



Micro
Framework

Обзор .NET Micro Framework

Алексеев Пётр
Доцент кафедры Радиоэлектронных систем управления (И4) БГТУ «Военмех»


MCSD.NET

www.netmf.ru

Pe



План

- ▶ Основы .NET MF
 - ▶ Особенности разработки и отладки
 - ▶ Низкоуровневый ввод/вывод и обработка прерываний
 - ▶ Последовательные порты
 - ▶ Графика
 - ▶ Flash-память
 - ▶ Сеть
 - ▶ Криптография
 - ▶ SideShow
 - ▶ Портирование
 - ▶ Дополнительные возможности
- 

ОСНОВЫ .NET MF

»» Часть 1

Знаете ли Вы что...

- ▶ Ваши навыки разработки .NET приложений, работающих в web, на мощных серверах и на рабочих станциях можно применить и для миниатюрных устройств, работающих на батарейках
- ▶ Приложения .NET могут работать без операционной системы, на «голом» железе
- ▶ Мы стоим на пороге глобального распространения множества устройств, умеющих самостоятельно обмениваться информацией, в том числе и через Интернет, при этом каждое такое устройство будет не универсальным, а предназначенным для выполнения единственной функции

Знакомьтесь:

.NET Micro Framework

- ▶ .NET Micro Framework – миниатюрная реализация .NET Framework для встраиваемого применения
 - Размер исполняющей среды 250 кб
 - Работает на микроконтроллерах ARM7 и ARM9
 - Не требует контроллера памяти
 - Запускается из Flash памяти
 - Использует среду разработки Microsoft Visual Studio 2008
 - Содержит модули работы с сетью, UART, I²C, SPI, USB
 - Позволяет работать с цветными графическими индикаторами, сенсорными экранами
 - Обеспечивает низкоуровневый ввод/вывод
 - Позволяет разрабатывать устройства SideShow

.NET без операционной системы

- ▶ А, собственно, зачем нужна эта ОС?
- ▶ Преимущества работы без ОС в .NET MF
 - Приложения запускаются и работают непосредственно на железе за счёт Bootable Runtime System
 - Упрощается разработка
 - Сокращаются требования к аппаратной платформе
 - Приложения по-прежнему имеют доступ к большей части .NET API
 - Возможность разработки многопоточных приложений
 - Безопасный код
- ▶ .NET MF предоставляет исполнительную среду, но не ОС
- ▶ Даже «большой» .NET Framework не является ОС

Архитектура .NET MF

Пользовательские приложения и библиотеки

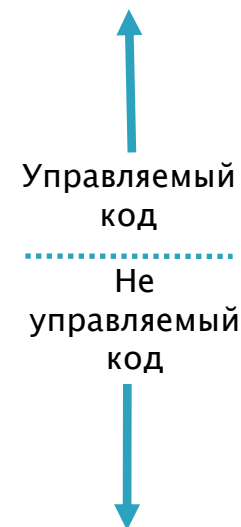
Библиотеки .NET WPF COMM ...

Общезыковая исполнительная среда Интерпретатор Система типов Сборка мусора Interop

Platform Abstraction Layer Таймеры Память Ввод/Вывод

Hardware Abstraction Layer Драйверы

Операционная система Средства



Особенности разработки и отладки

»» Часть 2

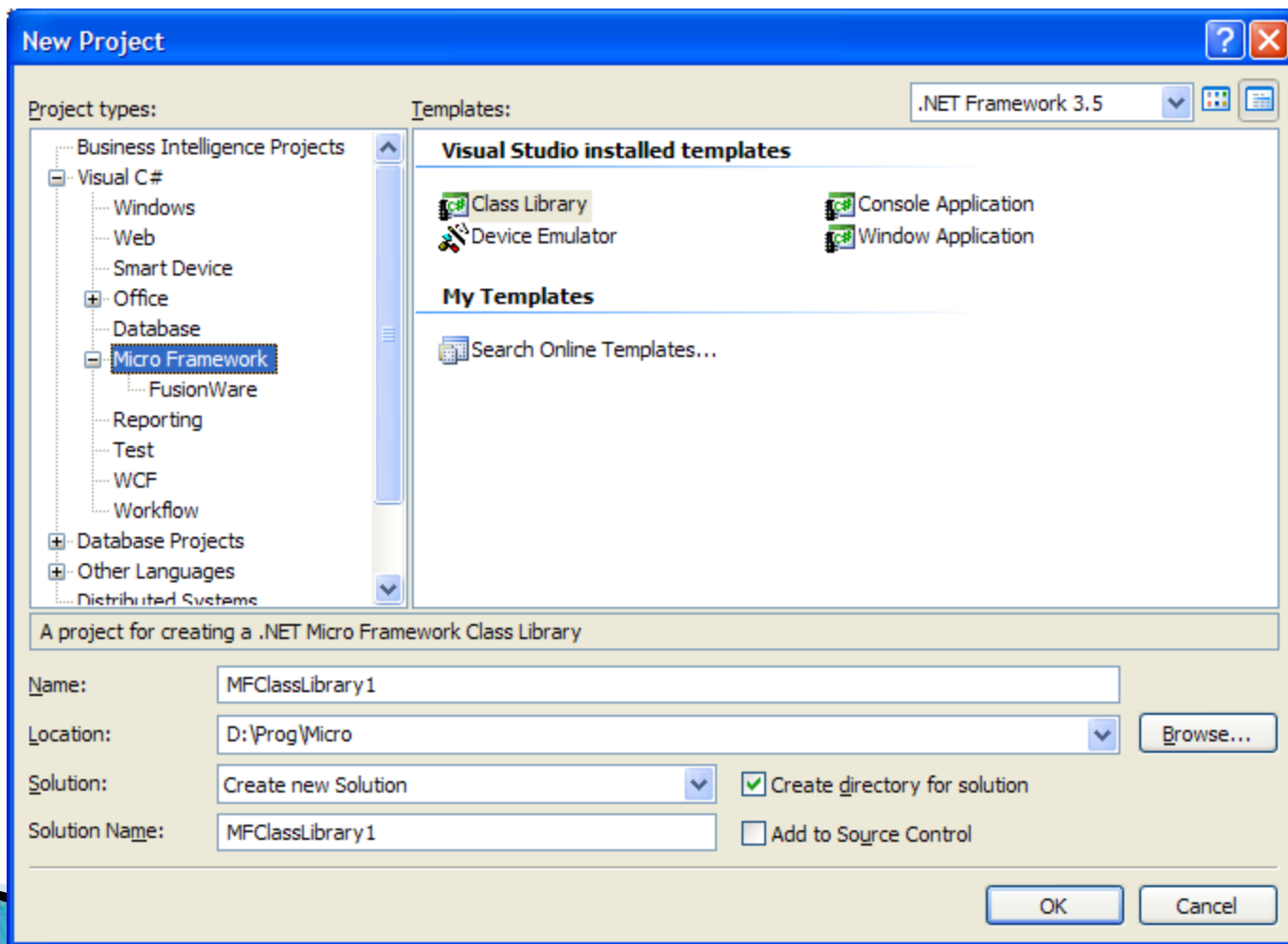
Инфраструктура разработки

- ▶ Среда разработки: Visual Studio 2008 SP1
- ▶ Необходимый пакет: .NET Micro Framework SDK
- ▶ Язык программирования: C#
- ▶ Справочная система: MSDN + Document Explorer
- ▶ Отладчик: встроенный в Visual Studio
- ▶ Вывод отладочной информации: в окно Output
- ▶ Отладка на железе: используется отладочная плата
- ▶ Отладка без железа: используется эмулятор из состава SDK к .NET Micro Framework или SDK к отладочной плате
- ▶ Инструмент обновления прошивок печатных плат: MFDeploy из состава SDK к .NET Micro Framework

.NET Micro Framework SDK

- ▶ Можно скачать на <http://www.microsoft.com/netmf>
- ▶ Состав
 - Плагин и шаблоны проектов для Visual Studio
 - Документация
 - Эмулятор
- ▶ После установки появляются новые типы проектов в разделе Visual C#/.NET Micro Framework
 - Class Library
 - Console Application
 - Window Application
 - Device Emulator

Типы проектов .NET Micro Framework



Подготовка программ в .NET MF




Отладочные платы

- ▶ Tahoe, Tahoe II
<http://www.devicesolutions.net/>
- ▶ Embedded Master Development System
<http://www.ghielectronics.com/>
- ▶ AUG AMI DevKit <http://www.aug-electronics.com/ami>
- ▶ Можно купить через сайты производителей или здесь:
www.microframework.eu



Эмуляторы

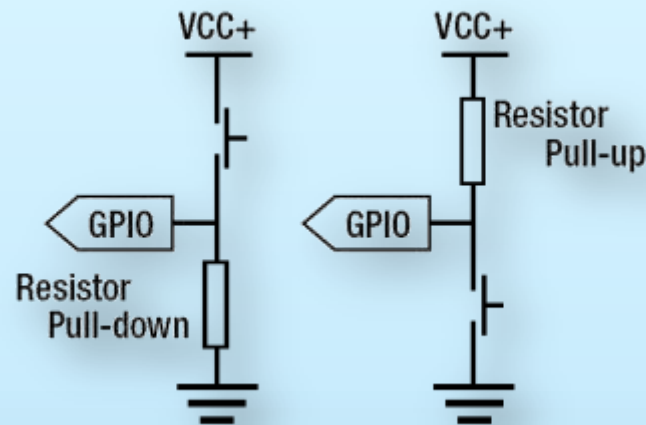
- ▶ Доступны для бесплатной загрузки с сайта производителей в составе SDK
 - ▶ Можно разработать свой собственный эмулятор с нужным функционалом
 - ▶ Разработка на .NET MF без использования отладочных плат бесплатна
- 

Низкоуровневый ввод/вывод и обработка прерываний

»» Часть 3

Пример: Работа с портом ввода-вывода

```
using System;
using System.Threading;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
namespace GpioInputPortSample
{
    public class Program
    {
        public static void Main()
        {
            InputPort inputPort = new InputPort(Cpu.Pin.GPIO_Pin2,
            false,
            Port.ResistorMode.PullDown);
            while (true)
            {
                bool state = inputPort.Read(); //polling of port state
                Debug.Print("GPIO input port at pin " + inputPort.Id +
                " is " + (state ? "high" : "low"));
                //enable device to sleep or emulator to react to Visual Studio
                Thread.Sleep(10);
            }
        }
    }
}
```



Пример: Обработка прерывания

```
using System;
using System.Threading;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
namespace GpioInterruptPortEdgeSample
{
    public class Program
    {
        public static void Main()
        {
            InterruptPort port =
                new InterruptPort(Cpu.Pin.GPIO_Pin3,
                    false, //no glitch filter
                    Port.ResistorMode.PullDown, Port.InterruptMode.InterruptEdgeBoth);
            port.OnInterrupt += port_OnInterrupt;
            Thread.Sleep(Timeout.Infinite);
        }
        private static void port_OnInterrupt(Cpu.Pin port, bool state, TimeSpan time)
        {
            Debug.Print("Pin=" + port + " State=" + state + " Time=" + time);
        }
    }
}
```

Последовательные порты

»» Часть 4

Пример: работа с I²C

```
using System;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
namespace I2CSample
{
    public class Program
    {
        public static void Main()
        {
            I2CDevice.Configuration config = new I2CDevice.Configuration(
                58, /* address */ 100 /* clockrate in KHz */ );
            I2CDevice device = new I2CDevice(config);
            byte[] outBuffer = new byte[] { 0xAA };
            I2CDevice.I2CWriteTransaction writeTransaction =
                device.CreateWriteTransaction(outBuffer);
            byte[] inBuffer = new byte[4];
            I2CDevice.I2CReadTransaction readTransaction =
                device.CreateReadTransaction(inBuffer);
            //execute both transactions
            I2CDevice.I2CTransaction[] transactions =
                new I2CDevice.I2CTransaction[] { writeTransaction, readTransaction };
            int transferred = device.Execute(transactions, 100 /* timeout in ms */ );
            //transferred bytes should be 1 + 4 = 5
        }
    }
}
```

Пример: работа с АЦП по SPI

```
using System.Threading;
using Microsoft.SPOT;
using Microsoft.SPOT.Hardware;
namespace SpiAdConverterSample
{
    public class Program
    {
        public static void Main()
        {
            ADC124S101 adc = new ADC124S101(Cpu.Pin.GPIO_Pin10, //chip select port
            5, //supply voltage
            15000, //clock rate in KHz
            SPI.SPI_module.SPI1 //first SPI bus
            );
            while (true)
            {
                float voltage = adc.GetVoltage(ADC124S101.AdcChannel.ADC1);
                Debug.Print("ADC1: " + voltage.ToString("F3") + " Volt");
                Thread.Sleep(10); //give emulator time to react to Visual Studio
            }
        }
    }
}
```


Драйвер для работы с АЦП

```
namespace SpiAdConverterSample
{
    public sealed class ADC124S101
    {
        public enum AdcChannel { ADC1, ADC2, ADC3, ADC4 };
        private readonly Cpu.Pin chipSelectPin;
        private readonly float supplyVoltage;
        private readonly SPI spi;
        private readonly ushort[] writeBuffer = new ushort[1];
        private readonly ushort[] readBuffer = new ushort[1];
        public ADC124S101(Cpu.Pin chipSelectPin, float supplyVoltage,
            uint clockRateKHz, SPI.SPI_module spiModule)
        {
            this.chipSelectPin = chipSelectPin;
            this.supplyVoltage = supplyVoltage;
            SPI.Configuration config = new SPI.Configuration(
                chipSelectPin, false, 1, 1, true, false, clockRateKHz, spiModule);
            this.spi = new SPI(config);
        }
        public float GetVoltage(AdcChannel channel)
        {
            this.writeBuffer[0] = (ushort)channel;
            this.readBuffer[0] = 0;
            spi.WriteRead(writeBuffer, readBuffer);
            ushort rawValue = readBuffer[0];
            return rawValue / 4096.0f * this.supplyVoltage;
        }
    }
}
```

Графика

»» Часть 5

Возможности по работе с графикой

- ▶ Графические примитивы
 - ▶ WPF
 - ▶ Сенсорный экран: прикосновения и жесты
 - ▶ Шрифты
 - ▶ Изображения BMP, GIF, JPG
- 

Flash-память

»» Часть 6


Хранилища данных

- ▶ Внутренняя Flash-память микроконтроллера
 - Класс Microsoft.SPOT.ExtendedWeakReference
 - Сериализация
- ▶ Карты памяти SD и USB
 - Файловая система FAT32 – обычные операции работы с файлами и папками
 - Класс Microsoft.SPOT.IO.RemovableMedia

Сеть

»» Часть 6

Сетевые возможности

- ▶ Беспроводная сеть 802.11
 - ▶ стек TCP/IP реализован полностью
 - ▶ DHCP
 - ▶ SSL
 - ▶ Сокеты
 - ▶ HTTP клиент и сервер
 - ▶ DPWS – Device Profile for Web Services
- 

Электронная почта

- ▶ .NET MF не поддерживает работу с электронной почтой
- ▶ Pavel Banský разработал компонент для отправки сообщений по SMTP
- ▶ Область имён Bansky.SPOT.Mail
- ▶ <http://bansky.net/blog/2008/08/sending-emails-from-net-micro-framework>

```
using (SmtpClient smtp = new SmtpClient("smtp.hostname.net", 25))
{
    // Send message
    smtp.Send("john@doe.com",
             "foo@bar.net",
             "Good news",
             "How are you Foo?");
}
```

Michael's Networking Toolkit

- ▶ <http://www.codeplex.com/mftoolkit>
- ▶ Автор Michael Schwarz
- ▶ Поддержка
 - DNS
 - ZigBee, XBee
 - Web, включая AJAX
 - SMTP, POP

```
Question q = new Question("ajaxpro.info", DnsType.MX, DnsClass.IN);
```

```
DnsResolver dns = new DnsResolver();  
dns.LoadNetworkConfiguration();
```

```
DnsResponse res = dns.Resolve(q);  
Console.WriteLine((res.Answers0 as MXRecord).ToString());
```

Криптография

»» Часть 7

Возможности криптографии

- ▶ Поддерживаемые алгоритмы шифрования
 - RSA – асимметричное шифрование и дешифрование, класс
Microsoft.SPOT.Cryptography.Key_RSA
 - Extended Tiny Encryption Algorithm (XTEA) – симметричное шифрование и дешифрование, класс
Microsoft.SPOT.Cryptography.Key_TinyEncryptionAlgorithm
- ▶ Возможность реализовать собственный алгоритм шифрования
 - Абстрактный класс
Microsoft.SPOT.Cryptography.Key

SideShow

»» Часть 8

Приложения SideShow


- ▶ Приложения–гаджеты, работающие на миниатюрных устройствах
- ▶ Для разработки нужны
 - SideShow Device SDK for .NET Micro Framework
 - Device Simulator
- ▶ <http://connect.microsoft.com/sideshow>



Портирование

»» Часть 9

Цели портирования

- ▶ Обеспечение взаимодействия с неуправляемым кодом
 - ▶ Перенос .NET MF на новые микроконтроллеры, устройства и операционные системы
 - ▶ Модификация существующей реализации .NET MF для конкретной платформы, например, добавление поддержки новых устройств
- 

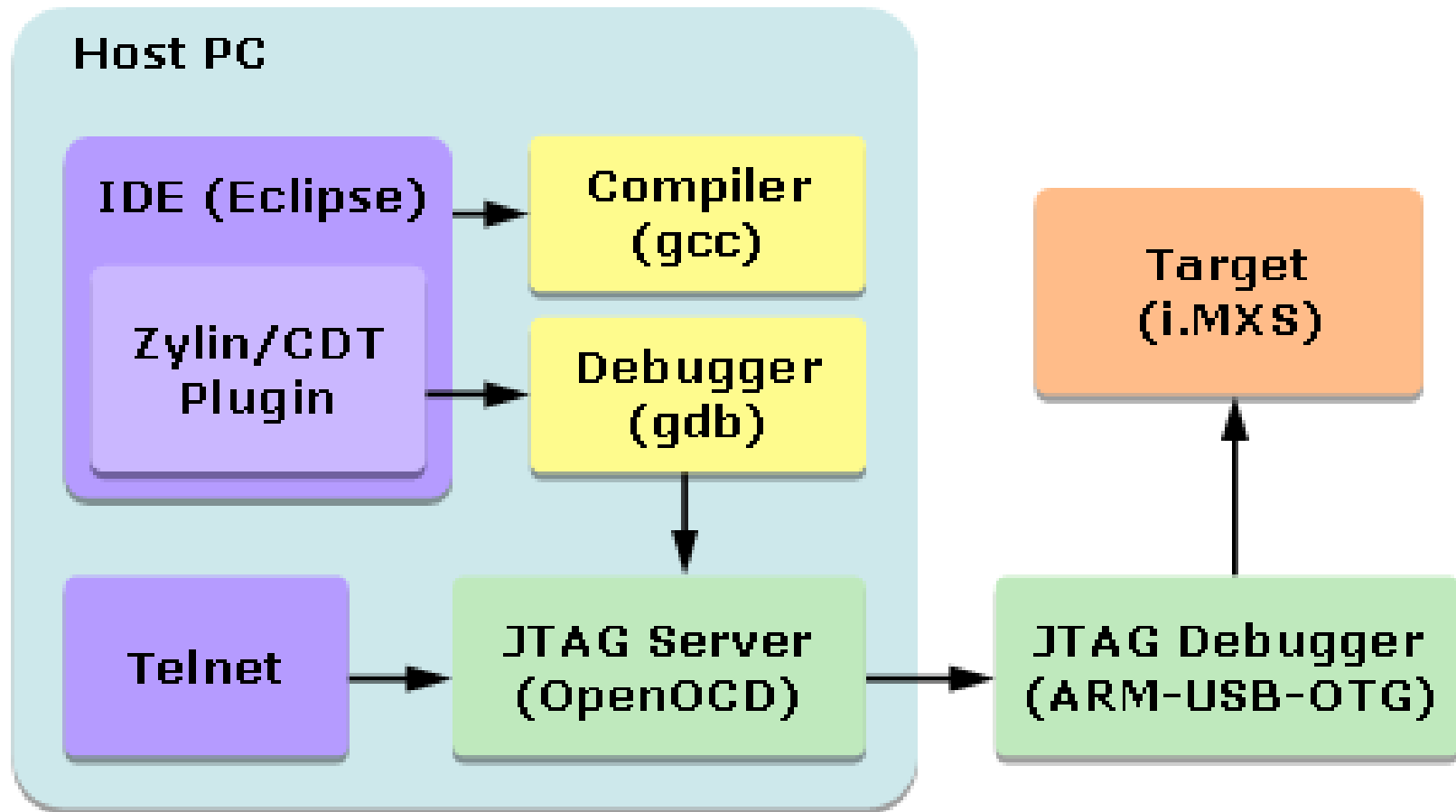
Действия при портировании

- ▶ Формирование шаблона проекта с помощью Solution Wizard, входящего в Porting Kit
 - Выбор архитектуры микроконтроллера
 - Установка параметров конфигурации платформы (параметры оперативной и энергонезависимой памяти, последовательных портов, оконечного устройства USB, системного таймера)
- ▶ Портирование HAL (примерно 120 функций)
- ▶ Портирование NativeSample (примерно 50 функций)
- ▶ Портирование PortBooter (примерно 110 функций)
- ▶ Портирование TinyBooter (примерно 70 функций)
- ▶ Добавление драйверов необходимых устройств

Инструментарий портирования

- ▶ Porting Kit
- ▶ Компилятор, отладчик C/C++
- ▶ Отладчик JTAG
- ▶ Возможный вариант
 - Компилятор GCC
 - Отладчик GDB
 - Накристалльный программный отладчик OpenOCD + программный отладчик JTAG (ARM-USB-OTG)
 - Eclipse в качестве IDE (с плагином CDT: C/C++ Development Tooling)
 - Железный отладчик Zylin (требуется обновление CDT для Eclipse <http://www.zylin.com/zylincdt>)

Инфраструктура разработки



Дополнительные ВОЗМОЖНОСТИ

»» Часть 10

Micro XAML

- ▶ .NET MF не поддерживает XAML
- ▶ Jan Kuřera разработал MicroXamlTool.exe
- ▶ <http://informatix.miloush.net/microframework/Utilities.aspx>
- ▶ Ограничение: работает для версии 3.0 Beta

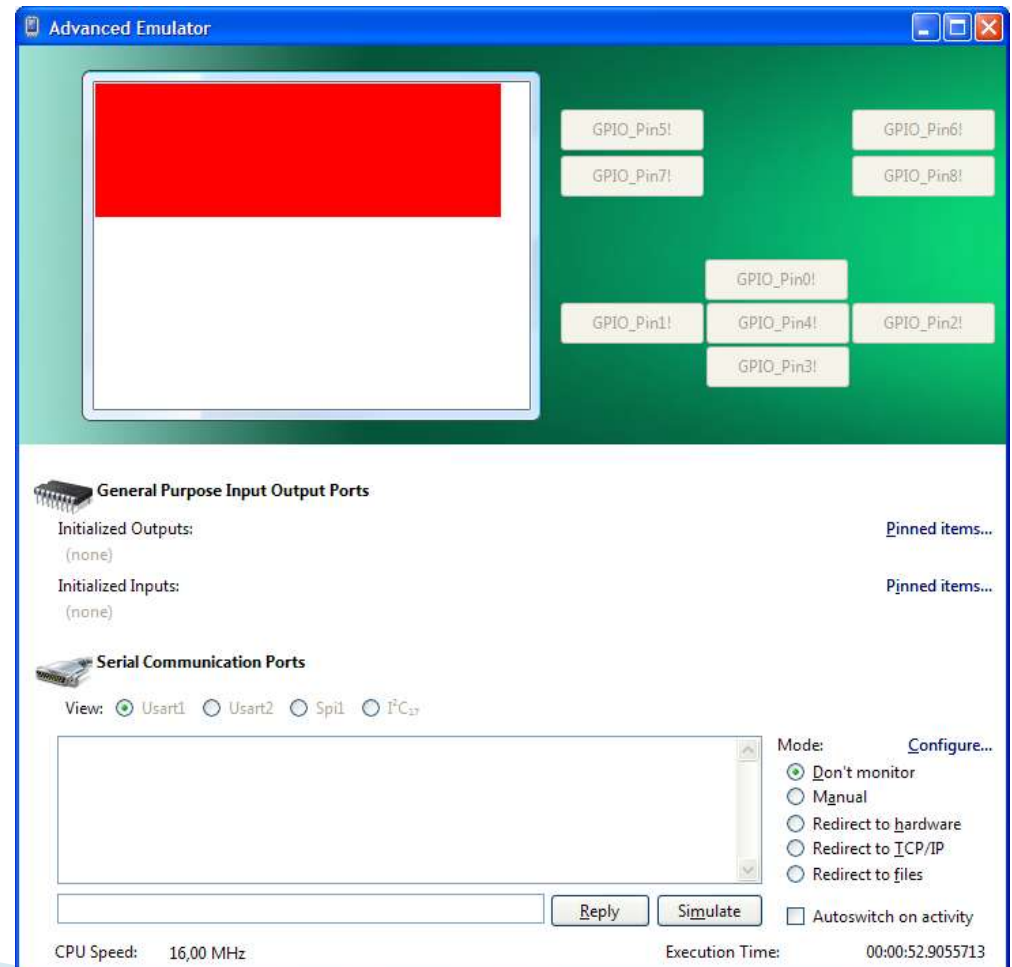
XML документация

- ▶ По непонятным причинам XML документация не поставляется в .NET MF SDK
- ▶ Jan Kuřera подготовил документацию для .NET MF 2.5
- ▶ <http://informatix.miloush.net/microframework/Utilities.aspx>

Продвинутый эмулятор

- ▶ <http://informatix.miloush.net/microframework/Utilities.aspx>

Поддерживает
.NET MF 3.0!



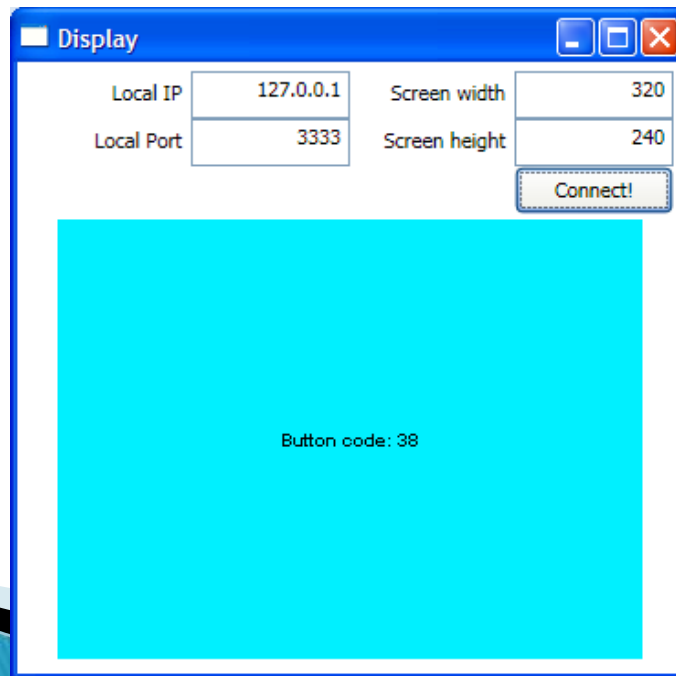
Реализация поддержки LINQ

- ▶ .NET MF не поддерживает LINQ потому что не поддерживает Generics
- ▶ Поддержка LINQ реализована в синтаксисе C#
- ▶ Как реализовать поддержку LINQ для .NET MF без Generics?
 - Задекларировать делегат предиката
 - Реализовать класс-перечислитель
 - Реализовать класс-фильтр
 - Реализовать методы расширения типовых операций, например, Where и Count
- ▶ В результате имеем реализацию LINQ to Objects в .NET MF!
- ▶ <http://blogs.oberon.ch/tamberg/2009-02-06/implementing-linq-on-the-dotnet-mf.html>

```
var a = new int[] {1, 2, 3, 4, 6, 8, 9, 9, 9};  
var n = a.Where(v => (int) v % 2 == 0).Count();  
var m = (from v in a where (int) v % 2 == 0 select v).Count();
```

Получение изображений из устройства

- ▶ Рецепт от Ondřej Piálek
- ▶ Создать на устройстве сервер, осуществляющий
 - Обработку события перерисовки экрана с получением снимка экрана
 - Прослушивание порта с отправкой изображения при перерисовке экрана
- ▶ Создать клиента, принимающего изображения
- ▶ <http://www.pialek.eu/blog/programming/dot-net-micro-framework/10-remote-control-of-a-device-streaming-display-frames-over-tcpip.html>



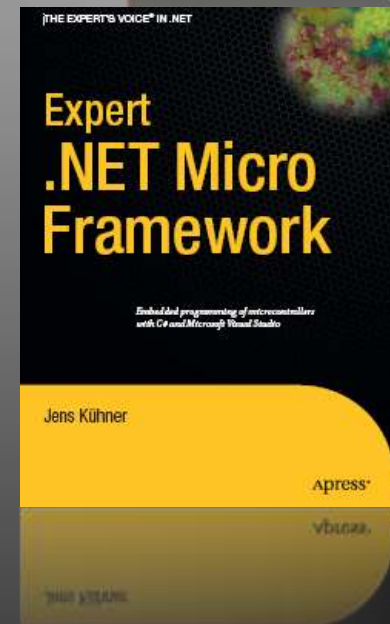
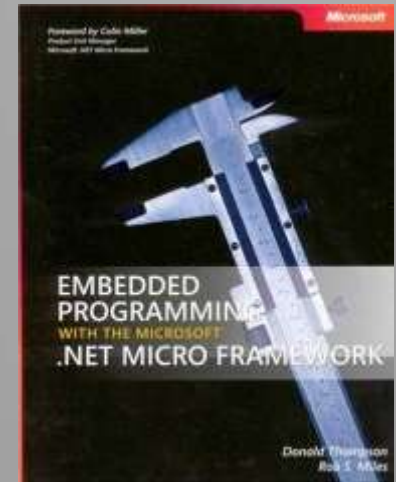
Дополнительная информация

▶ Книги

- Embedded Programming with the Microsoft .NET Micro Framework
- Apress Expert .NET Micro Framework Expert 2008

▶ Ссылки

- <http://www.microsoft.com/netmf>
- <http://www.microframework.eu>
- <http://www.netmf.ru>



Спасибо за внимание!

