WHITE PAPER

Cortex-M Processors and the Internet of Things (IoT)

Why the processor matters? What are we doing to enable IoT and what are the challenges?

Joseph Yiu
Andrew Frame

January 2013

Abstract

In the last two years we have seen a vast increase in the number of Cortex-M microcontroller devices, many of them with Ethernet connectivity. In this paper we look at how the Cortex-M processor designs fit the requirements of Internet of Things (IoT) scenarios, and how the ARM ecosystems such as CMSIS and middleware are evolving to enable IoT development. We also look into the challenges of achieving IoT, and new opportunities created by the IoT demands.

(In the last two years we have seen a vast increase in the number of Cortex-M microcontroller devices, many of them with Ethernet connectivity. In this paper we look at how the Cortex-M processor designs satisfy the requirements of Internet of Things (IoT) scenarios, and how aspects of the ARM ecosystems such as CMSIS and the wide variety of middleware are evolving to enable IoT development. We also look at the challenges of achieving IoT, and new opportunities created by the IoT demands.)

Overview

For many embedded product developers, the Internet of Things (IoT) might still sound purely like a marketing buzzword. While we have not seen any real coffee machine product with Internet connectivity yet (excluding hobbyist prototypes), there are many more electronic devices or infrastructure systems with built-in Internet connectivity. Over the last few years we see a lot more microcontroller devices with Ethernet connectivity. Some of them have a PHY built-in, and some are available as a complete module with WIFI connectivity.

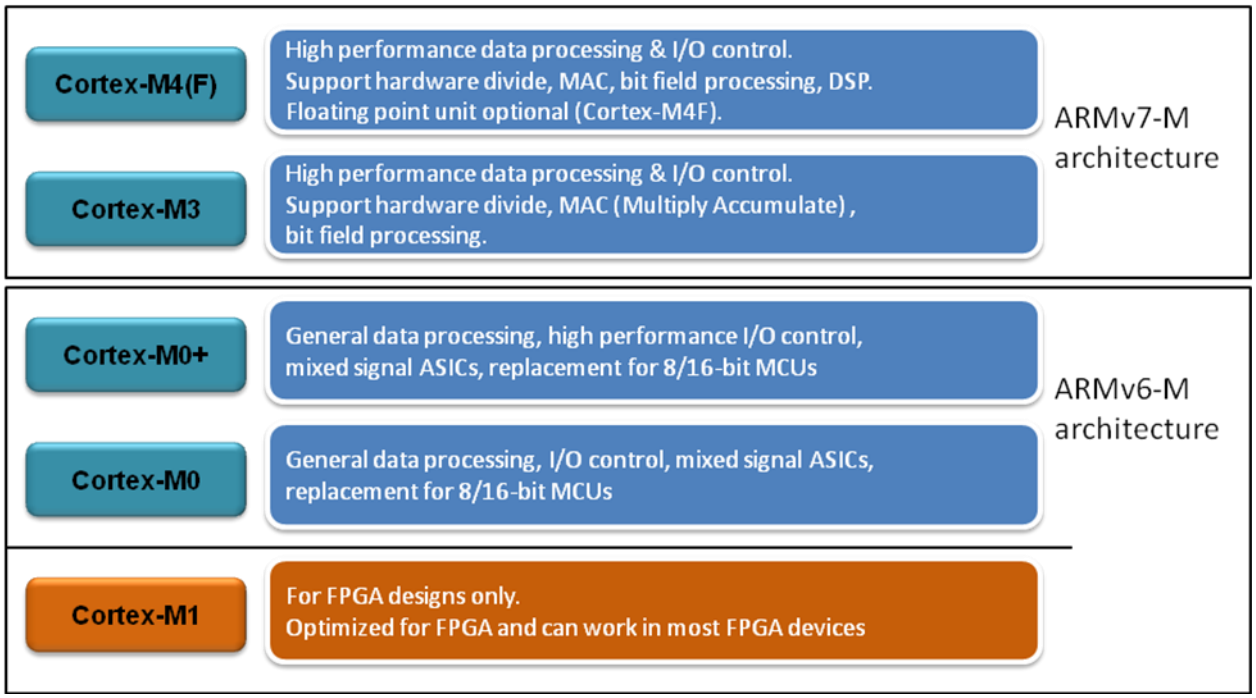
While it is possible to implement an Internet enabled embedded system using various types of processor architectures, some processor architectures are better than others in such application areas. In this paper we will analyse how the ARM Cortex-M processors match the requirements of IoT scenarios, what we are doing to support the necessary softwaredevelopment and the challenges in achieving wider IoT deployment.

While currently many Internet enabled embedded systems are implemented with application processors such as the ARM Cortex-A processors, ARM11 processors or x86 architecture, we will be focussing on

low cost microcontrollers in this paper because eventually, when the IoT market takes off, most of the IoT enabled devices will be based on low cost microcontrollers rather than application processors.

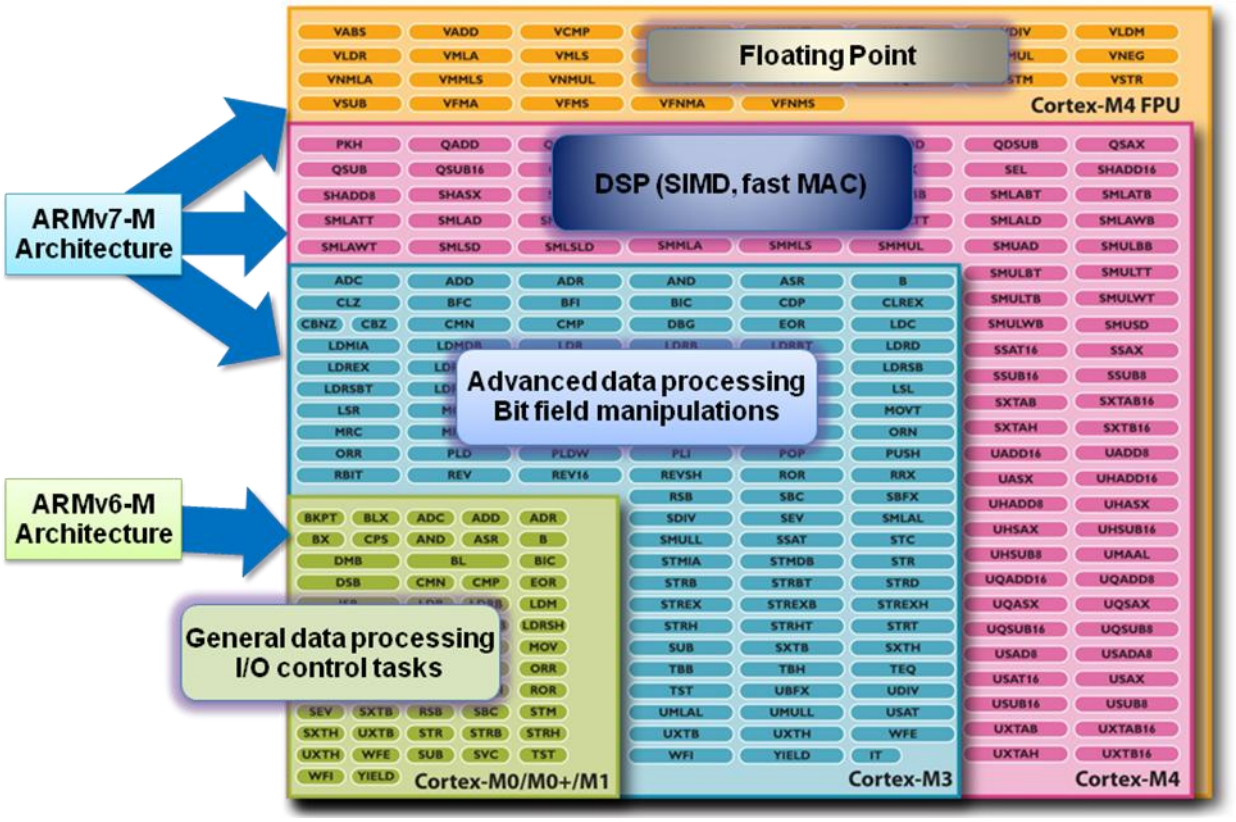
Background of the Cortex-M processors

The ARM Cortex-M3 processor was the first product to be developed in the Cortex-M processor family. Microcontrollers based on the Cortex-M3 were first released in 2006, and now there are five processors in the Cortex-M processor family, with 12 microcontroller vendors supplying microcontrollers based on the Cortex-M processors and thousands of devices available.



Different Cortex-M processor products support different ranges of instruction set. The Cortex-M0, Cortex-M0+ and Cortex-M1 processors are all based on the ARMv6-M architecture. The Cortex-M3 and Cortex-M4 are based on ARMv7-M architecture, which has a larger instruction set.

For general data processing and normal I/O control tasks, the ARMv6-M architecture is sufficient and provides the best energy efficiency. For more complex data handling, the processors from the ARMv7-M architecture such as the Cortex-M3 provide additional instructions which accelerate data processing, as well as providing additional instructions for hardware divide, bit field processing and Multiply-Accumulate (MAC). In more demanding data processing applications like Digital Signal Processing (DSP), the Cortex-M4 processor provides further instructions such as SIMD to enhance DSP performance. Some of the Cortex-M4 devices also include a floating point unit (i.e. Cortex-M4F) which provides optimized single precision floating calculation in the floating point hardware.



All of the Cortex-M processors provide the NVIC (Nested Vectored Interrupt Controller) for flexible and deterministic interrupt management, and provide comprehensive debug features based on CoreSight Debug Architecture.

Meeting the requirements of the IoT applications

The ARM Cortex-M processors are well suited to a wide range of applications, including IoT applications. In most typical IoT applications we can find the following processor system requirements:

- Low power
- High code density
- High performance
- Memory space and addressing modes
- OS friendly
- Security

Low power

From the users point of view many IoT systems are “always on”. In practice the processors or microcontrollers could be in sleep mode most of the time, and wake up regularly to decode received Ethernet data, process the required data and for OS task scheduling. Although many of them might be connected to the main electricity supply all the time, the energy consumption of the system can be considerable over a long period of time. The power wasted can be significant if the processor does not

have good energy efficiency or low standby power. And obviously, the requirement of low power is much more critical for battery powered products.

The Cortex-M processors are designed to be very energy efficient and provide various low power features including architectural sleep modes, clock gating support, multi-power domains, etc. This enables the microcontrollers designed with Cortex-M processors to maintain low active power as well as ultra low idle power during sleep modes, whilst not compromising on performance and features.

A processor with lower power consumption also has potential additional benefits by reducing the costs of the required power supply, and lower Electro Magnetic Interference (EMI). In many wireless and safety critical applications, low EMI is critical. Not only can it help the quality of the wireless communications, it also allows a reduction of the power required for data transmission, and reduced risk of causing interference problems to other devices which is especially important in medical and aviation applications.

High code density

All the Cortex-M processors are based on Thumb-2 Technology. The Thumb instruction set provides a mixture of 16-bit and 32-bit instructions under one operation state, and in most cases, the 16-bit instructions are used for better code density. The high code density allows the designer to implement a system with a microcontroller device using a smaller flash memory, which can reduce the cost of the IoT product. It can also reduce the power consumed by the microcontroller device as the flash memory size required is reduced.

Since most of the time 16-bit instructions are used, the processor often only needs to fetch instructions every other clock cycle. This reduces the power consumption of the system, as well as allowing more bus bandwidth for other data transfer operations. This can be important in devices with a fast Ethernet interface, as a DMA controller might be present in the system and need bus bandwidth to transfer data between the memory and the Ethernet interface.

High performance

All the Cortex-M processors are 32-bit processors, and offer much higher performance than many 8-bit and 16-bit microcontrollers. Although it might be possible to implement an IoT product using legacy 16-bit or even 8-bit microcontrollers, using 32-bit processors has a number of advantages:

With high processing power, it allows the IoT device to operate more reliably even if the amount of network activity generated by the end product might be unpredictable. If the network activity is higher than expected, an IoT device developed using a microcontroller with low performance might not be able to cope and crash. Using a 32-bit microcontroller provides sufficient performance headroom to handle unexpected network traffic. In some case, it might also provide a better chance for the IoT device to survive a DoS attack.

A higher performance microcontroller can also provide the opportunity to include extra security measures such as encryption and authorization. These features provide better security to the end users, and should be deployed if the IoT device handles any sensitive or personal data.

Obviously, better performance also enables better user experiences. For example, you can drive a graphic user interface easily and enable better feature set in your products.

Memory space and addressing modes

When the amount of internet activity increases in an IoT device, often it requires more memory to cope with the additional data for network packet processing. In addition, the increasing feature requirements of

IoT products means than you might need more memory space to accommodate the program image. As a result, having a larger addressable memory space can be beneficial for system reliability as well as the performance of an IoT product. However, in most 8-bit and 16-bit microcontrollers, the architecture only supports up to 64KB or 128KB of total memory space. This can become a major limitation for the potential features available in IoT products. In the Cortex-M microcontrollers, the architecture uses a 32-bit linear address and allows up to 4GB of total memory space. There is no need to use any memory paging technique, thereby allowing better performance and easier software development in Cortex-M microcontrollers when dealing with a larger memory space.

The Cortex-M processors support various address modes of memory access. These make the architecture C-friendly and hence provide good performance, as well as making it easy to program and debug.

OS friendly

In many IoT applications, it is essential to use an embedded OS to handle different tasks. The architecture of the Cortex-M processors is designed with OS support in mind. For example, the dual stack pointer mechanism allows an OS to be implemented easily, and at the same time allows efficient memory usage and reliable operation. The processors also include a 24-bit timer which is dedicated for OS System Tick interrupt generation. These features make the porting of an OS across different Cortex-M devices a simple task.

For safety critical embedded applications, the embedded OS can also utilize the optional Memory Protection Unit (MPU) to provide memory access isolation. This can prevent any application from corrupting data in other application task, hence greatly improve the reliability of a system.

Currently there are already more than 30 embedded OS available for the Cortex-M processors, and the number is still rising.

Security

Besides better reliability, the optional MPU also provides better security to the embedded system. When used together with the privileged and unprivileged execution levels, many of the tasks can be run in unprivileged mode to prevent security issues. For example, the TCP/IP stack can be run at unprivileged level. In this way, even if the TCP/IP stack has a vulnerability and gets attacked by a hacker, it is likely that the attacker can only gain unprivileged access and cannot access other memory areas protected by the MPU.

The high performance nature of the Cortex-M processors also makes it possible to implement sophisticated security measures in IoT. For example, an application can implement additional checking to validate the external inputs from the Internet, and handle encryption/decryption of the data transfers even without hardware AES accelerators. In addition, the higher performance also provides a better chance for an embedded device to survive a Denial-of-Service (DoS) attack.

The Cortex-M processors also have a fault exception handling capability, which triggers fault exception handlers when certain types of errors are detected. For example, when a system is under attack and an error condition has occurred, such as a branch to an invalid address space, the fault exception will be triggered and the fault handler can then carry out remedial actions such as restarting the system or killing a crashed application task.

The view from the system hardware level

The processor only occupies a small part of the silicon area in many modern microcontrollers. The rest of the area is often taken up by flash and SRAM, peripherals and other analog circuits. In order to allow a microcontroller to be used in IoT applications, the microcontroller designers also need to integrate various peripherals and features into the microcontroller products. Typically, these include connectivity such as an Ethernet or WIFI interface, or wireless communication interfaces such as Zigbee, RFID, etc.

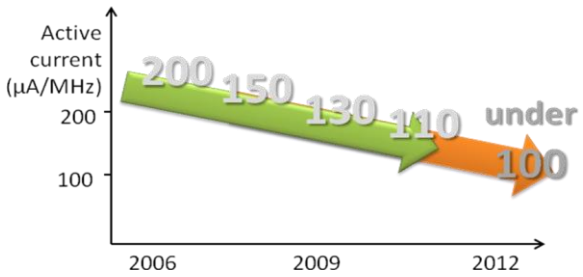
Richer connectivity → Higher processing requirement

Many microcontrollers also include features to enhance data-handling performance. For example, a DMA controller, a large data buffer in communication interface and multi-layer bus matrix for higher internal data bandwidth. In addition, many microcontroller vendors also include hardware accelerators for cryptography purposes such as AES accelerators. However, often it is still down to the processor to process the information and therefore the performance of the processor is still critical. The performance requirement might even be higher in products with an LCD display and Graphic User Interface (GUI), or in applications with additional connectivity such as USB, audio, memory cards, etc. The peak performance required for such systems can be significantly higher than traditional low cost microcontroller capability ranges.

Recently we have seen Cortex-M microcontroller products running at over 200MHz, and many others in the 100MHz frequency range. Combining this high frequency capability and the high execution efficiency, the Cortex-M based microcontrollers can be used in a wide range of IoT applications, and yet still have a very low power profile which is similar to traditional 16-bit or 8-bit microcontroller, and at a similar price point.

Low power at system level

Besides performance requirements, ease of system level integration is also often a key requirement. The Cortex-M processors support various low power interfaces to enable silicon designers to implement low power designs to achieve low operating power as well as idle power. Many Cortex-M microcontrollers available today need less than 200µA/MHz during operation.



Active current of low power Cortex-M microcontrollers

The latest member of the Cortex-M family, the Cortex-M0+ processor, has also included optimizations to further reduce the system level power consumption, for example, by reducing the number of flash memory accesses required for a given task.

Design scalability

With a 32-bit linear address space, the Cortex-M processors can be used to control a wide range of peripherals, whereas many 8-bit and 16-bit architectures might not be able to control vast number of peripherals, or might suffer performance reduction when memory paging techniques is used.

All the Cortex-M processors have an AHB Lite interface, an easy to use on-chip bus protocol which allows easy integration of memory and on-chip peripherals. It also makes it possible to implement designs with multiple processor cores with shared memory and shared peripherals. You can also add additional bus masters such as DMA controllers, or make some of the peripherals more intelligent which can initiate data transfers by themselves.

For development of a product series with a variety of processing requirements, you can easily switch between low cost microcontrollers with Cortex-M0/M0+ to high performance Cortex-M4 microcontrollers. The consistent architecture and CMSIS allow easily porting of application code across processors, and many microcontroller vendors now offer Cortex-M microcontrollers with compatible pin count to make this even easier.

For complex designs with multiple processor cores, the nature of multiple cores design can also lead to extra complexity in debugging. The Cortex-M processors incorporate the CoreSight Debug Architecture, which allows easy integration of a debug system for multiple cores. This allows the multiple cores in the design to be debugged with just one set of debug and trace connections.

Software challenges

From the hardware point of view, the current generation of ARM microcontrollers is easily capable of handling IoT applications. However, there are significant challenges which remain in software.

Software integration challenges

Besides the widest range of 32-bit microcontroller device choices from various vendors, the users of ARM microcontrollers can also take advantage of the broadest range of software offerings including development tools, OS and middleware. For example, a single IoT device might contain the following software:

- Product specific application code

- Device driver library from microcontroller vendor
- RTOS
- TCP/IP stack
- Wireless / wired communication stack
- Internet server/client software
- GUI library
- File system
- Codec for image/audio processing
- Cryptography library
- Toolchain specific runtime library functions

Currently there are more than 30 embedded OSes that can be used with Cortex-M microcontrollers, and you can use a compiler suite from over 10 vendors. There are also large numbers of companies providing or developing other middleware for ARM microcontrollers. The total ecosystem of the ARM microcontroller business is huge.

The task of integrating these software components into an application could be time consuming, and could impose potential risks caused by incompatibility between software components. To make it easier for embedded system developers, some microcontroller vendors developed a full set of libraries with an RTOS which can all work together out of the box. The MQX RTOS from Freescale is a good example.

However, for many other microcontroller users, they will have to integrate software components from different vendors. In order to allow better integration of software components and tools from different vendors, ARM has created the Cortex-M Software Interface Standard (CMSIS). The current CMSIS currently covers 5 areas:

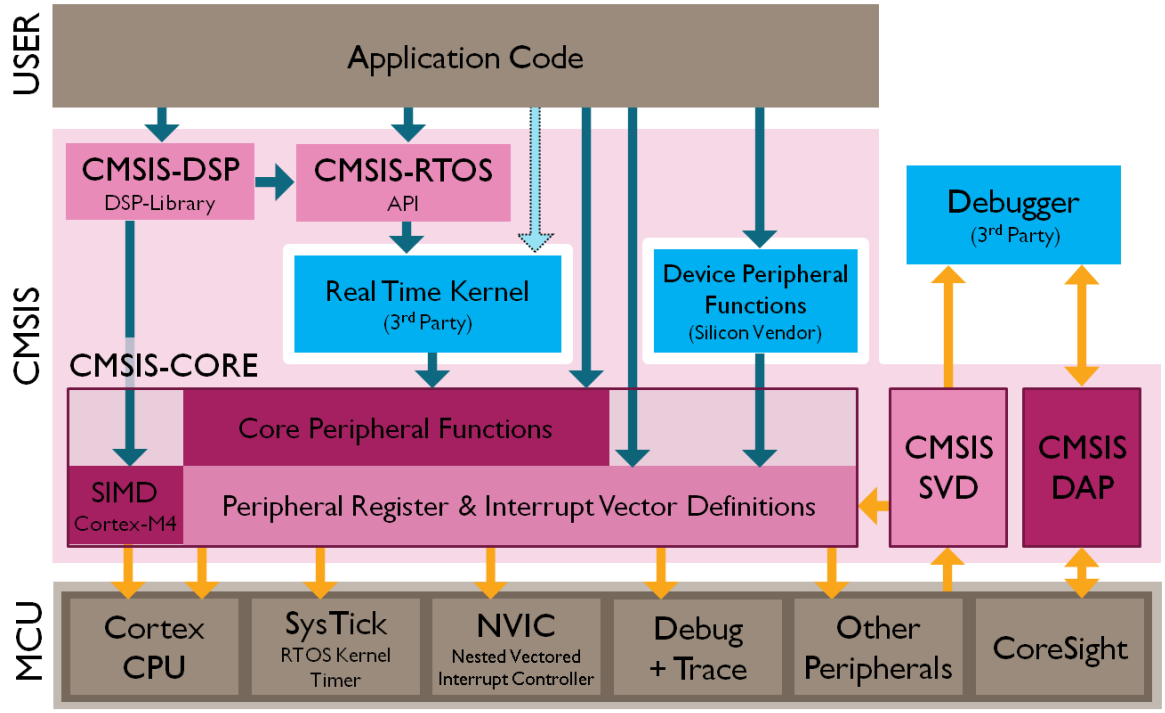
CMSIS-Core – a compiler independent API for software to access features on the processor such as interrupt control, system tick timer (SysTick) and special instructions.

CMSIS-DSP – a DSP library containing commonly used DSP and related functions optimized for Cortex-M4, and also supported on all Cortex-M microcontrollers.

CMSIS-SVD – System View Descriptions is an XML based data set that microcontroller vendors create for debuggers so that debuggers from different tool vendors can understand the available peripheral in a microcontroller device.

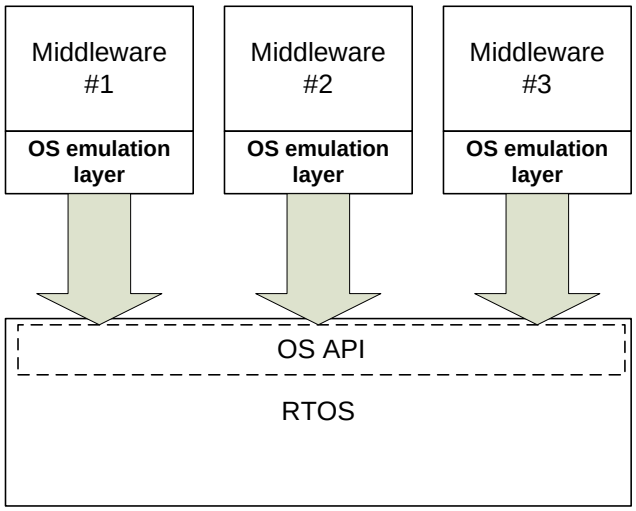
CMSIS-RTOS – A set of API for middleware and application code to access features such as task management in an embedded OS.

CMSIS-DAP – A reference USB debug adaptor design that can be ported to various microcontroller devices with USB interface.



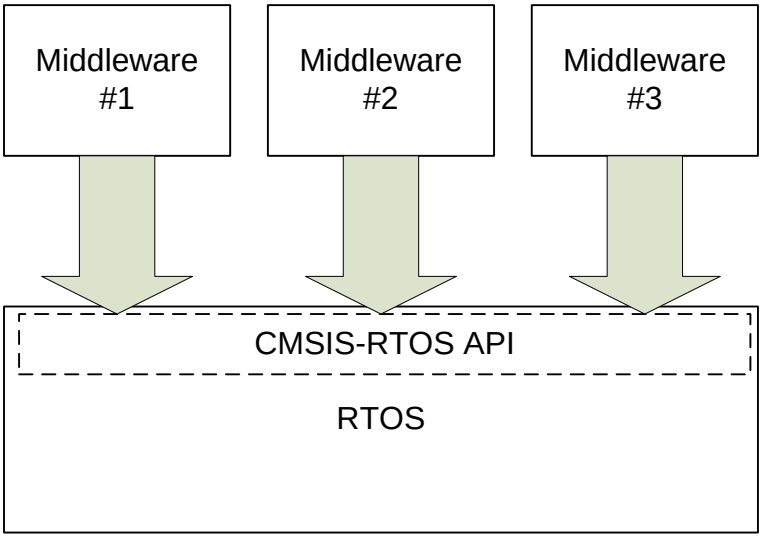
Overview of how various CMSIS projects work together

The CMSIS-RTOS is the recent addition to CMSIS. As software systems become more complex, many middleware products need to have interaction with the OS. Currently, many middleware packages which need to be RTOS-independent have to resort to an extra layer of OS porting interface. For example, LwIP has an OS emulation layer.



Currently middleware might need OS porting interface/OS emulation layer

However, implementing the OS porting might not be straight forward, and can increase software development time: imagine for each middleware component you are using for a project, you might need to develop an OS porting interface. For middleware vendors, it is equally difficult for them to make their middleware compatible with multiple embedded OS. In addition, trying to analyze problems faced by their customers can be equally difficult. As the ecosystem of the ARM Cortex-M microcontrollers expands, a consistent OS API is becoming a necessity.



CMSIS-RTOS : reduce the OS porting work for middleware

The CMSIS-RTOS is intended to solve this problem. It can be implemented natively on the OS, or implemented as a porting layer for an existing OS. The CMSIS-RTOS specification only specifies the API, not the exact implementations, and OS vendors can add their own addition features and APIs. Therefore embedded OS vendors can still make their OS differentiate from others.

IPv6 challenges

In order to enable full scale deployment of IoT devices, IPv6 technology play an important role. The challenges of IPv6 development include (and not limited to):

- TCP/IP stack
- Infrastructure
- Testing
- Performance
- Code size

TCP/IP stacks

Currently the number of TCP/IP stacks supporting IPv6 is limited to a few choices. For example, LwIP and uIP both support IPv6. Many other commercial TCP/IP stack vendors are currently working on IPv6 but it may take a few years for IPv6 to be more widely deployed in microcontrollers.

Infrastructure
When working on IPv6 system designs, you need to have network equipment that supports IPv6. While most modern PC platforms (Windows, Linux, Mac) all support IPv6, your home/business broadband routers, or even your ISP might not support this. There are various workarounds, for example, by setting up a gateway on your network to handle IPv6 to IPv4 tunneling. However, this can add additional complexity of setting up your development and testing environments.

Testing
While you can test your IPv6 enabled in a controlled environment, testing it in the field can be much harder. Partly because you might have no control over which TCP/IP protocol will be used. In addition, while there are lots of tools to help you testing IPv4, there are a limited number of tools for testing IPv6. Of course, this will change over time.

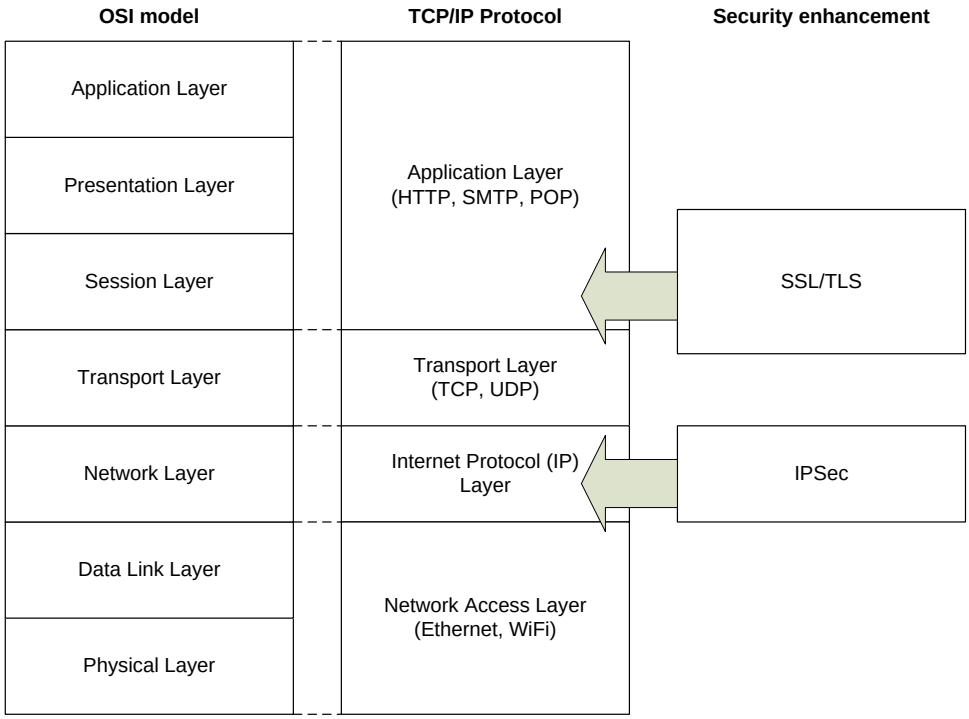
Performance
There can be two aspects related to performance: the first one is the performance of the microcontrollers will have to cope with the dual stacks implementation, and the second one is the performance and loss of IPv6 communication could be worst then IPv4 in some cases[1][2]. As a result, you might need to add additional headroom for communication performance/latency in your designs.

Code size
In order to implement dual stack, you need more program codes. While this is not a problem for PC or mobile computing platforms, this can be a problem for microcontrollers where the code size is generally limited to a smaller space. The code size can be significantly higher if secure communication protocol is needed. For this case, the high code density and 32-bit addressing of ARM microcontrollers can be a significant advantage over 16-bit and 8-bit microcontrollers.

Security challenges
Security is critical for the majority of internet connected devices. Unlike traditional personal computers, many embedded devices are always on, and connected to the internet all the time. Also, it is impossible to install firewall software on these devices. The migration to IPv6 also means that more and more IoT devices will be directly connected to the internet with unique IP address, instead of sheltered in a local network behind NAT (Network Address Translation).

In order to help improving security, many microcontrollers have built-in AES encryption accelerators and random number generators to accelerate encryption and decryption. On the software side, more and more software solutions are available to support secured connection such as Secure Sockets Layer (SSL) and Secure Shell (SSH), or even at IP level such as IPSec. However, software developers often still have to port these security software layers to utilize the security features (crypto accelerators). This is often a difficult task for inexperienced engineers.

[1] https://labs.ripe.net/Members/emileaben/measuring-world-ipv6-day-comparing-ipv4-and-ipv6-performance
[2] http://www.caida.org/workshops/isma/1202/slides/aims1202_acox.pdf



Typical methods for improving security of IoT applications

Note: SSL allows a secure connection to be established on top of normal TCP/IP connection, and IPSec allows encrypted data transfers at a lower communication level

To enable an embedded system to carry out secure connections such as SSL, a 32-bit processor is almost a must-have due to the code size and processing speed requirements. Typically the SSL code size is at least 50KB, and the overall code sizes can easily exceed 128KB[3].

On a completely different aspect, the limited MPU support on embedded OS is another challenge for security. In order to take full advantage of the memory protection feature, the embedded OS should implement MPU support. However, currently only a few embedded OS have MPU support.

Standardization

Openness is a key aspect of IoT system designs. In order to enable different parts of IoT systems to work together, standardization is essential. The required standardization is not only limited to the communication protocols, but also in the transaction level and up to the application levels.

For example, in order to enable sensor devices from one company to be connected to a gateway from another company, and enable the gateway to be connected to servers built by other third parties, standardized communication protocols between these systems are needed. Of course, the designs of the protocols not only need to be usable and scalable, but also need to be energy efficient, secure and low cost.

[3] http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/APPLICATION_NOTE/DM00024859.pdf

Currently there are varies activities in different organizations looking into establishing standards to enable a common platform for IoT application development:

- OPC Foundation. An industrial focus organization that has developed an architecture for scalable M2M (Machine-to-Machine) connectivity, including communication from small embedded devices like sensors to remote servers.
- The Internet of Things Initiative (www.iot-i.eu/public), an EU based program trying to bring various IoT communities together.

In 2012 ARM formed a UK industrial group with a number of companies including EnLight , Neul, Alertme and AquaMW to look into what needs to be done to enable IoT development. Also, ARM has joined the Weightless Special Interest Group, which is defining a new specification for low cost, low power radio standards for IoT devices.

Of course there are many companies developing various IoT solutions. Unfortunately, many of those activities are narrowly focused or just market exercises to push the products of those companies, without the aims to establish an open IoT ecosystem.

Opportunities

The future of IoT is full of challenges, but at the same time there are plenty of opportunities. Here we list a few obvious areas which have plenty of development potential.

Middleware & IPv6 stacks

To allow a wide range of potential IoT applications, there is a high demand for middleware and IPv6 capable TCP/IP stacks. Other middleware such as file systems, images and audio processing, VoIP, various servers and clients, network diagnosis tools and middleware for connectivity (e.g. USB, CAN) can also be in huge demand. Recently Avnet has worked together with ARM and setup up an online store for embedded software[4].

The landscape for middleware could also be changed significantly. Instead of having separate middleware components, there is a demand for a higher level of integration of middleware components because integration of various software components takes significant time and effort. The result is that IoT software platforms include OS, TCP/IP, security, and potentially GUI and multi-media components. This change might happen in several different ways:

a) silicon vendors might develop their own IoT platforms
b) silicon vendors might work with a number of middleware suppliers to produce an IoT platform
c) several middleware vendors might work together to produce IoT platform.

With the ongoing effort on trying to standardize communication protocols for IoT applications, additional middleware could be needed to enable these protocols to be implemented.

IoT modules and packages

In addition to software platforms for IoT applications, we might also see a lot more companies developing ready-to-use IoT platforms with both software and hardware. For example, WI-FI modules for microcontrollers, WI-FI node[5], board support packages, embedded system modules with preconfigured middleware, etc. These allow much faster concept-to-market development and can greatly reduce project risks for product developers.

[4] Avnet Embedded Software Store (www.embeddedsoftwarestore.com).
[5] http://www.edn.com/article/521814-Former_Apple_Google_Facebook_engineers_launch_IoT_startup.php

Microcontrollers
The IoT market provides plenty of chances for microcontroller vendors to establish product feature differentiations. Currently many Cortex-M microcontrollers with an Ethernet interface already provide various features such as hardware features to accelerate cryptography operations. Some of the microcontroller designs also provide innovative smart peripherals which can handle limited data handling without CPU intervention. In addition, they can also provide various software packages such as

- software code to accelerate commonly used SSL software with their specific crypto accelerators
- Protocol stack for connectivity peripherals (e.g. Bluetooth, USB, RFID)
- Device drivers for commonly used TCP/IP stacks
- Pre-packaged IoT software platforms

For these to happen, we can expect more collaboration between microcontroller vendors and middleware providers, as well as joint projects between multiple middleware suppliers.

Tools vendors
Software development tools are an essential part of this ecosystem. Various tools might be needed for IoT product development such as network traffic generation and analysis tools. Potentially, debuggers could have some of these utilities built in.

Training
The ability to develop IoT applications will be in high demand, especially around implementation of network security. For many embedded software developers with limited experience, starting to learn IoT programming might seem to be a big jump. However, it is likely to be well worth the investment and time.

The IoT can also be an interesting opportunity for companies that provide software training.

Conclusions
Do we have a conclusion? Well… yes and no…

We are expecting a large increase in IoT products in the next decade, and the capabilities provided by ARM processors are well suited to most of these applications. The ARM Cortex-M processor family meets the requirements of many IoT applications, and allows product designers to design a wide range of IoT products from low cost simple designs to high performance feature-rich products.

While the microcontroller devices are perfectly capable of running IoT applications, there are still lots of challenges in IoT product designs. The IoT is a new area for the microcontroller industry, and some of the technologies required for IoT applications might not be fully mature or not widely available. The diversity of the embedded software ecosystem also creates various challenges for product designers. But for the same reason, the IoT also provides a lot of opportunities for various parties in the ARM ecosystem as the designers might need new middleware, tools and knowledge.

Innovation happens all the time in the electronics industry. With new IoT opportunities, we could see new technologies, as well as well joint-ventures, or new business models emerging.