.00 BSC	
Exposed Pad Length	D2	1.40	1.50	1.60
Overall Width	E		3.00 BSC	
Exposed Pad Width	E2	1.20	1.30	1.40
Terminal Width	b	0.18	0.25	0.30
Terminal Length	L	0.25	0.35	0.45
Terminal-to-Exposed-Pad	K	0.20	-	-

Notes:

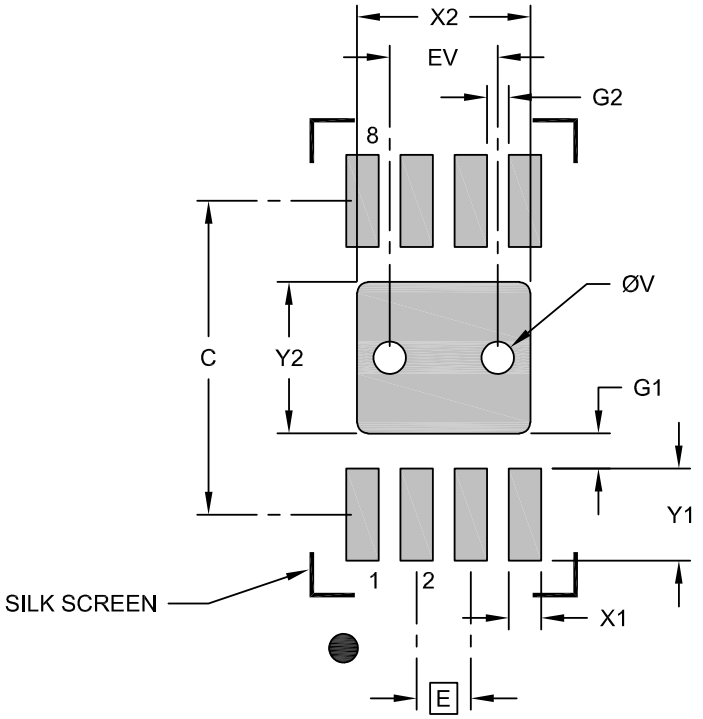
1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. Package is saw singulated
3. Dimensioning and tolerancing per ASME Y14.5M

BSC: Basic Dimension. Theoretically exact value shown without tolerances.
REF: Reference Dimension, usually without tolerance, for information purposes only.

Microchip Technology Drawing C04-21355-Q4B Rev C Sheet 2 of 2

8-Lead Ultra Thin Plastic Dual Flat, No Lead Package (Q4B) - 2x3 mm Body [UDFN] Atmel Legacy Global Package Code YNZ

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		0.50 BSC	
Optional Center Pad Width	X2			1.60
Optional Center Pad Length	Y2			1.40
Contact Pad Spacing	C		2.90	
Contact Pad Width (X8)	X1			0.30
Contact Pad Length (X8)	Y1			0.85
Contact Pad to Center Pad (X8)	G1	0.33		
Contact Pad to Contact Pad (X6)	G2	0.20		
Thermal Via Diameter	V		0.30	
Thermal Via Pitch	EV		1.00	

Notes:

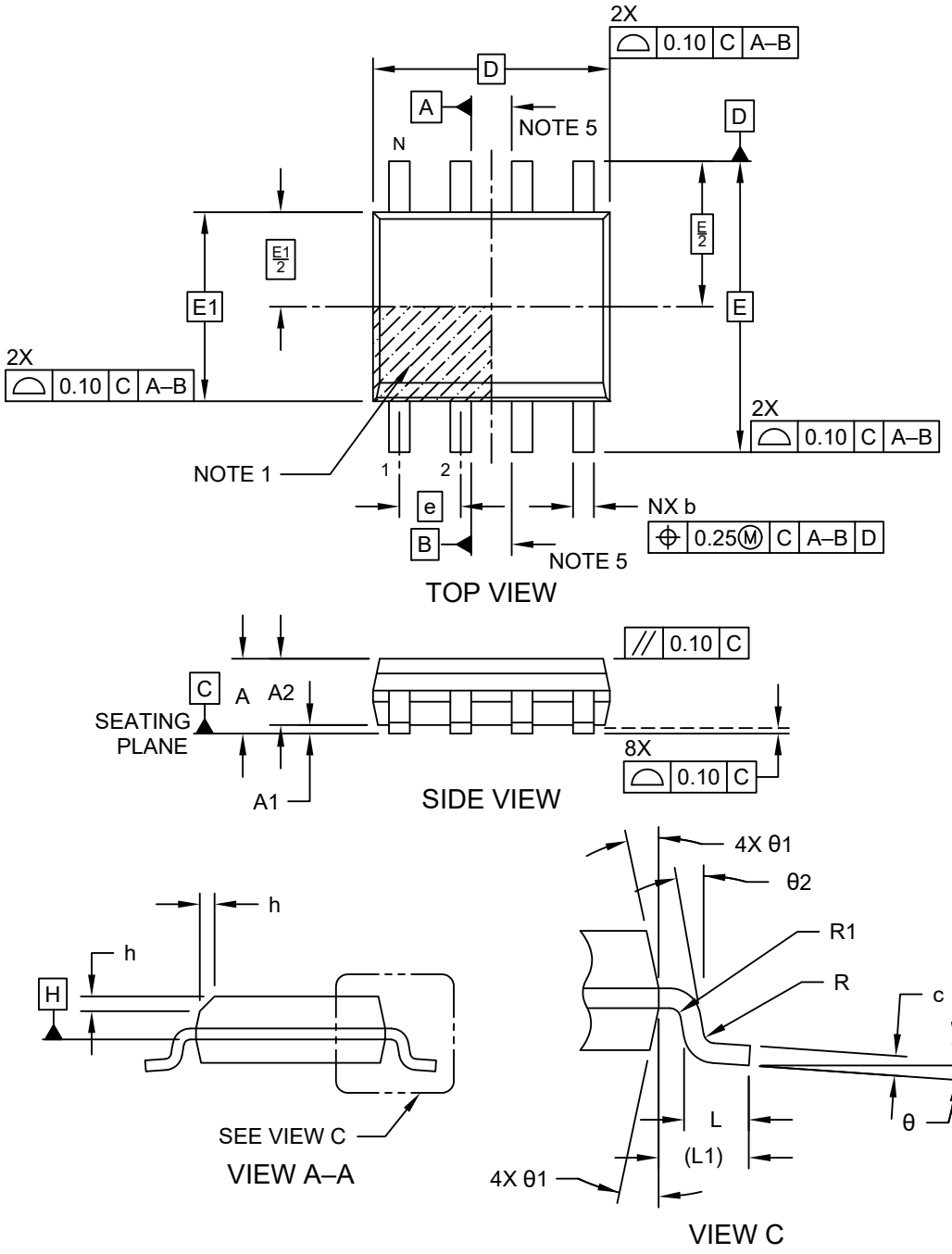
1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
2. For best soldering results, thermal vias, if used, should be filled or tented to avoid solder loss during reflow process

Microchip Technology Drawing C04-23355-Q4B Rev C

10.3 8-Lead SOIC

8-Lead Plastic Small Outline (C2X) - Narrow, 3.90 mm (.150 In.) Body [SOIC]
Atmel Legacy Global Package Code SWB

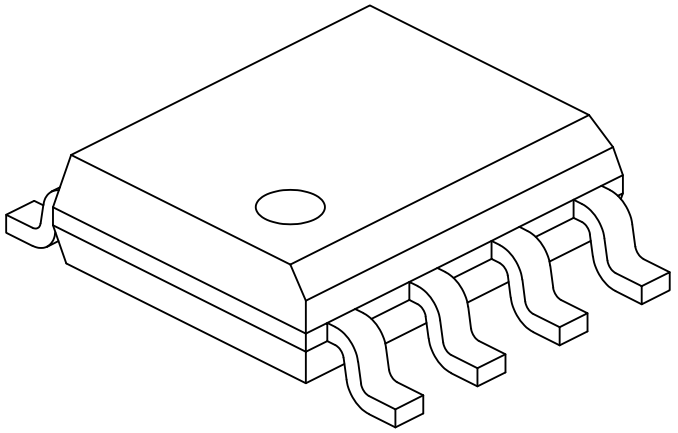
Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Microchip Technology Drawing No. C04-057-C2X Rev K Sheet 1 of 2

8-Lead Plastic Small Outline (C2X) - Narrow, 3.90 mm (.150 In.) Body [SOIC]
Atmel Legacy Global Package Code SWB

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Number of Pins	N		8	
Pitch	e		1.27 BSC	
Overall Height	A	–	–	1.75
Molded Package Thickness	A2	1.25	–	-
Standoff §	A1	0.10	–	0.25
Overall Width	E		6.00 BSC	
Molded Package Width	E1		3.90 BSC	
Overall Length	D		4.90 BSC	
Chamfer (Optional)	h	0.25	–	0.50
Foot Length	L	0.40	–	1.27
Footprint	L1		1.04 REF	
Lead Thickness	c	0.17	–	0.25
Lead Width	b	0.31	–	0.51
Lead Bend Radius	R	0.07	–	–
Lead Bend Radius	R1	0.07	–	–
Foot Angle	θ	0°	–	8°
Mold Draft Angle	θ1	5°	–	15°
Lead Angle	θ2	0°	–	–

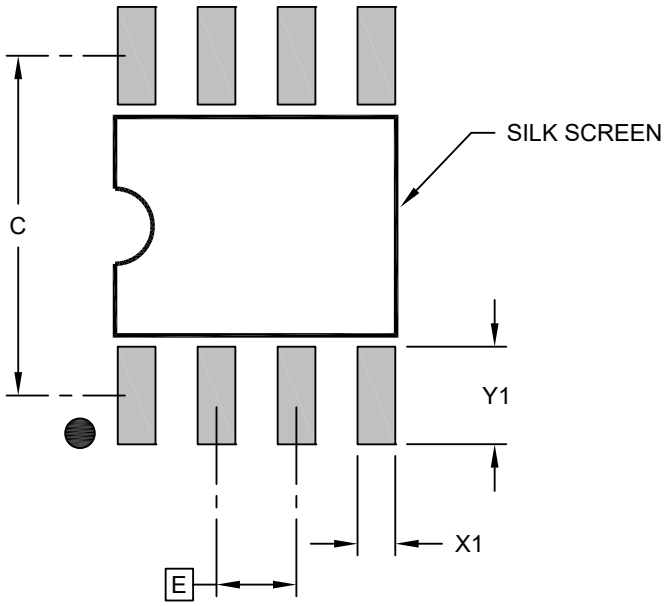
Notes:

1. Pin 1 visual index feature may vary, but must be located within the hatched area.
2. § Significant Characteristic
3. Dimensions D and E1 do not include mold flash or protrusions. Mold flash or protrusions shall not exceed 0.15mm per side.
4. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.
   REF: Reference Dimension, usually without tolerance, for information purposes only.
5. Datums A & B to be determined at Datum H.

Microchip Technology Drawing No. C04-057-C2X Rev K Sheet 2 of 2

8-Lead Plastic Small Outline (C2X) - Narrow, 3.90 mm (.150 In.) Body [SOIC]

Note: For the most current package drawings, please see the Microchip Packaging Specification located at http://www.microchip.com/packaging



RECOMMENDED LAND PATTERN

Units		MILLIMETERS		
Dimension Limits		MIN	NOM	MAX
Contact Pitch	E		1.27 BSC	
Contact Pad Spacing	C		5.40	
Contact Pad Width (X8)	X1			0.60
Contact Pad Length (X8)	Y1			1.55

Notes:

1. Dimensioning and tolerancing per ASME Y14.5M
   BSC: Basic Dimension. Theoretically exact value shown without tolerances.

Microchip Technology Drawing C04-2057-C2X Rev K

11. Product Identification System

To order or obtain information, e.g., on pricing or delivery, refer to the factory or the listed sales office.

PART NO	-	Trust Type	Trust Option	Package Type	-	Shipping Option
xxxxx	-	ttt	vvv	p	-	x

Device:	ECC608: Pre-configured Cryptographic Co-processor with secure hardware-based key storage	
Trust Type	TMNG	Type of Microchip Trust Product
Trust Option	TLS	Configuration associated with Trust Product.
Package Options	U	8-Pad 2 x 3 x 0.6 mm Body, Thermally Enhanced Plastic Ultra Thin Dual Flat No Lead Package (UDFN)
	S	8-Lead (0.150” Wide Body), Plastic Gull Wing Small Outline (JEDEC SOIC)
Shipping Option(1,2)		2k Minimum Order Quantity (MOQ) is required.
	B	10 Unit Bulk - Prototype Units

Examples:

- ECC608-TMNGTLSU: Trust Manager TLS, Provisioned, 8-UDFN, 2k MOQ, I^2C interface
- ECC608-TMNGTLSU-B: Trust Manager TLS, Provisioned Prototype, 8-UDFN, 10 units bulk, I^2C interface
- ECC608-TMNGTLSS: Trust Manager TLS, Provisioned, 8-SOIC 2k MOQ, I^2C Interface
- ECC608-TMNGTLSS-B: Trust Manager TLS, Provisioned Prototype, 8-SOIC 10 units bulk, I^2C interface

Notes:

1. Production orders will be delivered as Tape and Reel. Actual size of reel will vary based on customer order. 2k units is the minimum order quantity (MOQ) allowed.
2. Prototype units only come in 10-piece sample quantities with an either I^2C interface and can be provided in an SOIC or UDFN package option.

12. Revision History

Revision B (December 2024)

- Private Keys: Information related to Secondary Private Keys removed .
- Removed Section on AES Key Storage due to change in usage of configuration slots.
- 3.2.3 : Redefined slot 9 and changed slot 4 and 14 to type P. Application Specific Provisioned Slots have been removed.
- 3.2.4 : Modified the name on Slot 9.
- 6.3.1 : Modified to indicate AES keys can only be used from Slot 5.
- IP and Data Protection: Use case updated to drop reference to Slot 9 AES keys
- Back Matter Updated to Latest Microchip Format. Product Identification System moved to before revision history.

Revision A (March 2024)

- Original release of the document

Microchip Information

Trademarks

The “Microchip” name and logo, the “M” logo, and other names, logos, and brands are registered and unregistered trademarks of Microchip Technology Incorporated or its affiliates and/or subsidiaries in the United States and/or other countries (“Microchip Trademarks”). Information regarding Microchip Trademarks can be found at https://www.microchip.com/en-us/about/legal-information/microchip-trademarks.

ISBN: 979-8-3371-0166-8

Legal Notice

This publication and the information herein may be used only with Microchip products, including to design, test, and integrate Microchip products with your application. Use of this information in any other manner violates these terms. Information regarding device applications is provided only for your convenience and may be superseded by updates. It is your responsibility to ensure that your application meets with your specifications. Contact your local Microchip sales office for additional support or, obtain additional support at www.microchip.com/en-us/support/design-help/client-support-services.

THIS INFORMATION IS PROVIDED BY MICROCHIP “AS IS”. MICROCHIP MAKES NO REPRESENTATIONS OR WARRANTIES OF ANY KIND WHETHER EXPRESS OR IMPLIED, WRITTEN OR ORAL, STATUTORY OR OTHERWISE, RELATED TO THE INFORMATION INCLUDING BUT NOT LIMITED TO ANY IMPLIED WARRANTIES OF NON-INFRINGEMENT, MERCHANTABILITY, AND FITNESS FOR A PARTICULAR PURPOSE, OR WARRANTIES RELATED TO ITS CONDITION, QUALITY, OR PERFORMANCE.

IN NO EVENT WILL MICROCHIP BE LIABLE FOR ANY INDIRECT, SPECIAL, PUNITIVE, INCIDENTAL, OR CONSEQUENTIAL LOSS, DAMAGE, COST, OR EXPENSE OF ANY KIND WHATSOEVER RELATED TO THE INFORMATION OR ITS USE, HOWEVER CAUSED, EVEN IF MICROCHIP HAS BEEN ADVISED OF THE POSSIBILITY OR THE DAMAGES ARE FORESEEABLE. TO THE FULLEST EXTENT ALLOWED BY LAW, MICROCHIP’S TOTAL LIABILITY ON ALL CLAIMS IN ANY WAY RELATED TO THE INFORMATION OR ITS USE WILL NOT EXCEED THE AMOUNT OF FEES, IF ANY, THAT YOU HAVE PAID DIRECTLY TO MICROCHIP FOR THE INFORMATION.

Use of Microchip devices in life support and/or safety applications is entirely at the buyer’s risk, and the buyer agrees to defend, indemnify and hold harmless Microchip from any and all damages, claims, suits, or expenses resulting from such use. No licenses are conveyed, implicitly or otherwise, under any Microchip intellectual property rights unless otherwise stated.

Microchip Devices Code Protection Feature

Note the following details of the code protection feature on Microchip products:

- Microchip products meet the specifications contained in their particular Microchip Data Sheet.
- Microchip believes that its family of products is secure when used in the intended manner, within operating specifications, and under normal conditions.
- Microchip values and aggressively protects its intellectual property rights. Attempts to breach the code protection features of Microchip products are strictly prohibited and may violate the Digital Millennium Copyright Act.
- Neither Microchip nor any other semiconductor manufacturer can guarantee the security of its code. Code protection does not mean that we are guaranteeing the product is “unbreakable”. Code protection is constantly evolving. Microchip is committed to continuously improving the code protection features of our products.