The Current MCU Strategy for Azure

By Sean D. Liming and John R. Malin
Annabooks – www.annabooks.com

September 2022

Those Billions of Cloud-Connected Devices

The success of Azure has paid big dividends for Microsoft. It took a big effort to change the mindset from a Windows, Server, and Office scenario to a cloud strategy that has embraced other platforms, devices, and a plethora of programming languages and APIs. The biggest investment has been in Visual Studio and application programming paradigms to support, not just developing Windows applications, but applications for Android, Linux, and iOS. The initial focus was to connect smartphones, laptops, tablets, and general PCs to the Azure platform. These general-purpose devices are low-hanging fruit, which are the easy targets to get connected to first. What is and has been projected are billions of single-application/dedicated devices connecting to the Cloud. These devices are little sensors and control systems that produce data that can be collected or acted upon in real-time. It is the embedded system becoming embedded/IoT that will bring in the big revenue numbers in the future. These devices run on FPGAs or Micro Controller Units (MCU). Unlike the well know Central Processing Units (CPU) like those from Intel and AMD, MCUs are systems on a chip that combine the CPU with I/O and sometimes internal memory. MCUs come in different architectures: 64-bit, 32-bit, 16-bit, and 8-bit. Market research as far back as 2013 through today's market research shows that 32-bit MCUs dominate the market. 16-bit and 8-bit MCUs follow a distant second. 64-bit MCUs are just coming into the market. Most of the MCUs are ARM-based, but Intel and AMD have MCU solutions for Intel Architecture.

MS-DOS was the last time Microsoft supported a 16-bit CPU. For the past 25 years, Microsoft's embedded/IoT offerings have supported 64-bit and 32-bit CPU and MCU devices. The Windows NT Embedded line and Windows CE were a big part of the strategy at the turn of the century, but as technology evolves, the strategy shifted:

- Windows CE become obsolete as the drive to support only one operating system for all devices came to be with Windows on ARM.
- There was the short-lived .NET Micro Framework trying to play in the low-end device space, but it was rushed out of R&D before it was polished as a commercial quality product. There was no path to support Azure. Also, Raspberry Pi and Arduinos hit the market at the same time, which were cheaper and had better development tools.
- The latest big push was with two Windows 10 offerings: Windows 10 IoT Enterprise and Windows 10 IoT Core. Windows 10 IoT Core was intended to be a replacement for Windows CE, but the limitation of only running UWP applications killed the product before it could gain any traction. That leaves Windows 10 IoT Enterprise as the only solution available.

The embedded Windows line, Windows NT Embedded to Windows 10 IoT Enterprise, has done very well over the past 25 years, and it will continue to do so into the future. The problem is that Windows only reaches a small percentage of the embedded/IoT market. The latest edition of Windows 10 IoT Enterprise LTSC only supports Intel Architecture 64-bit CPU/MCUs and ARM 64-bit MCUs.

Every cloud provider is well aware of the opportunity of the potential billions of connected devices, and they are not lying still. Google has made investments in home automation. Amazon has taken stewardship of the popular FreeRTOS kernel, which runs on many MCUs, and linked it to Amazon Web Services (AWS). The lack of 32-bit MCU support from Microsoft is a big product offering gap. In addition, embedded/IoT device manufacturers have not had the warmest experience with Microsoft over the past 10 years. Many device manufacturers, who spent years investing in Windows CE development, were left a drift with no path forward after Windows CE was canceled.

Many of these companies moved on to Linux. The new solutions from Microsoft that were offered were dropped as fast as they arrived. There are many cloud providers and data centers to connect to. With billions of devices to be connected to the Cloud, what is going to compel device manufacturers to use Azure as a cloud service?

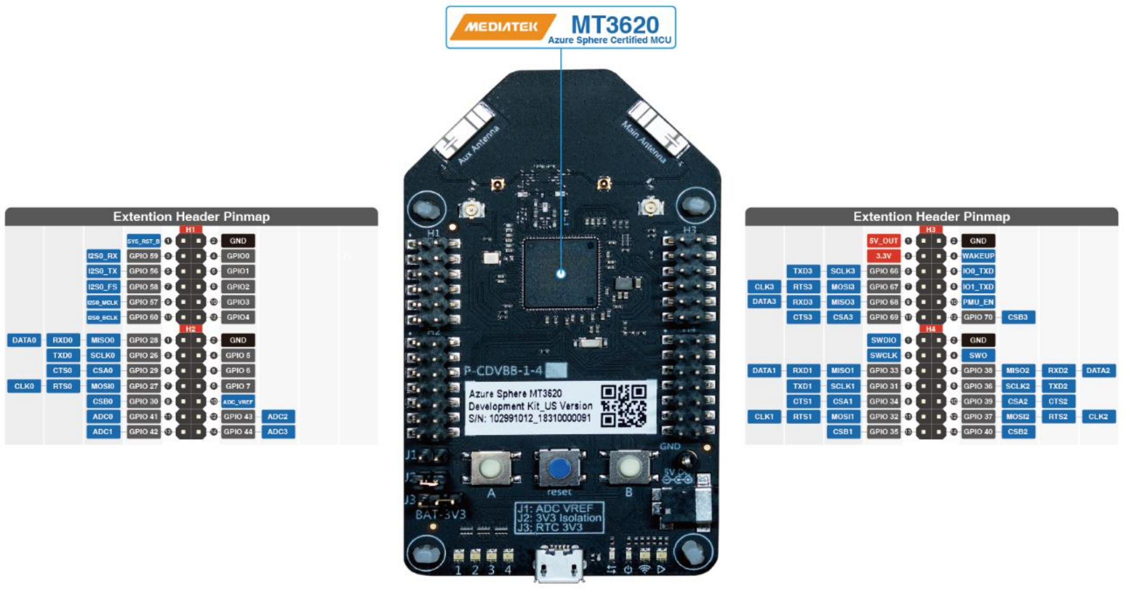
The initial answer has been to focus on the Cloud side:

- The Azure team developed Azure IoT Hub, which made it easy to plug IP-connected devices into the Cloud.
- The Azure team has provided a number of Azure design templates to help developers get started.
- Azure Edge addresses the latency and connectivity issues with on-premises micro cloud application support.
- Last but not least, Azure IoT Central removes much of the complexity of gathering and displaying data from different connected devices.

Solutions in the Cloud are great, but what about an on-device offering? Recognizing the need to fill in the gaps, Microsoft has slowly developed support for 32-bit MCUs so developers can build a device that can go from silicon to the Cloud. The following are the three current product offerings for 32-bit MCUs.

Azure Sphere

Microsoft researchers looked at the issues for connected devices and released a paper titled: *The Seven Properties of Highly Connected Devices.* The research led to creating a custom silicon solution with a security subsystem built in that implements certificate security from hardware to application. The final result is a technology called Azure Sphere. The MediaTek MT3620 is the first silicon to implement the solution to be marketed as Azure Sphere. Azure Sphere is a complete system-on-chip solution that includes a multi-Core ARM solution (ARM Cortex-A7 and ARM Cortex-M4) with all the popular MCU I/O such as UART, I2C, SPI, PWM, ADC, and GPIO. That is not all, the chip also includes RAM and flash storage memory that runs a tiny Linux kernel to handle all the Azure connections. Instead of device manufacturers having to maintain an operating system, Azure Sphere reduces the development effort to simply writing a C-language application to access the I/O and send data to Azure. Maintenance for the Linux kernel is handled by Microsoft and the OEM can control when the Linux kernel gets updated. We wrote about Azure Sphere 3 years ago, *Azure Sphere SDK: First Look,* which demonstrated a few applications and features.


MEDIATEK MT3620
Azure Sphere Certified MCU
Extention Header Pinmap
H1
SYS_RST_B 1 2 GND
I2S0_RX GPIO 59 3 4 GPIO0
I2S0_TX GPIO 56 5 6 GPIO1
I2S0_FS GPIO 58 7 8 GPIO2
I2S0_MCLK GPIO 57 9 10 GPIO3
I2S0_BCLK GPIO 60 11 12 GPIO4
H2
DATA0 RXD0 MISO0 GPIO 28 1 2 GND
TXD0 SCLK0 GPIO 26 3 4 GPIO 5
CTS0 CSA0 GPIO 29 5 6 GPIO 6
CLK0 RTS0 MOSI0 GPIO 27 7 8 GPIO 7
CSB0 GPIO 30 9 10 ADC_VREF
ADC0 GPIO 41 11 12 GPIO 43 ADC2
ADC1 GPIO 42 13 14 GPIO 44 ADC3
Extention Header Pinmap
H3
5V_OUT 1 2 GND
3.3V 3 4 WAKEUP
TXD3 SCLK3 GPIO 66 5 6 IO0_TXD
CLK3 RTS3 MOSI3 GPIO 67 7 8 IO1_TXD
DATA3 RXD3 MISO3 GPIO 68 9 10 PMU_EN
CTS3 CSA3 GPIO 69 11 12 GPIO 70 CSB3
H4
SWDIO 1 2 GND
SWCLK 3 4 SWO
DATA1 RXD1 MISO1 GPIO 33 5 6 GPIO 38 MISO2 RXD2 DATA2
TXD1 SCLK1 GPIO 31 7 8 GPIO 36 SCLK2 TXD2
CTS1 CSA1 GPIO 34 9 10 GPIO 39 CSA2 CTS2
CLK1 RTS1 MOSI1 GPIO 32 11 12 GPIO 37 MOSI2 RTS2 CLK2
CSB1 GPIO 35 13 14 GPIO 40 CSB2
Aux Antenna
Main Antenna
H1
H3
Azure Sphere MT3620
Development Kit_US Version
S/N: 102991012_18310000091
A
reset
B
BAT-3V3
J1: ADC VREF
J2: 3V3 Isolation
J3: RTC 3V3
1 2 3 4

Azure Sphere and the MT3620 were launched back in 2018. There have been presentations that show other silicon vendors producing solutions, but the pandemic and chip shortage appears to have delayed these plans. Regardless, Azure Sphere has found success with some customers. Solely focusing on application development and not having to port, develop, and maintain the operating system is very appealing. The simple connection to Azure IoT Hub and Azure IoT Central is an attractive solution to look into.

More information on Azure Sphere can be found here.

Azure RTOS

FreeRTOS was released in 2003 and has been ported to many MCUs. Amazon took stewardship of FreeRTOs in 2017 and they quickly made the connections to AWS. With the aforementioned Azure Sphere only available from a single chip manufacturer, Microsoft acquired Express Logic, the maker of ThreadX, in 2019. Like FreeRTOS, ThreadX is another popular RTOS that supports a number of MCUs. With the acquisition, the initial development effort was put into connecting ThreadX to Azure, which lead to a new name, Azure RTOS. Azure RTOS is comprised of 8 components. The first 7 have been with ThreadX for a while:

- Azure RTOS ThreadX which is the core RTOS with multithreading capability.
- Azure RTOS FileX to access storage.
- Azure RTOS GUIX for GUI application.
- Azure RTOS TraceX is a Windows-based analysis tool to get a graphical view of running threads.
- Azure RTOS NetX is a high-performance TCP/IP stack.
- Azure RTOS NetX Duo is an industrial-grade TCP/IP stack that supports IPv4 and IPv6.
- Azure RTOS USBX is a host USB stack.

The 8th component, or add-on, that gives ThreadX the new name is the Azure IoT SDK, which provides the middleware layer to connect devices to Azure. Azure RTOS doesn't require a connection to Azure to be licensed for a device, but the SDK makes writing applications to connect to Azure easier. The aforementioned Azure Sphere is a multicore solution. The Linux kernel only runs on one of these cores, the other cores can support Azure RTOS. There is a programming example demonstrating the two working together on GitHub.

Many MCU vendors have their own IDE tools that support creating Azure RTOS applications, and a few have made the adjustments to support Azure RTOS by name. Different IDEs from different vendors is a fragmented development story. Porting Azure RTOS to new MCUs is the real challenge just as it was for Windows CE. It will be interesting to see how and if Azure RTOS gets tightly integrated into Visual Studio or VS Code over time.

More information on Azure RTOS can be found here.

Azure IoT SDKs

The final piece to the current strategy is a number of Azure IoT SDKs that support different devices and programming languages. These SDKs fall into two camps: Device SDKs and Embedded Device SDKs. The Device SDK supports different programming languages: .NET, Python, node.js, Java, and C. The Embedded Device SDKs are for systems with limited computing resources, and there are currently three of them. There is a bare metal SDK with its own name: "Azure SDK for Embedded C". We already touched on the SDK for Azure RTOS. The most interesting is the Azure IoT SDK for FreeRTOS, which supports a variety of MCUs on the market. These SDKs combined with Azure Sphere and Azure RTOS cast a wide net for the market.

More information on the Azure IoT SDKs can be found here.

The Future is Always in Motion

A small ROMable Windows operating system is no more, but finding how to win in the billions of cloud-connected devices market space is critical to Azure's growth. In a few short years, Microsoft has defined its new playing field by filling the 32-bit MCU gap and how these devices connect to Azure. Casting a wide net with different offerings is an interesting approach. Device manufacturers will choose the best path forward. A strategy is only a theory and a journey. The success of this MCU strategy will be in the product planning and execution. The real value proposition that Microsoft brings to the table is what they do best: development tools. How all these product offerings are integrated into the development tools and make the whole product development and life cycle better, is the question that Microsoft needs to address. As with any journey, we will have to see how the strategy evolves.

Reference

Microcontroller Market Size, Share, Trends, Opportunities & Forecast (verifiedmarketresearch.com)

Microcontroller Market Size | Industry Report, 2022-2030 (grandviewresearch.com)

MCU market turns to 32-bits and ARM - EETimes

FreeRTOS - Wikipedia

ThreadX - Wikipedia

The IoT Show - Azure Sphere and Azure RTOS