element14 ESSENTIALS

# Secure MCUs for IoT Edge Applications

*Sponsored by* NXP



## 1. Introduction

As IoT data processing moves from the cloud to the edge, edge computing now plays a prominent role in the next generation of the Internet of Things (IoT). This focus on edge computing and architecture has created an increased need for microcontrollers to have enhanced integrated security, greater processing power, and dramatic power consumption improvements. This learning module covers the purpose, function, and challenges of IoT edge applications and edge device security, and will introduce you to NXP's LPC5500 single and dual-core 100MHz Cortex®-M33 microcontroller (MCUs) series, which are ideal for a wide range of IoT edge applications.

## 2. Objectives

*Upon completion of this learning module, you will be able to:*

- Understand the edge and the purpose of edge computing
- Be familiar with the trends and challenges in edge computing
- Describe the microcontrollers suitable for IoT edge applications
- Explain the main security features of the LPC5500 MCU series

## 3. What is the Edge?

An edge device is a piece of hardware located at the boundary of a network that handles data flow control or connectivity between disparate networks. It performs some standard functions such as transmission, monitoring, routing, processing, storage, filtering, and translation of data passing between networks. IoT edge devices collect data from sensors, communicate with each other, and can connect to the cloud directly or through a gateway edge device by way of wireless connectivity protocols like Wi-Fi®, ZigBee, and LoRa. Figure 1 shows an edge network where different sensors are connected with microcontroller-based wireless modules and communicate with the cloud through a gateway. The gateway/router/firewall are types of networking devices that connect a group of edge devices to the Internet and wide area network—allowing for data to flow seamlessly.

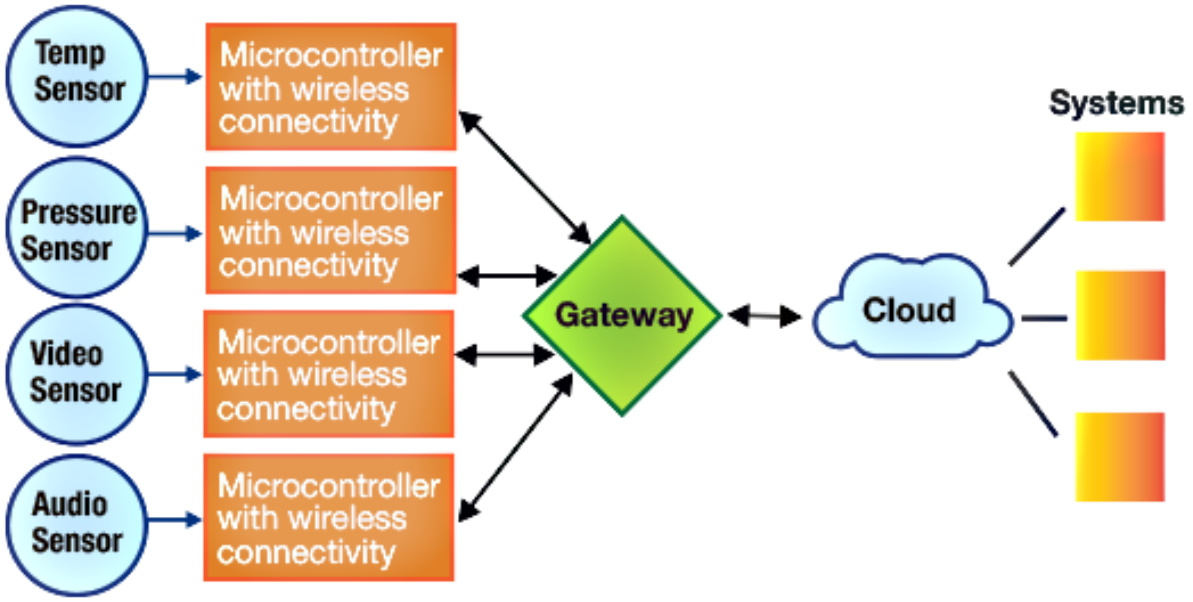


*Figure 1: Generic edge device network with gateway in an IoT ecosystem*

- ***3.1 Cloud vs Edge Computing***

In IoT applications, the data generated by end digital devices is growing exponentially as thousands of devices are being added in large IoT solutions. The traditional model of processing and storing data in the cloud has become too costly to meet the requirements of the end user. This has inspired a move towards edge computing, which processes device data closer to the source or end device. The edge has gradually grown to support advanced services, including wireless capabilities, Dynamic Host Configuration Protocol (DHCP) services, security functions, domain name system (DNS) services, and analytics.

IoT solutions have been implemented in critical application areas such as surveillance, automotive, healthcare, energy management, and more. While some of these areas can manage with delayed analytics in the cloud, some applications need a real-time response with low latency, especially for machine learning, artificial intelligence, and neural networks. As such, the ideal models for edge processing would be with a scalable hybrid architecture built on the cloud using machine learning, while the inference task is performed at the edge.

For applications such as audio or video recognition, where patterns and inferences need to be recognized instantaneously, it is not possible to stream all the data to the cloud where the artificial intelligence (AI) resides, because of the massive data and power restrictions. Edge-based AI is highly responsive in real time and has significant advantages, spanning greater security built into edge devices and less data flowing in and out of the network.

- ***3.2 Advantages of Edge Computing***

Edge computing provides a number of advantages that will allow developers to go beyond the constraints of cloud computing. In this section, let's discuss some of these advantages.

- **Reduced Network Latency:** When an IoT application requires quick responses, it is not possible to send large amounts of data to the cloud for processing and wait for a response for taking actions. For example, consider a safety-critical control system operating an industrial machine that must be stopped immediately if an operator is in a danger zone; the system must take action as soon as the sensor detects danger. The processing of human recognition and the execution of the decision to stop a machine should be performed at the edge due to reduced network latency.
- **Reduced Data Processing Cost:** The vast amount of data generated by sensors and actuators are not always relevant to a specific IoT application. For example, a temperature sensor generating a reading every second may not always provide information for an actionable response. Edge computing allows us to filter and process the data locally before sending it to the cloud, thus reducing the amount of data transmission, storage, and processing at the cloud, reducing the overall cost.

- **Strategic Use of Network Connectivity:** Most IoT edge deployments are done in remote installations where uninterrupted Internet connectivity might be a challenge. IoT implementations in a cloud environment are severely hampered if the network is interrupted or the available bandwidth is very low. Edge computing offers the ability of local computation, storage, and action without a network, while the important data can be transferred to the cloud when the network becomes available for further analysis.

- **Improved Data Privacy and Security:** Edge computing makes an IoT solution more secure because it decreases the number of devices connected to the Internet, reducing the exposure of data to the larger Internet. Data filtering on local edge devices reduces the amount of sensitive data being transmitted.

- **Reduced Energy Consumption:** Edge computing reduces energy consumption by transferring most of the processing and filtering of data away from the cloud to a local server on the edge. Also, reduced transmission of data throughput from edge devices saves energy for communication.

- **Reducing Impact of Cloud Disruptions:** By utilizing cloud computing in a distributed edge architecture, the impact of cloud network disruptions is limited.

## 4. Design And Development For The Iot Edge

The 'edge' brings forward various challenges for developers designing IoT architectures. In this section, we highlight some of the critical challenges of edge computing.

- **Privacy and Security:** A significant challenge in the deployment of the edge computing model is privacy and security. Edge device security is a big challenge, since the edge can be a convenient entry point to the network and core systems, making it vulnerable to cyberattacks. Beyond the threat of cyberattacks, physical security (tampering with a device) is also a threat that may not exist in the controlled environment of a data center. The technologies activated by the core of edge computation, such as peer-to-peer systems, wireless networks,and distributed systems must be secured while keeping in consideration that the interoperability and integration of devices must not be compromised. Moreover, specific data control access mechanisms should be implemented on edge frameworks to ensure data privacy.

- **Programmability:** One of the advantages of cloud computing is the infrastructure transparency to the user, because computing deployed only on the cloud and programs written in any language are compiled for a specific target platform. In edge computing, programs are written and deployed on edge devices, and there are a large number of embedded platforms from the many microcontroller manufacturers currently in the market. These devices need the development of customized application programs and have different runtime, which can cause difficulties for the programmer in writing an application for an edge computing model.

- **Standardization:** In edge computing, the number of edge devices is increasing exponentially. Each device on the edge needs a specific naming system for detecting the edge device, addressing, programming, and communication in the network system. At the present time the edge computing model has no efficient naming standard available. To communicate in a heterogeneous device network, edge designers need to learn various network protocols such as Bluetooth® Low Energy, ZigBee, Wi-Fi, and so on.

- **Data Abstraction:** In a well-connected home, there could be 50 devices that can sense, communicate, compute, and potentially actuate. An area of 1,000 houses could have about 50,000 devices producing vast amounts of data. A large portion of this data may be irrelevant, and hence should be deleted at the primary stage of data processing. Therefore, it is essential to abstract the data on the edge, and transfer only the necessary data to the gateway; this prospect is a significant challenge for edge computing. The microcontroller in edge devices needs to learn the specific algorithm to filter the data, and it should be able to predict the data to be sent to the gateway or cloud. Deciding the degree of abstraction is always a challenge, as some services or applications may be affected if too much raw data is filtered out. Edge devices should also have noise attenuation, event detection, and privacy protection features.

- **Services on the Edge:** In an IoT network such as a smart home, multiple services are deployed at the edge of the network, and each may have different priorities. For example, critical services (such as a fire alarm) should be processed earlier than regular services (such as data storage). In health-related services, heart failure detection should have a higher priority compared with another service such as entertainment. IoT is a dynamic system with new sensors and services being added regularly and existing services being improved for performance; microcontrollers used for edge devices should be able to detect and prioritize accordingly, be compatible with the upgrade requirements, and update the edge application on the fly.

## 5. Edge Architecture

Edge architectures can vary, but they generally use three types of components: edge sensors and actuators, edge devices, and edge gateways. Figure 2 shows the device hierarchy, with the cloud as the root, and edge gateways as a mediator above edge devices, and sensors and actuators located at the edge.

Edge sensors and actuators are devices which do not have processors. They are connected either directly to edge devices, gateways or via low power radio technologies. Edge devices are the intelligence for computation on data received from sensors, and they send commands to actuators. Edge devices are connected to the cloud either directly or through an edge gateway. Edge Gateways

run complete operating systems. They have more CPU power, memory, and storage. Gateways act as mediators between the cloud and the edge devices. Edge gateways and edge devices both forward selected subsets of pre-processed IoT data to services running in the cloud (e.g., machine learning, storage services, or analytics services), and receive commands from the cloud, like data queries, configurations, or machine learning models. An analytics module running in the cloud analyzes data coming from edge gateways and edge devices. A dashboard module can be deployed in the cloud to provide a global data view.

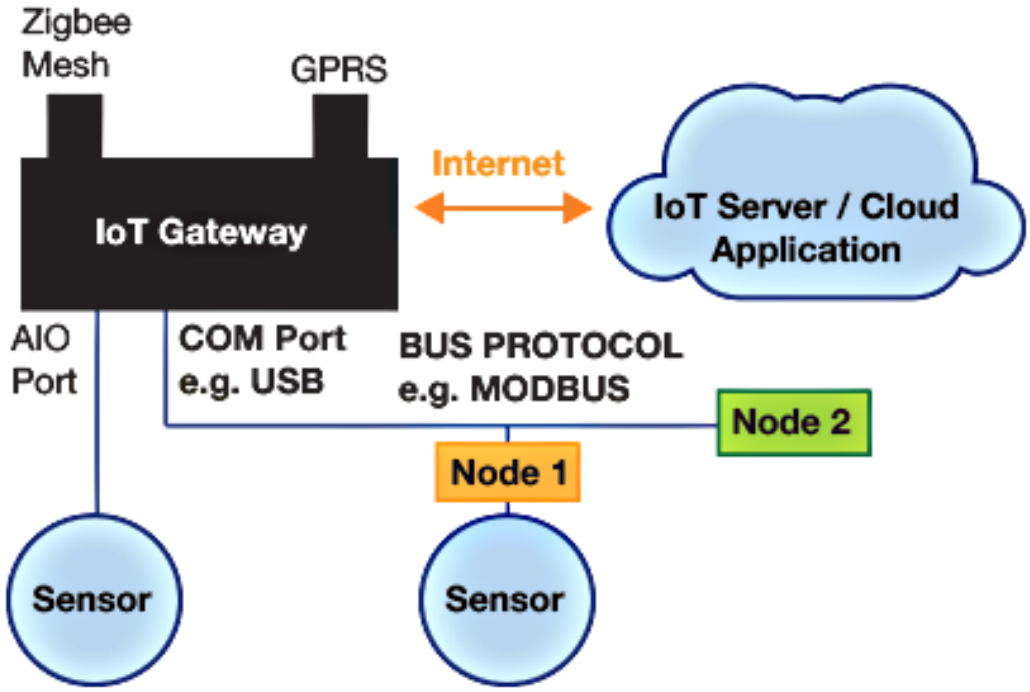


*Figure 2: IoT Edge Hierarchy*

## 6. Secure MCUs For IoT Edge Applications

With the introduction of new technology such as edge computation, microcontrollers are getting greater attention and updated designs as chipset manufacturers meet the growing requirements for IoT edge intelligence. These new secure MCUs for IoT edge applications offer low power and multiple connectivity options, as well as a combination of intelligence, security, and wireless capabilities. In this section, we will discuss NXP's LPC5500 MCU series, which offers secure edge computing at the software and hardware level, as well as essential technologies that enable low-latency, low-power, and high-throughput solutions for greater efficiency, privacy, and security.

- ***6.1 Overview***

The LPC5500 MCU series, the market's first Arm® Cortex®-M33-based MCU, offers product architecture enhancements and greater integration over previous NXP MCU generations. It offers power consumption improvements and advanced security features, including SRAM PUF (physically unclonable function) based root-of-trust and provisioning, real-time execution from encrypted images, asset protection with Arm TrustZone® technology, and on-chip memory with up to 640KB flash and

320KB SRAM to enable the efficient execution of complex edge applications. The LPC5500 series also provides dual-core Cortex-M33 capability with tightly coupled accelerators for digital signal processing and cryptography.

- ***6.2 Arm Cortex-M33 technology***

Cortex-M33 is the Arm processor which is applicable to IoT edge applications, with security being built into the CPU.  It is built for highly featured IoT and embedded products. Cortex-M33 offers a 20% performance improvement over Cortex-M3 and Cortex-M4 based MCUs. It uses Armv8-M architecture and a 32-bit instruction set with floating point and DSP capabilities for complex applications. In addition, the Cortex-M33 offers a dedicated co-processor interface for accelerating compute intensive operations. Cortex-M33 provides a range of new capabilities for designers, including machine learning inference on the edge. The following are some key advantages of Cortex-M33:

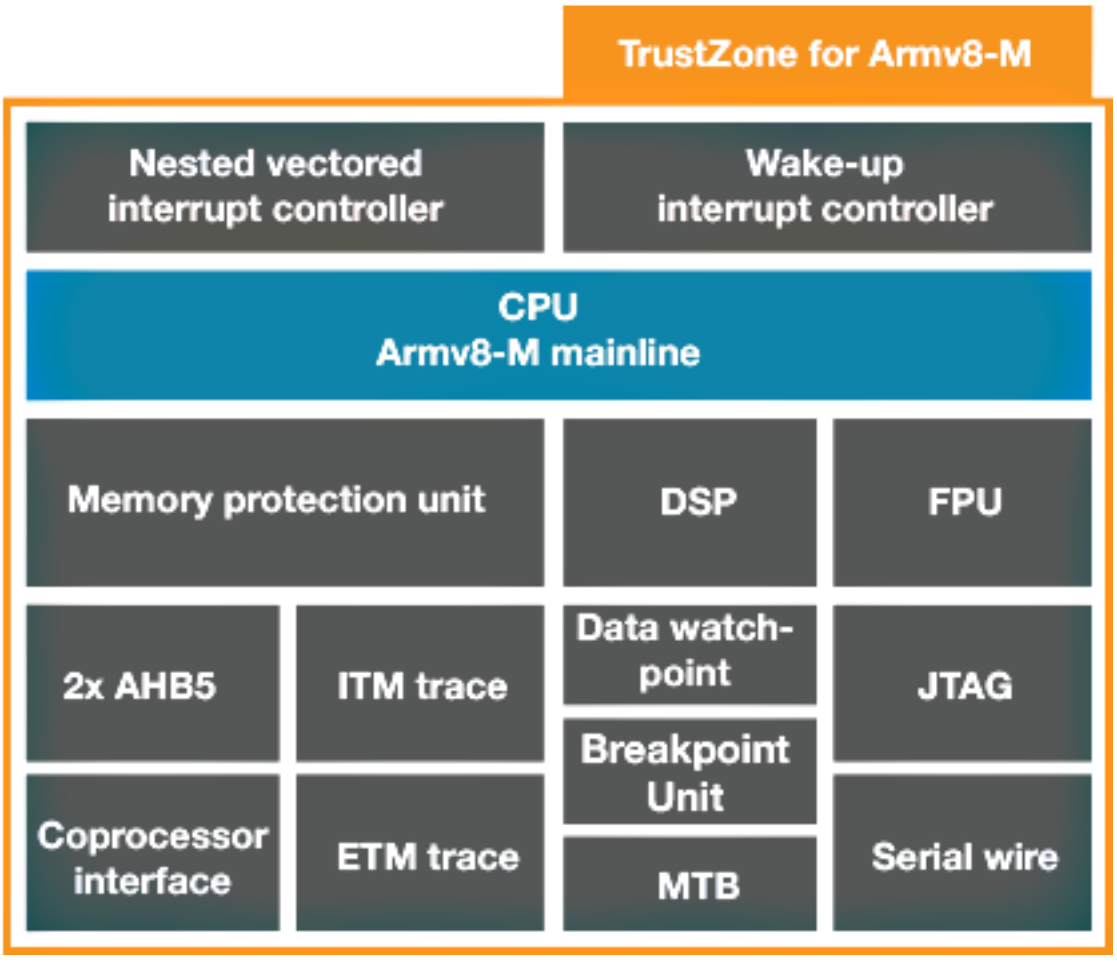


*Figure 3: Arm Cortex-M33 Block Diagram.*  Image Source: ARM

***TrustZone Security Isolation***

Arm TrustZone technology is a System on Chip (SoC) and CPU system-wide approach to security. TrustZone for ARMv8-M security extension is optimized for ultra-low power embedded applications. It enables multiple software security domains that restrict access to secure memory and I/O to trusted software only.

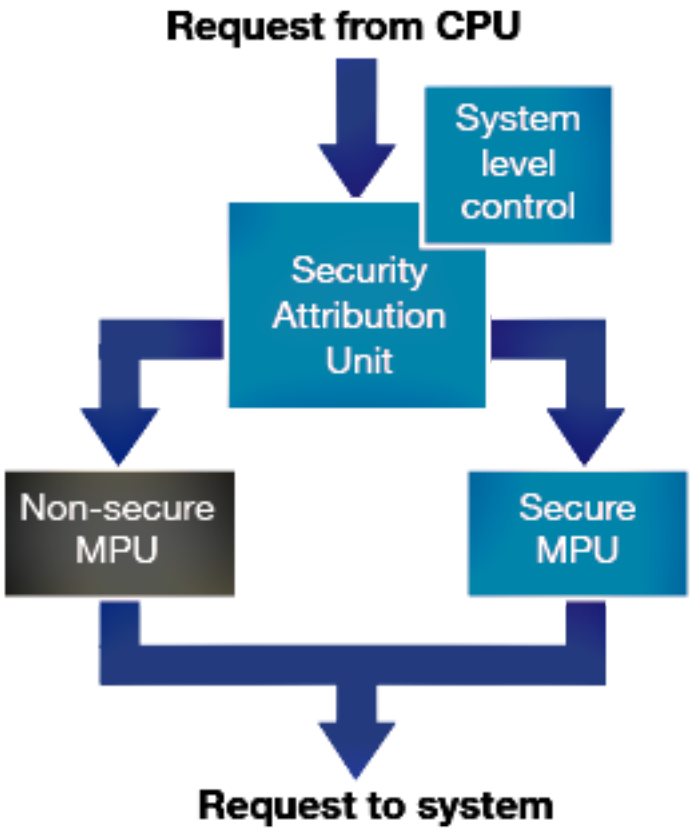


*Figure 4: TrustZone enables the system and the software to be partitioned into Secure and Non-secure worlds.* Image Source: ARM

TrustZone for Armv8-M is the foundation of security for embedded applications. It provides the means to implement separation and access control to isolate trusted software and resources to reduce the attack surface of critical components. TrustZone enables on a single CPU the system and the software to be partitioned into Secure and Nonsecure worlds, providing the benefits of lower device cost, real-time performance, low latency interrupts, efficient isolation, functional safety, and more.

In TrustZone, security is defined by the address. When a request comes from the CPU, the security attribution unit (SAU) decides at the system level whether the request should be considered a secure or non-secure address, and then sends the memory address to the secure or nonsecure memory protection unit (MPU) before sending it to the rest of the system (Figure 4).

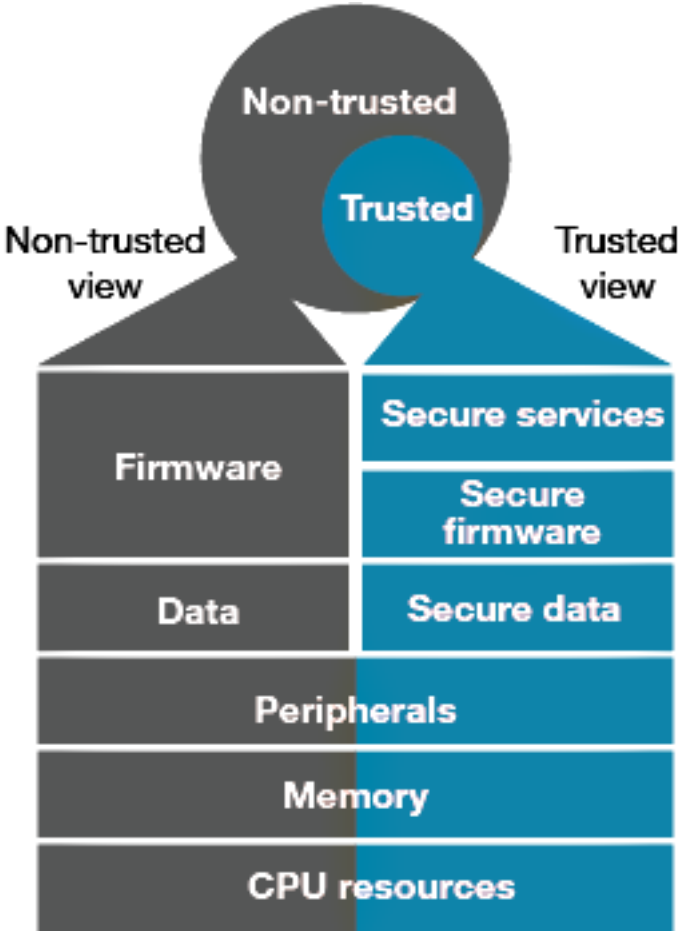


*Figure 5: One CPU with TrustZone for Armv8-M: processor partitioned into trusted and non-trusted worlds.* Image Source: ARM

Systems can be secured-by-design through placing only the most critical security routines, such as boot code, secure configuration, security keys, encryption libraries and firmware updates, in the secure TrustZone. The processor also supports the guidelines from the Platform Security Architecture (PSA), which is a framework to provide security for both hardware and software. PSA offers a consistent foundation for designers and developers working on IoT and embedded devices.

***Longer Battery Life***

The Cortex-M33 processor provides built-in low power features which allow designers to optimize power usage in IoT edge devices where power consumption plays a critical role, especially in battery-powered devices. It provides three low-power modes for saving energy to match processing demands. It has active and sleep modes that can operate down to 1.71V with very low power consumption.

***Development Tools***

These LPC5500 MCU series processors can be used with the Arm Keil® MDK tool suite and Arm Compiler development tools for programming. In addition, using the ULINKplus debug adapter with Keil MDK the application's power consumption can be analyzed to improve the energy-efficiency of the device in terms of both hardware and software.

- ***6.3 Security Enhancements***

Reliable embedded security is achieved by layering the protection through adding more hardware and software to create more layers. The LPC5500 series uses a multi-layered, hardware-enabled protection scheme. This layered security approach protects embedded systems in the following ways:

- Secure boot for hardware-based immutable root-of-trust
- Certificate-based secure debug authentication
- Encrypted on-chip firmware storage with real-time, latency-free decryption

These features in conjunction with Arm Cortex-M33 enhancements of Arm TrustZone for Armv8-M and memory protection unit (MPU) ensures physical and runtime protection with hardware-based, memory-mapped isolation for privilege-based access to resources and data.

- ***6.4 Secure Boot***

Device trustworthiness is attained within the LPC5500 series with a ROM-based secure boot process that utilizes device-unique keys to create an immutable hardware 'root-of-trust.' The keys can be locally generated on-demand by an SRAM-based Physically Unclonable Function (PUF), which uses the behavior of standard SRAM memory available in any digital chip to extract a unique pattern or “silicon fingerprint.” They are virtually impossible to clone or predict. This makes them very suitable for applications such as secure key generation and storage, device authentication, flexible key provisioning and chip asset management. Thus, closed loop transactions between the end-user and the original equipment manufacturer (OEM) are permitted, allowing the elimination of third-party key handling in potentially insecure environments.

***- 6.5 Secure Execution Environment***

A secure execution environment (SEE) is a secure and isolated region within a processor. It insures that the code and data loaded in the SEE are confidentially protected with integrity. A secure execution environment needs to have the following components:

- Secure isolation -- isolate secure and non-secure worlds
- Secure boot -- execute only authorized firmware
- Secure primitives -- cryptography primitives, including hashing, encryption, decryption, authentication
- Secure storage -- secure keys, code, and data confidentiality
- Secure update -- OTA firmware update, revoke keys, and anti-rollback
- Secure debug -- only authenticated parties allowed to debug

A secure execution environment improves the symmetric and asymmetric cryptography for edge-to-edge, and cloud-to-edge communication by generating device-unique secret keys through the SRAM

PUF. The security for public key infrastructure or asymmetric encryption is enhanced through the Device Identity Composition Engine security standard.

- ***6.6 LPC55S6x Microcontroller Family***
-

The LPC55S6x MCU family builds on the general-purpose Cortex-M33 based microcontroller introduced with the LPC5500 series. It leverages the Armv8-M architecture to introduce better performance and advanced security capabilities, including TrustZone-M and co-processor extensions. The LPC55S6x family enables these co-processors extensions and leverages them to bring significant signal processing efficiency gains from a proprietary DSP accelerator, offering a 10x clock cycle reduction. An optional second Cortex-M33 core offers flexibility to balance high performance and power efficiency.

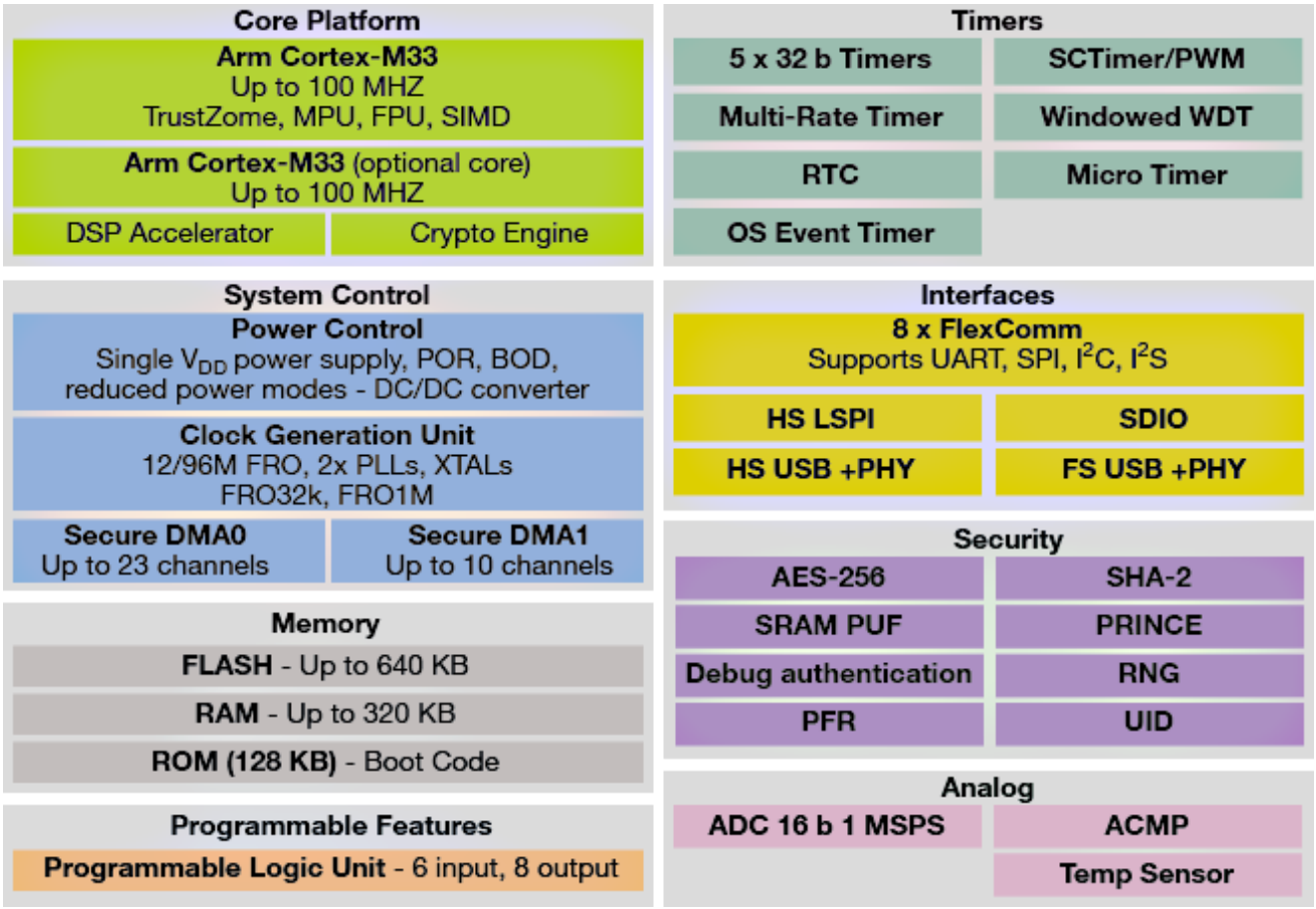


*Figure 6: 55S6x MCU Family Block Diagram*

LPC55S6x MCU devices feature a unique integrated security ecosystem providing layers of protection for embedded systems while protecting end products from unknown or unexpected threats over its life cycle, including SRAM PUF based root of trust and provisioning, real-time execution from encrypted images and debug authentication.

The following are some key security features implemented in the LPC55S6x MCU family:

- **Advanced Encryption Standard (AES):** LPC55S6x devices provide an on-chip hardware AES encryption and decryption engine for data encryption or decryption, data integrity, and proof of origin. Data is encrypted or decrypted by the AES engine using a key from the PUF, or software supplied key. The AES engine supports 128 bit, 192 bit, or 256 bit key in Electronic Code Book (ECB) mode, Cipher Block Chaining (CBC) mode, or Counter (CTR) mode. The AES engine supports 128-bit key in ICB (Indexed Code Book) mode that offers protection against side-channel attacks.

- **Secure Hash Algorithm (SHA):** LPC55S6x devices provide on-chip hash support to perform SHA-1 and SHA-2 with 256-bit digest (SHA-256). Hashing is a way to reduce arbitrarily large messages or code images to a relatively small fixed size unique number called a digest. The SHA-1 Hash produces a 160-bit digest (five words), and the SHA-256 hash produces a 256-bit digest (eight words).

- **SRAM PUF (Physically Unclonable Function) for key generation and identity.** PUF enables the secure generation of a unique device fingerprint and device-unique cryptographic keys. The unique and unclonable keys provide significant security benefits over other means of key injection or storage.

- **PRINCE:** LPC55S6x devices offer support for real-time encryption and decryption for on-chip flash using the PRINCE encryption algorithm. PRINCE is faster when compared to AES because it can decrypt and encrypt without adding extra latency. PRINCE operates as data is read or written, without the need to first store data in RAM, and then encrypts or decrypts to another space. This functionality is useful for asset protection, such as securing application code, securing stored keys, and enabling secure flash update.

- **ROM-based secure boot, debug authentication, and support for secure storage.** The ROM supports boot loading to internal flash and supports factory programming through an In System Programming (ISP) mechanism over I/O serial interfaces.

- **ARMv8-M TrustZone with secure and non-secure memory protection units and a Secure Attribution Unit (SAU).**

- **Implementation Defined Attribution Unit (IDAU), secure bus controller, DMA, and secure GPIO**

- **Crypto engine CASPER (Cryptographic Accelerator and Signaling Processing Engine with RAM) using the Cortex-M33 coprocessor interface.** The CASPER engine is a hardware accelerator capable of running asymmetric cryptographic such as Elliptic Curve Cryptography (ECC).
- **True Random Number Generator (TRNG).** The TRNG module is a hardware accelerator module that generates 256-bit entropy. The purpose of the module is to generate high quality, cryptographically secure, random data. Random number generators are used for data masking, cryptographic, modeling and simulation applications which employ keys that must be generated in a random fashion.

- **Protected Flash Region (PFR), a Customer Field Programmable Area (CFPA), and the Customer Manufacturer Programmable Area (CMPA) with the boot configuration**

### *-6.7 PowerQuad Hardware Accelerator*

PowerQuad is a hardware accelerator that consists of several internal computation engines: Transform engine, Transcendental function engine, Trigonometry function engine, Dual biquad IIR filter engine, Matrix accelerator engine, FIR filter engine, and CORDIC engine. With PowerQuad integrated within the LPC5500 MCU series, it can execute DSP tasks with better performance than the CMSIS-DSP library, which is implemented by pure software.

PowerQuad is integrated with the Arm Cortex-M33 co-processor interface. It can be accessed through the co-processor instructions. The PowerQuad hardware module is designed to accelerate some general DSP computing tasks, including the math functions, matrix functions, filter functions and the transform functions (including FFT). As the computing is totally executed by specific hardware other than the Arm core, it runs fast,  saves CPU time and offloads the Cortex-M33 cores for other tasks. The PowerQuad can be considered as a simplified DSP hardware but with less power consumption and well integrated inside the Arm ecosystem.



Take the Quiz