The Flex Platform

Dr. Mauro Marinoni

Scuola Superiore Sant'Anna
Pisa, Italy

Email: m.marinoni@sssup.it

Graduate School on RT Kernels for Microcontrollers

dsPic architecture and

Details on the Flex board

Mauro Marinoni (RETIS Lab)
m.marinoni@sssup.it

Overview

Embedded Systems

What's an embedded system?

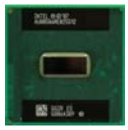
- *"An embedded system is a computer system designed to perform one or a few dedicated functions, often with real-time computing constraints. It is embedded as part of a complete device often including hardware and mechanical parts."* - **Wikipedia**

Computation Power

- Heterogeneous architectures: Harvard, MIPS, ARM, x86,…;
- Different data path width: from 8, 16, and 32;
- Width range of frequency: from some MHz up to GHz;
- With or without MMU;
- RAM: from tens of byte up to GByte;
- Different sets of I/O buses and peripherals;
- Performances oriented vs. Low Power

Computation Power (Eg.)

- **Microchip PIC10F100**
  - 8bit Harvard
  - 375Kbyte Flash
  - 16byte RAM
  - 2MHz clock
  - 0,5mW MAX
- **Intel Atom 230**
  - 32bit x86
  - No Flash
  - Up to 4Gb RAM
  - 1,6GHz clock
  - 4W



Our HW target

- **Small embedded systems:**
  - 16bit Harvard architecture;
  - <100MHz clock;
  - <1Mb Flash;
  - <64Kb RAM;
- **E.g. :** *AVR5*, *dsPIC*, *ARM9,…*

SW Architecture

- **There are several different types of software architecture in common use:**
  - Simple control loop
  - Interrupt controlled system
  - Cooperative multitasking
  - Preemptive multitasking or multi-threading
  - Microkernels and exokernels
  - Monolithic kernels

Why an embedded O.S. ?

- It reduces the complexity of the application;
- It increases the reusability of the code;
- It simplify the SW debugging;
- It reduces the time to market:
- ...

Why a Real-Time embedded O.S. ?

- An embedded applications typically presents a lot of interactions with the environment;
- That requires a management of the response time to an external event.

Our SW target

- **We focus on Embedded OS between cooperative multi-threading and microkernel:**
  - **TinyOS**
  - **NanoRK**
  - **Contiki**
  - **Erika Enterprise**

TinyOS

- Cooperative multitasking and interrupt
- Provides interrupt management and FIFO scheduling in a few hundred bytes of code
- Component-based OS written in NesC
- Used for networked wireless sensors (Motes)
- Developed by University of California (Berkeley), Intel and Crossbow Technology
- `http://www.tinyos.net`

NanoRK

- Fully preemptive reservation-based real-time OS;
- Multi-hop networking support for use in WSN;
- Runs on the FireFly Platform and the MicaZ motes;
- It supports fixed-priority preemptive multitasking, along with support for CPU, network, sensor and actuator reservations;
- Provides virtual energy reservations that allows the OS to enforce system and task level energy budgets;
- Developed by Carnegie Mellon University
- `http://www.nano-rk.org`

Nano-RK

Contiki

- Multi-tasking operating system for memory-efficient networked embedded systems and wireless sensor networks:
  - Loadable modules, multiple network stacks, multiple threading models;
- It provides IP communication, both for IPv4 and IPv6 (with the certified uIPv6 stack), and low-power radio communication mechanism ;
- It provides a software-based power profiling mechanism that keeps track of the energy expenditure of each sensor node.
- `http://www.sics.se/contiki`

Erika Enterprise

- It's a minimal real-time kernel for single and multicore embedded systems.
- It's a free, open-source implementation of the OSEK/VDX API with an OSEK OIL compiler integrated into Eclipse, and OSEK ORTI support for Lauterbach debuggers.
- It makes multi-core application development easy: hiding the complexity of the underlying architectures.
- It implements innovative scheduling algorithms such as Fixed Priority with preemption thresholds, Stack Resource Policy (SRP), and Earliest Deadline First (EDF), which can be used to schedule tasks with real-time requirements.

Erika Enterprise (kernel)

- Interface similar to the one proposed by the OSEK/VDX consortium for the OS, OIL, ORTI standards;
- RTOS API for: Tasks, Events, Alarms, Resources, Application modes, Semaphores, Error handling.
- Support for preemptive and non-preemptive multitasking;
- Support for fixed priority scheduling and Preemption Thresholds;
- Support for Earliest Deadline First (EDF) scheduling;
- Support for stack sharing techniques, and one-shot task model to reduce the overall stack usage;
- Support for hook functions before and after each context switch.
- Support for periodic activations using Alarms;
- Support for shared resources;
- Support for centralized Error Handling;

Erika Enterprise (RT-Druid)

- Development environment based on the Eclipse IDE;
- Support for the OIL language for the specification of the RTOS configuration;
- Full integration with the Cygwin development environment to provide a Unix-style scripting environment;
- Apache ANT scripting support for code generation.

Erika - supported MCU

- **Currently available as a product for:**
  - Microchip dsPIC33, dsPIC30, PIC24
  - Atmel AVR5 (ATMega128)
  - Altera Nios II
  - ARM7TDMI
- **Portings started:**
  - Freescale S12
  - Microchip PIC32
  - Lattice MICO32
- **Prototypes, with some code available on the SVN**
  - Infineon Tricore1
  - H8
  - ST10 (tiny and segmented model)
  - PPC 5xx

Microcontroller

dsPIC33f

- The dsPIC30F architecture is a modified Harvard Bus Architecture.
  - This means that the program and data memories are accessed by separate buses.
  - However, there are mechanisms to store and access constant data from the program memory space.
  - This enables more efficient use of the available on-chip memory for some applications.

Harvard architecture (1)

- The **Harvard architecture** is a computer architecture with physically separate storage and signal pathways for instructions and data.
  - no need to make the two memories share characteristics (word width, timing, implementation technology, and memory address structure)
  - instructions can be stored in read-only memory while data memory generally requires read-write memory.

Harvard architecture (2)

- The **Modified Harvard** architecture is very much like the Harvard architecture, but
  - provides a pathway between the instruction memory and the CPU that allows words from the instruction memory to be treated as read-only data.
  - This allows constant data, particularly text strings, to be accessed without first having to be copied into data memory, thus preserving more data memory for read/write variables.

dsPIC microcontroller

- The Microchip 16bit microcontrollers present:
  - 16 bit (data) modified-Harvard RISC architecture;
  - High-performance MCU capability;
  - Integrated DSP capability (dsPIC families);
  - Separate program and data buses;
  - Powerful peripherals on chip;
  - Idle & Sleep modes;
  - Switch between clock sources in real-time;
  - Flexible Interrupt structure.

dsPIC microcontroller

- Microcontrollers basic features:
  - 16-bit core up to 40 MIPS a 80 MHz;
  - Integrated DSP Module;
  - 6KB-256KB Flash Program Memory;
  - 256B-32Kb RAM Data Memory;
  - Up to 4 DMA channels;
  - Up to 8 16-bit Comparators;
  - Up to 8 PWM Generators;
  - Up to 8 16-bit Timers (32 bit mode).

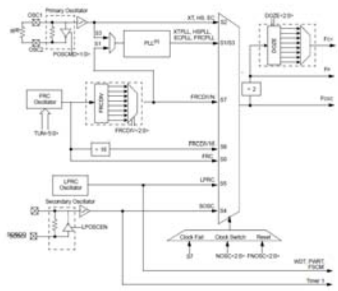
dsPIC microcontroller

- Microcontroller I/O capabilities:
  - Up to 86 I/O pins16 channels;
  - A/D module with:
    - Up to 2 A/D converters;
    - 500KSPS@12-bit
    - 1MSPS@10-bit
  - Motor Control PWMs
  - Quadrature Encoder Interface
  - Analog Comparators

dsPIC microcontroller

- Microcontroller Connectivity:
  - UART ports (IRDa support)
  - CAN bus version 2.0A/B
  - Data Convert Interface (I^2S/AC97)
  - Parallel Master Port
  - I^2C / SMBus
  - SPI bus

Frequency Issue

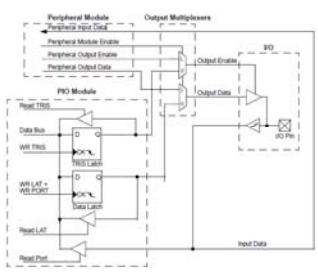


- All timings and devices depends from system clock;
- Adjustments need to be carefully managed!

MCU pins

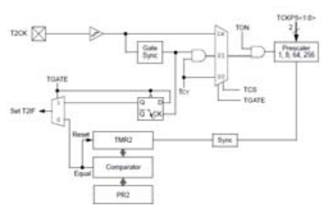
- Almost all pins work as General purpose digital IO or with specific functionalities;
- Pins destination is forced by device activation (e.g. UART, I2C) or with bitmasks in specific registers (e.g. ADC)

General Purpose IO



- GPIO are divided in ports.
- Each port has 3 registers:
  - TRIS: defines the pin as input or output;
  - PORT: value read for input pins;
  - LAT: assigned value for output pins;

Timers



- Can work in 16bit and 32bit (paired) mode;
- Different clock sources;
- Selectable Prescaler Settings;
- Overflow value register;
- Readable Timer register;
- Generate Interrupt events;

Interrupt Controller

- The **IFS** registers maintain all of the interrupt request flags. Each source of interrupt has a Status bit, which is set by the respective peripherals or external signal and is cleared via software.
- The **IEC** registers maintain all of the interrupt enable bits. These control bits are used to individually enable interrupts from the peripherals or external signals.
- The **IPC** registers are used to set the interrupt priority level for each source of interrupt. Each user interrupt source can be assigned to one of eight priority levels.

Boards Setup

Hardware Setup

- **The Flex Board**
  - **Why another evaluation board?**
- **Expansion boards**
  - **Multibus board**
  - **Demo board**
  - **Motion board**
- **Amazing ball**
- **MiniFlex Board**

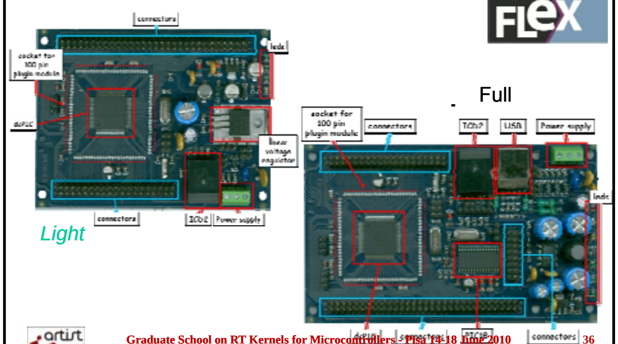
Why another Evaluation Board?

- Typically, demo boards are:
  - big!
  - limited pin counts MCU
  - most of the pins used for LEDs, buttons, …
  - difficult to expand!
  - poor connection with development PC
- **FLEX:**
  - small size (7x10 cm)
  - 100 pin dsPIC
  - all pins free on connectors
  - 2.54 pitch, no SMD expertise required!
  - PIC18 for USB connection

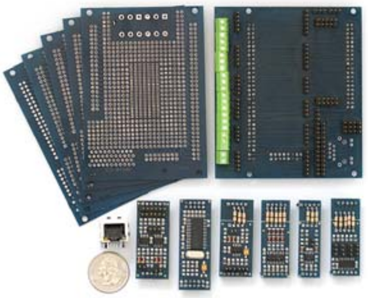
FLEX: other features

- switching power supply
- resettable fuses
- daughter boards
- support fot ERIKA O.S.

FLEX: versions



FLEX: *add-on boards*



Available:
- **Thru Hole**
- **Multibus** (CAN, SPI, I2C, Serial, Ethernet, Konnex)
- **DemoBoard**
- **MotionBoard**

FLEX: Multibus board

2 Serial ports
(RS232 / RS422 / RS485 / TP-UART)

2 CAN ports

1 I2C port

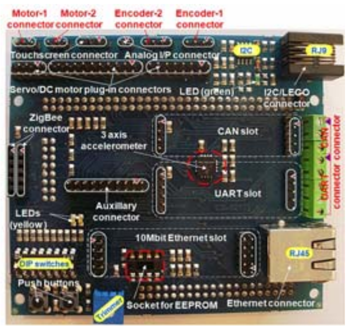
1 SPI port

1 10Mbit Ethernet

FLEX: Demo Board

3-Axis Accelerometer
Temperature sensor
2 channels DAC
Light sensor
Infrared I/O
LCD 2x16
8 LED
4 buttons
RS232/485/422 socket
802.15.4 Wireless

FLEX: MotionBoard

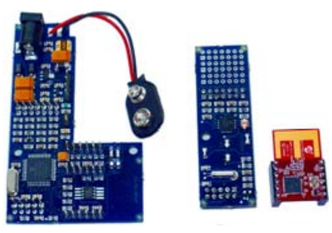


2 LED
2 buttons
1 trimmer
8bit DIP switch
3-Axis Accelerometer
2 channels Encoder
2 channels motor PWM
Ethernet socket
EEPROM socket
RS232/485/422 socket
Lego Mindstorm port

Amazing Ball

- Ball and plate where:
  - Ball position acquired with a touch screen;
  - Angles controlled with 2 servo;
- Flex + MotionBoard
- Wired and wireless connection.

MiniFlex



- PIC24FJ64JA004 (16 Mhz) microcontroller PIC
- Pins for PICkit programming
- 1KB Serial EEProm
- ZigBee
- Real-time clock
- 2 x DIP switches
- 2 x LEDs
- Buzzer
- 3-axis accelerometer
- Thermal sensor
- Light sensor
- Battery monitor
- 9 V battery + DC in jack

First Example

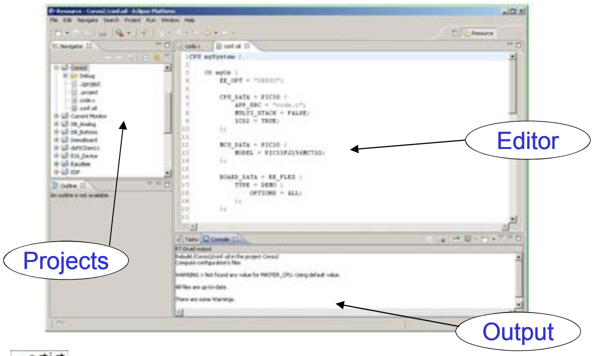
Development Chain

- To obtain a running demo:
  - *Hardware*:
    - FLEX Board;
    - Microchip ICD3;
  - *Software*:
    - Microchip MPLab IDE;
    - Microchip C30 compiler;
    - Evidence Erika Enterprise.

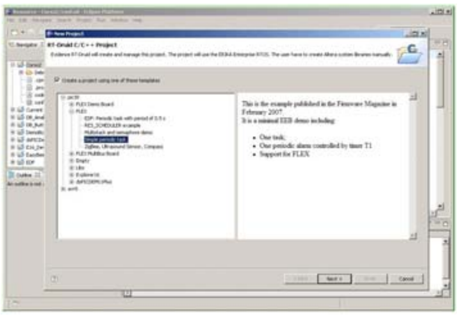
Erika Package

- It contains:
  - **Cygwin** Environment;
  - **Eclipse** 3.5;
  - **RT-Druid** plugin;
  - **Erika Enterprise** Kernel;
  - Erika support for **ScicosLab** code generator.

Eclipse Environment

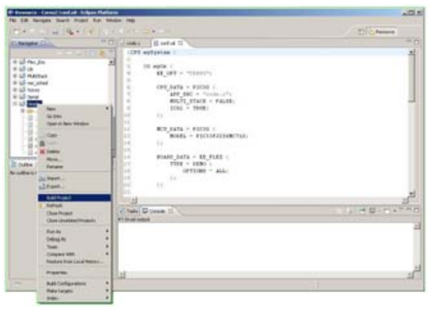


My Demo - Create from template



File → New ⟶ RT-Druid Oil and C/C++ Project

My Demo - Compile the code

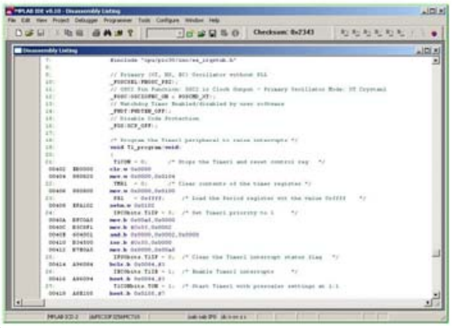


It uses makefile and C30 compiler to produce the executable for the target architecture.

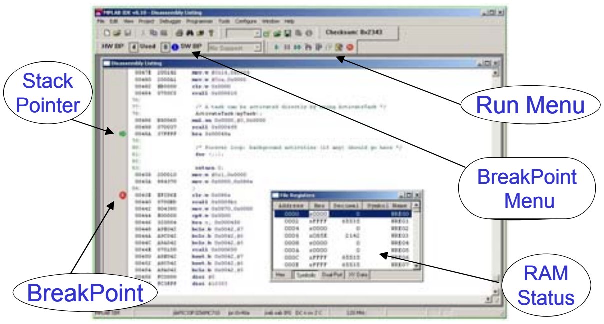
My Demo
Import and Flash in MPLab

- Configure → Select Device…
  - Choose dsPIC33FJ256MC710
- Programmer → Select Programmer →
  - MPLAB ICD2
- File → Import
  - Find the pic30.cof inside **`ProjectDir/Debug`**
- Programmer → Connect
- Programmer → Program
- Programmer → Release from Reset

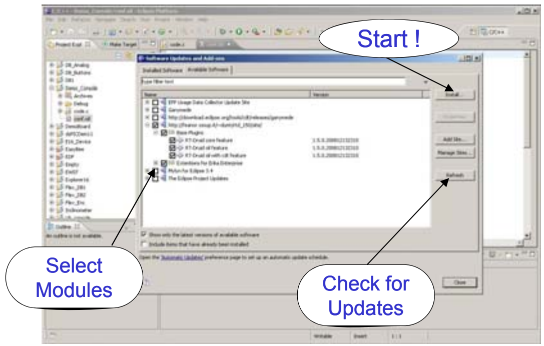
My Demo
View the code in MPLab



My Demo
Debug the code in MPLab



Online Upgrade



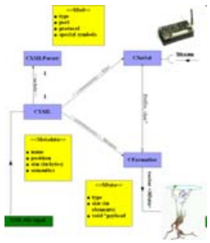
Demos

Services

- FAT FileSystem:
  - Uses SD/MMC as storage media;
  - Connected using SPI.
- Remote programming:
  - Bootloader to change the firmware without external HW;
  - Receives new binary from serial line or wireless (ongoing project);

Vision

- Visual tracking
  - Recognizes a “template” and moves the camera.
- Line camera
  - Reduces the RAM requirements;
  - Used to monitor passing objects (e.g. cars);
  - Used to detect shapes (e.g. production lines).
- Stereovision
  - Uses two cameras in order to compute distance of object in the scene.
- DAQ
  - a ROOT based Data AcQuisition platform for debugging embedded devices



Acquisition

- Inertial Measurement Unit:
  - Hardware:
    - 3-axis accelerometer;
    - 3-axis gyroscope;
    - GPS (for outdoor application);
    - Inclinometer;
  - Target:
    - Monitor angles and position both absolute and differential;
    - Compensate drift, noise and random walk to increase precision;
- Biomedical sensors:
  - Hardware:
    - ECG (Electrocardiography);
    - O^2 level
    - Breathe rate
  - Monitor people with risk of Heart Failure.

Communication

- uWireless
  - Modular stack;
  - Supports different radio, microcontroller and OS;
  - Implementation of 802.15.4;
  - Drivers for CC2420 and MRF24J40.
- MiWi
  - Lighter than 802.15.4 from Microchip
- TCP/IP
  - Porting from Microchip (Loop!!!)

Control

- SegWay
  - 2 DC motors + encoders
  - 3-axis accelerometer as inclinometer
- Ball and plate
  - 2 Servomotors
  - Touch panel
- Inverted pendulum
  - DC motor
  - 1 digital and 1 analog encoders
- 6-legged robot
  - 18 DoF with servomotors

Questions

Mauro Marinoni
[m.marinoni@sssup.it]

- Retis:
  - http://retis.sssup.it
- Evidence:
  - http://www.evidence.eu.com