

Introductie

Een embedded platform inclusief .NET framework, passend in een footprint van ongeveer 300 kB. Sinds maart 2007 is het .NET Micro Framework een feit. Het .NET Micro Framework is nog niet echt bekend in Nederland. Hoog tijd dus voor een introductie.

We hadden natuurlijk al het .NET (Full) Framework en het kleinere, compacte broertje, het .NET Compact Framework. Toch was er volgens Microsoft Research behoefte aan een nog compacter framework, met een nog kleinere footprint, energiezuiniger en speciaal voor embedded devices. Microsoft Research verwacht namelijk dat veel elektronische (huishoudelijke) apparaten in de toekomst uitgerust worden met een 32-bits processor, batterijen, netwerkaansluiting en gebruik maken van nieuwe protocollen zoals Z-Wave en ZigBee. Om ook op deze devices de beschikking te hebben tot een .NET-framework, is het .NET Micro Framework ontwikkeld.

Net zoals het .NET Compact Framework is het .NET Micro Framework een subset van het .NET (Full) Framework. Het .NET Micro Framework is de kleinste subset tot nu toe en heeft een footprint van ongeveer 300 kB. Met dit framework kunnen applicaties in managed code ontwikkeld worden en als ontwikkelaar heb je de volledige Visual Studio 2005 omgeving tot je beschikking. Momenteel is C# de enige managed taal die ondersteund wordt door het framework.

Toepassingen

Op dit moment wordt het .NET Micro Framework al voor een aantal toepassingen gebruikt. De bekendste toepassing is Vista SideShow. Een SideShow device is een extra scherm voor je Vista computer en kan gebruikt worden om informatie te tonen (zie *Figuur 1*).



Figuur 1: De Asus W5Fe

De Asus W5Fe laptop heeft bijvoorbeeld een ingebouwd SideShow device. Hierop kunnen bijvoorbeeld emails en foto's bekeken worden en kan muziek afgespeeld worden, en dit alles zonder de laptop op te starten.

Wat heeft dit framework te bieden?

Hieronder een overzicht van de belangrijkste eigenschappen van het .NET Micro Framework:

- Geïntegreerd in Visual Studio 2005
- Applicaties ontwikkelen in managed code (C#)
- Kleine footprint (250 – 500 kB)
- Kan zonder besturingssysteem draaien
- Draait op 32 bits processors
- Vereist geen MMU ondersteuning
- Lage hardwareprijs
- Energiezuinig
- Ondersteunt industriële communicatie standaarden zoals: SPI, I²C, GPIO en U(S)ART
- Hardware interfaces zijn beschikbaar via object georiënteerde libraries
- Beschikt over een uitbreidbare emulator

De Footprint

Microsoft heeft momenteel drie verschillende embedded platformen:

- Windows CE
- Windows XPe
- .NET Micro Framework

Het .NET Micro Framework is een platform an sich, aangezien de runtime omgeving besturingssysteem-functionaliteiten bevat. In het hoofdstuk ‘Common Language Runtime’ wordt hier verder op ingegaan.

In *Tabel 1* is aangegeven hoe groot de footprint is van de verschillende embedded platformen met en zonder managed code support. Merk op dat de footprint van het .NET Micro Framework, waarin managed code ondersteund wordt, ongeveer even groot is als de footprint van Windows CE zonder managed code support. De combinatie van de kleine footprint en de managed code omgeving van het .NET Micro Framework is dan ook erg interessant!

Embedded Platform	Footprint	Unmanaged code	Managed code
.NET Micro Framework	250 – 500 kB	Nee	Ja
Windows CE	300 kB +	Ja	Nee
Windows CE + .NET CF	12 MB	Ja	Ja
Windows XPe + .NET	40 MB +	Ja	Ja

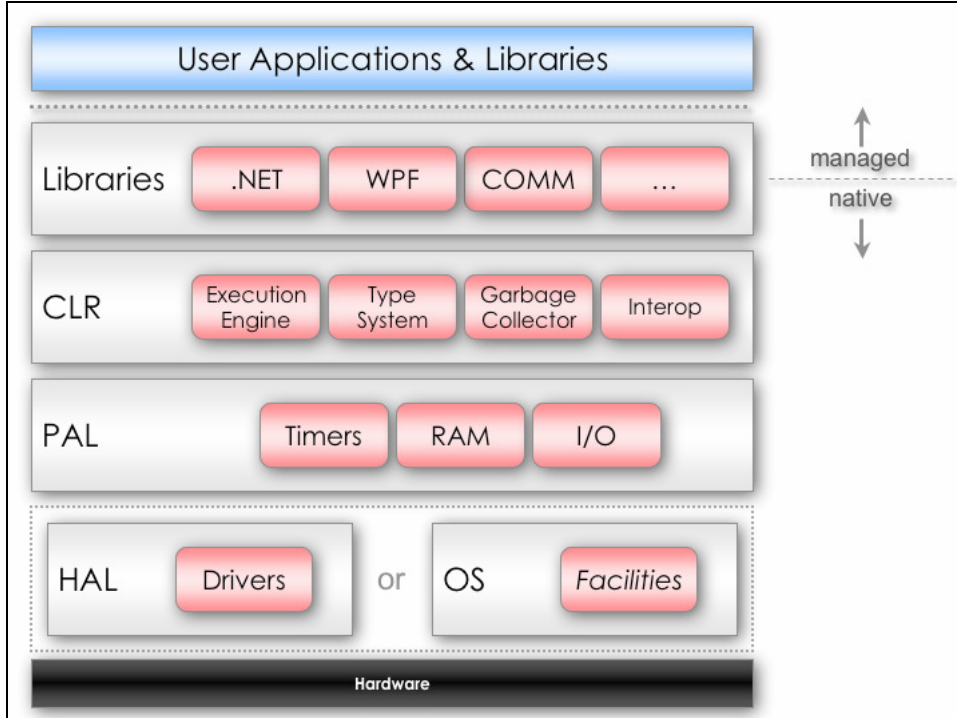
Tabel 1: De footprint van de verschillende Microsoft embedded platformen.

In tegenstelling tot wat op veel websites is te lezen, ondersteunt het .NET Micro Framework nog geen Interop functionaliteit. Interop, oftewel inter-operation, maakt het mogelijk om vanuit managed code een platform invoke (P-invoke) te doen naar unmanaged code.

Het kan soms gewenst zijn, vanwege performance verbeteringen, een stuk code in een lagere programmeertaal (een native taal als C of C++) te coderen. Met het .NET Micro Framework kan dit nog niet, aangezien daar interop functionaliteit voor nodig is. Hopelijk komt deze functionaliteit snel beschikbaar. Het .NET Micro Framework is immers in het leven geroepen voor embedded devices. Juist op deze devices wil je de mogelijkheid hebben om uit te kunnen wijken naar native code.

Architectuur

In het figuur hieronder, *Figuur 2*, is de software architectuur van het .NET Micro Framework weergegeven.



Figuur 2: De software architectuur van het .NET Micro Framework

Het .NET Micro Framework kan opgedeeld worden in vier componenten:

- Library classes
- Common Language Runtime (CLR)
- Platform Abstraction Layer (PAL)
- Hardware Abstraction Layer (HAL)

Library classes

De class library van het .NET Micro Framework bestaat uit een subset van de .NET Base Class Libraries (BCL) en de SPOT Namespaces. SPOT staat voor Smart Person Object Technology en richt zich, volgens Microsoft, op het verbeteren van het gebruik van alledaagse objecten door het toevoegen van software. Er zijn bijvoorbeeld SPOT horloges die automatisch hun tijd aanpassen aan de hand van de tijdzone waar de drager van het horloge zich op dat moment bevindt, en die relevante nieuwsberichten voor die regio op het horloge tonen.

Om de subset van de BCL portable te houden met het (Full) Framework is de BCL namespace zoveel mogelijk intact gehouden. Indien er belangrijke functionele wijzigingen zijn gedaan, is de betreffende functionaliteit verhuisd naar een andere namespace. Zo was de footprint van Windows Forms (*System.Windows.Forms*) te groot en is er een afgeslankte versie van het Windows Presentation Foundation (WPF) (*Microsoft.SPOT.Presentation*) ontwikkeld.

Windows Presentation Foundation

Een goede keuze van Microsoft is het toevoegen van de Windows Presentation Foundation (WPF) classes aan het .NET Micro Framework.

Momenteel maken alleen het .NET (Full) Framework en het .NET Micro Framework gebruik van deze classes. Natuurlijk bevat het .NET Micro Framework wel een afgeslankte versie van het WPF. XAML, de markup-taal voor de gebruikersinterfaces in het WPF is bijvoorbeeld achterwege gelaten. Het WPF stelt een programmeur in staat om op een makkelijke manier een gebruikersinterface voor zijn device te maken.

Namespaces

Hieronder is een opsomming gegeven van de namespaces waaruit het .NET Micro Framework bestaat:

```
System
System.Collections
System.Diagnostics
System.IO
System.Net
System.Net.Sockets
System.Reflection
System.Resources
System.Runtime.CompilerServices
System.Runtime.Remoting
System.Text
System.Threading

Microsoft.SPOT
Microsoft.SPOT.Cryptography
Microsoft.SPOT.Hardware
Microsoft.SPOT.Input
Microsoft.SPOT.Messaging
Microsoft.SPOT.Presentation
```

Common Language Runtime

De Common Language Runtime (CLR) is de Runtime, oftewel virtuele machine, van .NET. Ook het .NET Micro Framework bevat een Runtime omgeving. Het is immers managed code waarin gecodeerd wordt. Bij het ontwikkelen van deze Runtime zijn veel concessies gedaan om de footprint van het framework zo klein mogelijk te houden.

Bij het .NET (Full) Framework en het .NET Compact Framework wordt met behulp van Visual Studio de geschreven C# code omgezet in een tussentaal: Microsoft Intermediate Language (MSIL). De IL-code wordt vervolgens op het device gezet. Tijdens het runnen van de applicatie wordt door de Execution Engine in CLR de IL-code 'Just-in-time' (JIT) gecompileerd naar native code om vervolgens op het systeem uitgevoerd te kunnen worden.

In tegenstelling tot het .NET (Full) Framework en het .NET Compact Framework bevat het .NET Micro Framework geen JIT compiler. Hier bestaat de Execution Engine uit een Intermediate Language Interpreter die, zoals de naam al aangeeft, de IL-code interpreteert en uitvoert. De code wordt dus niet eerst naar native code omgezet, maar gelijk geïnterpreteerd. Door deze implementatie kan de footprint kleiner gehouden worden, is er alleen maar een IL-versie van de applicaties op het device aanwezig en niet nog eens een native versie. Tevens kan er nog eens geheugen bespaard worden doordat IL-code compacter is dan native code.

Zoals in het hoofdstuk 'De footprint' is aangegeven bevat het .NET Micro Framework geen functionaliteit voor het aanroepen van native code. Merk op dat het implementeren van deze functionaliteit nu ook een stuk ingewikkelder is, aangezien de Execution Engine de IL-code interpreteert in plaats van het eerst compileert naar native code en vervolgens uitvoert. De CLR in het .NET Micro Framework kan dan ook niet omgaan met native code.

De CLR is een bootable runtime, wat wil zeggen dat de runtime-omgeving zelf besturingssysteem-functionaliteit bevat zoals: booting support, interrupt afhandeling, threading, process en heap management. De CLR kan dus direct op de hardware draaien en een besturingssysteem is hierdoor overbodig. Ook dit scheelt in de grootte van de footprint. Indien extra functionaliteit, zoals een uitgebreide scheduler voor real-time gedrag gewenst is, kan er gekozen worden om gebruik te maken van een onderliggend besturingssysteem. Dit kan elk willekeurig 32 bits besturingssysteem zijn. De HAL (Hardware Abstraction Layer) en PAL (Platform Abstraction layer) zullen in dit geval gepoort moeten worden om zo de CLR te voorzien van de functionaliteit van het toegevoegde besturingssysteem.

Hardware- en Platform Abstraction Layer

De hardware of het besturingssysteem waar het framework op draait wordt afgescheiden van de CLR door de Hardware Abstraction Layer (HAL) en de Platform Abstraction Layer (PAL) die bovenop de HAL geplaatst is. De HAL verzorgt onder andere het booten en de I/O afhandeling. De PAL verzorgt bijvoorbeeld de timers en de asynchrone communicatie.

Indien het .NET Micro Framework gepoort moet worden naar een andere CPU, dan dient alleen de HAL aangepast te worden. Indien er een ander platform gebruikt moet gaan worden, zal ook de PAL aangepast moeten worden. Het poorten van het .NET Micro Framework is momenteel alleen nog maar weggelegd voor Microsoft porting partners. Bedrijven kunnen partner worden indien ze een overeenkomst hebben gesloten met Microsoft. Doordat er weinig informatie bekend is over de interne structuur van de HAL en PAL is het voor een ontwikkelaar moeilijk om er achter te komen wat er onder water precies gebeurt. De porting-kit er bij pakken is dus voor velen ook geen optie.

Extensible Emulator

Het .NET Micro Framework bevat een krachtige, uitbreidbare emulator. Waar bij het .NET Compact Framework alleen een paar eigenschappen van de emulator ingesteld kunnen worden, biedt de emulator van het .NET Micro Framework een rijk gevulde toolbox.

De emulator kan geconfigureerd worden met behulp van XML. GPIO pins kunnen aangepast worden net zoals bijvoorbeeld de seriële poort, I²C en SPI. Ook kunnen eigen Emulator Components geschreven worden, zodat de (toekomstige) hardware gesimuleerd kan worden. Hierdoor kan zelfs zonder hardware al begonnen worden met coderen.

Conclusie

Het .NET Micro Framework heeft veel te bieden en past toch in een kleine footprint (250 – 500 kB). Doordat geheugen steeds goedkoper wordt en de 32 bits processoren steeds energiezuiniger, wordt een framework zoals het .NET Micro Framework erg interessant. Tevens vergroot een framework zoals deze de productiviteit aanzienlijk, aangezien het geïntegreerd is in Visual Studio, een Extensible Emulator, object georiënteerde hardware interfaces en libraries zoals Windows Presentation Foundation bevat.

Jacob Bijsterbosch

Referenties

- Embedded Programming with the Microsoft .NET Micro Framework
Donald Thompson, Rob S. Miles
- Introducing the .NET Micro Framework: Product Positioning and Technology White Paper
Donald Thompson, Colin Miller
- The .NET Micro Framework Platform SDK
<http://msdn2.microsoft.com/en-us/library/bb410018.aspx>
- SPOT-light on Microsoft's SPOT Technology
[http://www.windowsfordevices.com/articles/AT6914689493.ht](http://www.windowsfordevices.com/articles/AT6914689493.html)
ml