

# New Electronics

## Changing the embedded development model with Microsoft .NET Micro Framework

24/08/2009  [Email to a friend](#)

While there is a broad range of embedded applications in need of complete and highly granular control of all operational aspects of a device, there is also a world of embedded applications out there that can easily do without. This white paper from Digi International takes a closer look.

### Author

Mike Rohmoser, senior product manager for Digi International

### Download Articles

[Digi.pdf](#)

### Supporting Information

<http://www.digiembedded.com>

## Changing the embedded development model with Microsoft .NET Micro Framework

The development model for embedded devices is traditionally viewed as extremely complex with the need for highly specialized design expertise and deep knowledge of a target's intricate software and hardware design aspects. With all that, of course, come prolonged design cycles and competing project timelines resulting in high development cost and long time-to-market.

While it certainly still holds true that there is a broad range of embedded applications in need of complete and highly granular control of all operational aspects of a device, there is also a world of embedded applications out there that can easily do without. And given that almost everyone has to deal with engineering resource constraints on a daily basis, wouldn't it be ideal to engage the non-embedded engineering software team as well as re-use existing code from the server/desktop side? After all, today's constant push towards pervasive networked device collaboration also implies tight integration with the backend network infrastructure, which is typically built on software utilizing non-embedded desktop and server class systems. Signs of an increasing trend to pull enterprise server and desktop application software components into the embedded space are already visible across other major platforms, however, often at the expense of severely elevated hardware requirements due to the lack of

optimization for embedded targets. This is exactly where the .NET Micro Framework provides a fresh and innovative perspective in the form of a new embedded development option.

### Microsoft .NET Micro Framework Architecture

Introduced in early 2007, Microsoft .NET Micro Framework is a lightweight implementation of the .NET Framework. Compared to the smallest managed configuration of Windows Embedded CE, it's fraction of the size. We are talking about a few hundred kilobytes instead of multiple megabytes. Even though it is very compact and by design does not offer the complete feature set of Windows Embedded CE, it still scales nicely from simple, single-function devices up to more sophisticated and powerful multi-function devices with state-of-the-art user interfaces. The particular focus of .NET Micro Framework is on the growing number of network-enabled embedded devices with 32-bit processors, which are rapidly displacing the previously dominant 8- and 16-bit based devices. The .NET Micro Framework platform was specifically designed to meet the requirements and extended capabilities of this new generation of 32-bit devices, and built from the ground up instead of being a derivative work of an existing Microsoft embedded platform.

From an internal architecture point of view, the .NET Micro Framework consists of the following layers:

- Application Layer  
C# application utilizing built-in and user libraries/services
- Class Library Layer  
Core subset of .NET libraries and application services
- Runtime Component Layer

#### CLR – Common Language Runtime

Runtime environment providing execution engine, thread management, garbage collection, exception handling, and other services

#### PAL – Platform Abstraction Layer

Hardware-independent abstraction layer providing services to the CLR, including memory management, debugging, and asynchronous procedure calls.

#### HAL – Hardware Abstraction Layer

Interface between PAL and hardware or operating system providing access to hardware functions

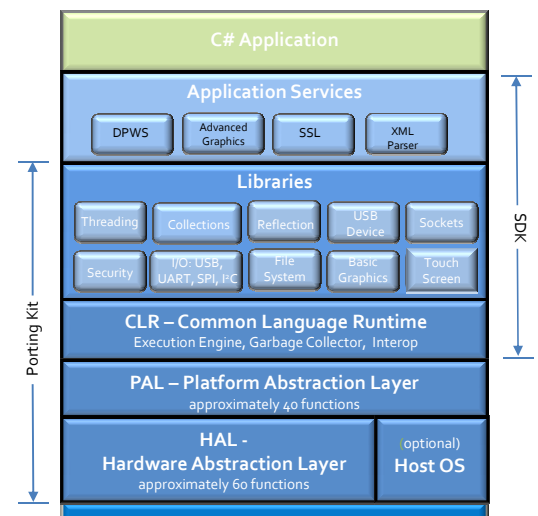


Figure 1: .NET Micro Framework 3.0 Architecture

- Hardware Layer  
Processor platform and integrated peripherals

The CLR in the runtime layer interprets language independent intermediate code (Common Intermediate Language – CIL) generated by the Visual Studio compiler, and handles aspects such as type safety and garbage collection, including safeguards resolving very common application software problems caused by memory leaks and unsafe pointers. While this sophisticated level of code management does result in non-deterministic application behavior, it also provides an exceptionally robust and safe environment for a large number of embedded applications that do not require real-time performance. The CLR concept is also completely independent from the application programming language and the underlying hardware, and makes .NET Micro Framework a future-proof platform with unique programming language flexibility and virtually unlimited porting capabilities.

### **Changing the software development model**

The .NET Micro Framework offers a C# managed code environment and seamless integration into the Microsoft Visual Studio environment, including full on-device debug support. This means that developers that are already experienced with .NET Framework and the Visual Studio tools can take immediate advantage of their existing skills and enjoy designing embedded applications without going through a significant learning curve.

The C# language is powerful and easy to learn, and also allows new and previously unfamiliar developers to be productive very quickly. The actual C# application development process is completely shielded from the low-level design details of the hardware platform by simply utilizing the provided .NET class libraries, application services such as Devices Profile for Web Services (DPWS), and hardware-specific interface support. An XML-defined and extensible hardware emulator is also part of the development tools, and makes testing and developing software possible without involving any actual target hardware. If needed, managed drivers can be written in C# for components attached to device interfaces, e.g. SPI or I<sup>2</sup>C, given that they are already supported by the .NET Micro Framework libraries.

All this and the fact that existing .NET code used in enterprise server and desktop applications is easily shared with a .NET Micro Framework application on an embedded device dramatically reduces traditional design risks and greatly accelerates the software development process.

Community provided software for .NET Micro Framework is widely available through .NET Framework sample code, but continues to expand into platform-specific development activities such as the C# implementation for DNS, ZigBee, POP, SMTP, and HTTP web server support provided by Microsoft MVP Michael Schwarz on CodePlex (<http://www.codeplex.com/mschwarztoolkit>). Other interesting hands-on resources available are Pavel Banský's blog (<http://bansky.net/blog/>), and of course the Microsoft newsgroup (microsoft.public.dotnet.framework.microframework).

### **Changing the hardware development model, too**

While the software aspect is certainly a key factor, the best software cannot live up to its expectations without the supporting hardware. Processor support for the .NET Micro Framework is available for a variety of 32-bit platforms, including ARM7 and ARM9 models as well as ADI Blackfin. The Microsoft Porting Kit allows customers to adapt .NET Micro Framework to their target hardware. However, porting .NET Micro Framework to custom hardware does require specific skill sets that may or may not be available in your organization. A much more efficient approach that fully embraces the new software development model of .NET Micro Framework is the use of embedded processor modules rather than engaging in discrete hardware design efforts, whenever possible.

Embedded processor modules provide a complete and functional system with processor, memory, and supporting circuitry, on a compact single-component module design. Combining the benefits of the Microsoft .NET Micro Framework with off-the-shelf, low-cost module platforms like Digi International's line of network-enabled embedded processor modules creates a rapid product development solution that has already proven to get customers from development to functional product within 3 to 6 months.

Modules change the hardware development model in the same dramatic way .NET Micro Framework changes the traditional software development model, with benefits such as:

- Extremely short product design cycle and time-to-market
- Software design can start immediately on actual hardware platform
- Pre-certified module designs further reduce overall hardware design risk
- Carrier board design simplified, inexpensive, and less likely to go through re-spins
- Single vendor supporting both the hardware and software platform
- Optional migration path to component integration to match high-volume production cost expectations and/or mechanical design considerations

Developers should look for module manufacturers with a strong .NET Micro Framework offering, including complete and easy-to-use development kits providing the Microsoft .NET Micro Framework SDK and full platform support, allowing immediate C# application development. The ideal hardware solution is built on a manufacturer's own processor platform, which makes a potential future migration to a discrete design much easier from a final product integration and cost point of view.

### **Products and applications**

One of the most compact modules on the market is the Digi Connect ME, which is one of the modules that is part of Digi International's .NET Micro Framework product offering. This embedded module integrates Digi's own ARM7TDMI based NS7520 processor running at 55 MHz, with 2 MB flash, 8 MB RAM, UART interface, GPIO, and a 10/100 Mbit Ethernet interface in a compact RJ-45 form factor. Complete development kits are available, and currently support the .NET Micro Framework 2.5 release.



Figure 2: Digi Connect ME module

A good example of the Digi Connect ME in use with .NET Micro Framework is a ZigBee-to-Ethernet gateway for a tank level monitoring application. The .NET Micro Framework application on the gateway collects and processes information from the individual ZigBee enabled sensors, and feeds it back to a central location through an on-site DSL connection. Reusing existing .NET Framework code from the enterprise server side seamlessly integrates the solution into the corporate backend with only minor development effort. The application also provides simple web server functionality for on-site configuration and local status monitoring. ZigBee connectivity in the gateway is incorporated by utilizing a Digi XBee-PRO RF module connected to the Digi Connect ME's UART interface. It is easy to see how solutions like this can be applied to other applications by taking direct advantage of the accelerated customization capabilities offered by .NET Micro Framework and a modular hardware platform design.

The Microsoft .NET Micro Framework delivers on the promise of dramatically improved productivity and highly accelerated software development by extending the reach of the .NET Framework technology and the desktop computing model down to intelligent connected devices. Significantly reducing designs risks and time-to-market, it creates a new software development model that is complemented by the hardware development model of embedded modules. It is a powerful combination introducing a dramatic paradigm shift that simplifies embedded development and makes it immediately accessible to an entirely new developer audience and class of devices.

*Mike Rohmoser is senior product manager for Digi International. For more information visit [www.digiembedded.com](http://www.digiembedded.com).*