

Zürich, im Januar 2008

Microsofts .NET für Embedded Systems

Das *.NET Micro Framework* ist das kleinste Mitglied der .NET Plattform-Familie von Microsoft, ausgelegt auf kleine Embedded Systems. Embedded Systems sind „versteckte Computer“, die oft nicht einmal ein Display besitzen. Sie stecken in Kassen, Getränkeautomaten, Verkehrssampeln, Stromzählern, Druckern usw. Das .NET Micro Framework hat zum Ziel, die Programmierung solcher Systeme radikal zu vereinfachen, durch die Verwendung gängiger .NET Technologien wie C# und Visual Studio.

Vielleicht ist diese neue Plattform auch für Sie von Interesse. In diesem Infoblatt geben wir Ihnen einen kurzen Überblick über seine wichtigsten technischen Eigenschaften und Einsatzmöglichkeiten.

Falls Sie eine Projektidee haben, die vom .NET Micro Framework profitieren könnte, setzen Sie sich mit uns in Verbindung!

Dr. Cuno Pfister

Neue Chancen – und eine neue Herausforderung

Embedded Systems gewinnen rapide an Bedeutung. Während vor einigen Jahren in einem Automobil noch wenige Mikroprozessoren eingebaut waren, sind es in einem heutigen Oberklasse-Fahrzeug 50 bis 80 Stück. Möglich wurde dies dank stetiger Fortschritte bei der Produktion von integrierten Schaltungen, was zu immer billigeren und gleichzeitig leistungsfähigeren Prozessoren, Flash-Speichern und Displays, sowie zur Vernetzung von Prozessoren geführt hat.

Zugang zum Internet ist ein Paradebeispiel für diesen Trend. Es wird immer häufiger verlangt, dass ein Embedded System über das Internet kommunizieren kann. So kann z.B. ein Chemikaliertank selbständig eine Nachbestellung auslösen, wenn sein Inhalt zur Neige geht. Maschinen können Frühwarnungen auslösen, um teure Stillstandszeiten zu vermeiden. Das Internet ermöglicht sogar ferngesteuerte Software-Updates, um z.B. verbesserte Verfahren zur Früherkennung von Störungen einfach, schnell und billig zu installieren.

Software-Updates von Geräten, u. U. hunderte von Kilometern vom nächsten Service-Center entfernt, sind jedoch nur dann verantwortbar, wenn die Geräte funktionsfähig bleiben, selbst falls während des Updates Fehler auftreten. Zudem dürfen nur autorisierte Stellen ein Software-Update durchführen können. Das bedeutet, dass solche Geräte Software für Internet-Protokolle, Sicherheitsmechanismen und vieles mehr benötigen. Dies erweitert die Möglichkeiten von Embedded Geräten enorm und schafft, derzeit noch kaum abschätzbare, Chancen für neuartige Produkte und Services.

Im gleichen Masse explodiert jedoch auch die Komplexität der Software-Entwicklung. Sie wird zunehmend zum Engpass bei der Entwicklung neuer Geräte. Projektrisiken, Verzögerungen bei der Auslieferung und Zuverlässigkeitsprobleme bei ausgelieferten Geräten können enorme Kosten verursachen.

Microsofts .NET und das .NET Micro Framework

Solche Herausforderungen gibt es auch für PC und Server. Dort haben sich in den letzten Jahren objekt-orientierte Programmiersprachen, integrierte Entwicklungsumgebungen und moderne Laufzeitsysteme bewährt. Zwei derartige Plattformen haben sich am Markt durchgesetzt: Java und .NET. Sie unterstützen „Managed Code“, d.h. Programm-Code, bei dem wichtige Fehlerquellen – insbesondere bei der Speicherverwaltung – grundsätzlich ausgeschlossen sind.

Diese Plattformen sind typischerweise sehr gross und kommen für kleine Embedded Systems nicht in Frage. Dies wird sich jedoch in Zukunft dank speziell für den Embedded Markt entwickelter, neuer Implementierungen ändern.

Microsofts .NET Micro Framework wurde ursprünglich für den Eigenbedarf von Microsoft geschaffen, nämlich für Projekte wie die SPOT Uhren (SPOT steht für Smart Personal Object Technology), für Settop-Boxen und für die neuen Vista SideShow-Displays in Laptops. Neuerdings wird die Plattform auch an andere Firmen lizenziert.

Anders als das volle .NET Framework, das Windows voraussetzt, oder das .NET Compact Framework, welches auf Windows CE aufsetzt, benötigt das .NET Micro Framework *kein* separates Betriebssystem – es ist eine „bootable .NET runtime“. Dank dieses minimalistischen Ansatzes genügen typischerweise je 500 KB Flash- und Arbeitsspeicher. Dies ist etwa 20 mal weniger als für Systeme mit dem .NET Compact Framework und etwa 1000 mal weniger als für Systeme mit dem vollen .NET Framework für PC und Server.

Zur Programmierung steht die Programmiersprache C# und die Entwicklungsumgebung Visual Studio zur Verfügung. Damit kann Code komfortabel auf echte Geräte heruntergeladen, ausgeführt und „debugged“ werden. In der Praxis wird man dennoch in der Regel mit dem Emulator arbeiten. Der Emulator ist erweiterbar, um z.B. spezielle Peripheriegeräte der eigenen Hardware simulieren zu können. Dies hat den Vorteil, dass man nicht auf die Fertigstellung der Hardware warten muss, und somit Hardware und Software parallel entwickeln kann.

Das .NET Micro Framework unterstützt eine Teilmenge der Klassenbibliothek des vollen .NET Frameworks, ergänzt um Funktionalität, die für Embedded Systems relevant ist: digitale Ein- und Ausgänge, serielle Schnittstellen usw. Selbst Gerätetreiber können in Managed Code implementiert werden.

Damit werden die Vorteile von .NET auch für kleine Embedded Systems zugänglich: Höhere Produktivität, Robustheit, Sicherheit und Wartbarkeit. Darüber hinaus vereinfacht das .NET Micro Framework die Software-Entwicklung, indem es die Verwendung des gleichen Know-hows, der gleichen Tools und sogar der gleichen Software-Komponenten für Embedded Systems, Smartphones, PC und Server erlaubt.

Welche Hardware kann man einsetzen?

Für Embedded Systems wird oftmals eigene Hardware entwickelt, optimiert für eine bestimmte Aufgabe. Eine Plattform wie das .NET Micro Framework muss zuerst auf diese Hardware portiert werden. Dies ist eine nichttriviale Aufgabe, für die es spezielle Werkzeuge, spezielles Know-how und einen sogenannten Porting Kit von Microsoft braucht.



Zu Evaluationszwecken gibt es fixfertige Boards, auf die das .NET Micro Framework bereits portiert worden ist, sodass man sofort mit der Entwicklung beginnen kann. Das i.MXS Development Kit von Freescale (siehe Bild) ist ein Beispiel eines solchen Boards.

Diese Art Boards wird man selten für echte Anwendungen einsetzen, da meistens nicht alles schon so ist, wie man es braucht. Ein komplett eigenes Prozessorboard zu entwickeln ist jedoch aufwendig. Als Mittelweg kann man Prozessormodule einsetzen, bei denen die kniffligen technischen Probleme, wie z. B. das Mehrlagen-Lay-

out des Boards, bereits gelöst sind. Ein solches Modul enthält typischerweise den Mikroprozessor sowie Flash- und Arbeitsspeicher. Man muss dann lediglich noch eine einfache Trägerplatine mit den nötigen Peripheriebausteinen und Steckern entwickeln und das Modul aufstecken. Das

„Meridian CPU“-Modul von Embedded Fusion ist ein Beispiel, bei dem das .NET Micro Framework bereits portiert wurde.

Weitere Produkte gibt es von anderen Firmen oder befinden sich in Entwicklung. Zurzeit sind dies alles Rechner, die auf der stromsparenden ARM-Architektur basieren, weitere Prozessorarchitekturen sollen folgen.

Anwendungsbereiche und Grenzen

Das .NET Micro Framework eignet sich besonders für Embedded Systems, die mit anderen Systemen kommunizieren, oder über grafische Benutzerschnittstellen verfügen. Das können z. B. Fernsteuerungen, Protokollkonverter, „Human Machine Interfaces“, mobile Messgeräte, Laborgeräte, Billetautomaten oder Geräte der Gebäudetechnik sein.

Im Vergleich zur PC-Programmierung muss man hier bewusster mit Ressourcen umgehen: Prozessorleistung, Speicher, Strom, Netzwerkbandbreite. Solides Software



Engineering ist also gefragt. Doch das „Herumpfriemeln“ mit einzelnen Bits von obskuren Hardware-Registern gehört der Vergangenheit an, stattdessen werden Hardware-unabhängige objekt-orientierte Abstraktionen von Peripheriegeräten benutzt. Allein dadurch wird das Entwickeln von Embedded Systems ungemein vereinfacht und beschleunigt. Nebenbei wird die resultierende Software zuverlässiger und auch sehr portabel.

Manche Embedded Systems müssen maximale Reaktionszeiten auf bestimmte Ereignisse garantieren. Auf solche harten „real-time“ Anforderungen ist das .NET Micro Framework nicht ausgelegt.

Kostenmässig ist das .NET Micro Framework für alle Systeme interessant, die in kleinen bis mittleren Stückzahlen hergestellt werden – sagen wir, weniger als 100'000 Stück.

Wie sind wir auf das .NET Micro Framework gekommen?

Wir haben die fundamentale Bedeutung von Managed Code bereits vor 20 Jahren an der ETH Zürich erlebt, wo die Professoren Niklaus Wirth und Jürg Gutknecht die Programmiersprache und das Betriebssystem Oberon entwickelt haben. Dort haben wir gelernt, wie man solche Plattformen nutzt und auch selbst konstruiert, auch für sehr kleine Systeme. Durch die spätere Entwicklung des Echtzeitbetriebssystems Jbed haben wir zusätzliche Erfahrung mit einer echtzeitfähigen Managed Code Plattform gewonnen.

In den letzten Jahren haben wir in Kundenprojekten intensiv mit dem .NET Compact Framework gearbeitet. Diese .NET Implementierung für mobile Geräte hat sich bewährt. Sie benötigt jedoch ein darunterliegendes Betriebssystem und ist damit für viele Embedded Systems zu gross. Zudem fehlt ihr Unterstützung für das Schreiben von Treibern in Managed Code.

Unser Interesse an einer noch kleineren Plattform, geeignet auch für „tiefer eingebettete“ Systeme, ist geblieben. Denn wir sind davon überzeugt, dass dort die fortschreitende Hardware-Entwicklung und die Integration mit dem Internet einen enormen Innovationsschub auslösen wird. Eine Managed Code Plattform für kleine Embedded Systems wird deshalb je länger, desto attraktiver. Hier möchten wir unsere Erfahrung an vorderster Front einbringen.

Dies hat uns bewogen, eine direkte Verbindung zum .NET Micro Framework Entwicklungsteam in den USA aufzubauen. Heute sind wir die ersten und bisher einzigen Lösungspartner von Microsoft in der Schweiz für das .NET Micro Framework. Zurzeit portieren wir das Framework auf die Hardware eines Kunden, gleichzeitig entwickeln wir eine Anwendung dafür.