Introduction to Embedded Systems Design

Introduction

- What is an Embedded System?
  - Application-specific computer system
  - Built into a larger system
- Why add a computer to the larger system?
  - Better performance
  - More functions and features
  - Lower cost
  - More dependability
- Economics
  - Microcontrollers (used for embedded computers) are high-volume, so recurring cost is low
  - Nonrecurring cost dominated by software development
- Networks
  - Often embedded system will use multiple processors communicating across a network to lower parts and assembly costs and improve reliability

Options for Building Embedded Systems

Dedicated Hardware

Software Running on Generic Hardware

Implementation	Design Cost	Unit Cost	Upgrades & Bug Fixes	Size	Weight	Power	System Speed
Discrete Logic	low	mid	hard	large	high	?	very fast
ASIC	high ($500K/ mask set)	very low	hard	tiny - 1 die	very low	low	extremely fast
Programmable logic – FPGA, PLD	low	mid	easy	small	low	medium to high	very fast
Microprocessor + memory + peripherals	low to mid	mid	easy	small to med.	low to moderate	medium	moderate
Microcontroller (int. memory & peripherals)	low	mid to low	easy	small	low	medium	slow to moderate
Embedded PC	low	high	easy	medium	moderate to high	medium to high	fast

Example Embedded System: Bike Computer

- Functions
  - Speed and distance measurement
- Constraints
  - Size
  - Cost
  - Power and Energy
  - Weight
- Inputs
  - Wheel rotation indicator
  - Mode key
- Output
  - Liquid Crystal Display
- Low performance MCU
  - 8-bit, 10 MIPS

Benefits of Embedded Computer Systems

- Greater performance and efficiency
  - Software makes it possible to provide **sophisticated control**

- Lower costs
  - Less expensive components can be used
  - Manufacturing costs reduced
  - Operating costs reduced
  - Maintenance costs reduced

- More features
  - Many not possible or practical with other approaches

- Better dependability
  - Adaptive system which can compensate for failures
  - Better diagnostics to improve repair time

ARM

Embedded System Functions

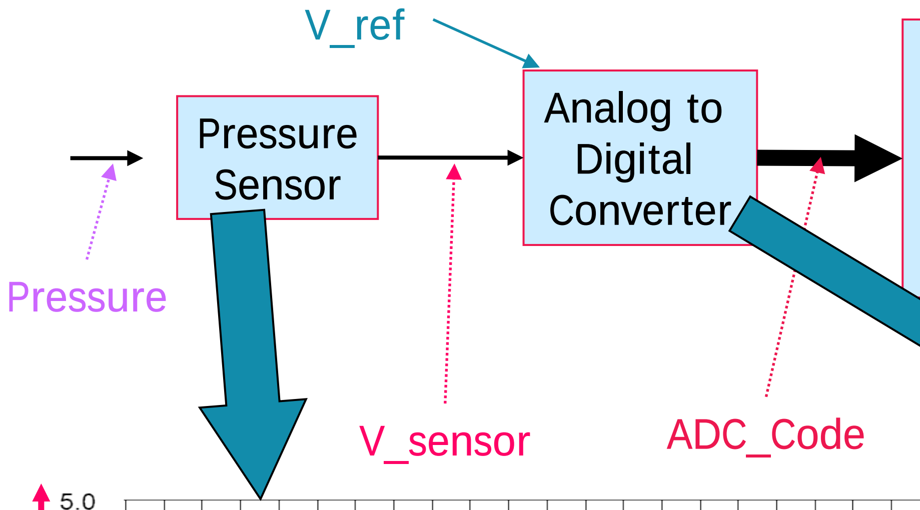
- Closed-loop control system
  - Monitor a process, adjust an output to maintain desired set point (temperature, speed, direction, etc.)
- Sequencing
  - Step through different stages based on environment and system
- Signal processing
  - Remove noise, select desired signal features
- Communications and networking
  - Exchange information reliably and quickly

Attributes of Embedded Systems

- Interfacing with larger system and environment
  - Analog signals for reading sensors
    - Typically use a voltage to represent a physical value
  - Power electronics for driving motors, solenoids
  - Digital interfaces for communicating with other digital devices
    - Simple - switches
    - Complex - displays

ARM

Example Analog Sensor - Depth Gauge




```
// Your software
ADC_Code = ADC0->R[0];
V_sensor = ADC_code*V_ref/1023;
Pressure_kPa = 250 * (V_sensor/V_supply+0.04);
Depth_ft = 33 * (Pressure_kPa – Atmos_Press_kPa)/101.3;
```

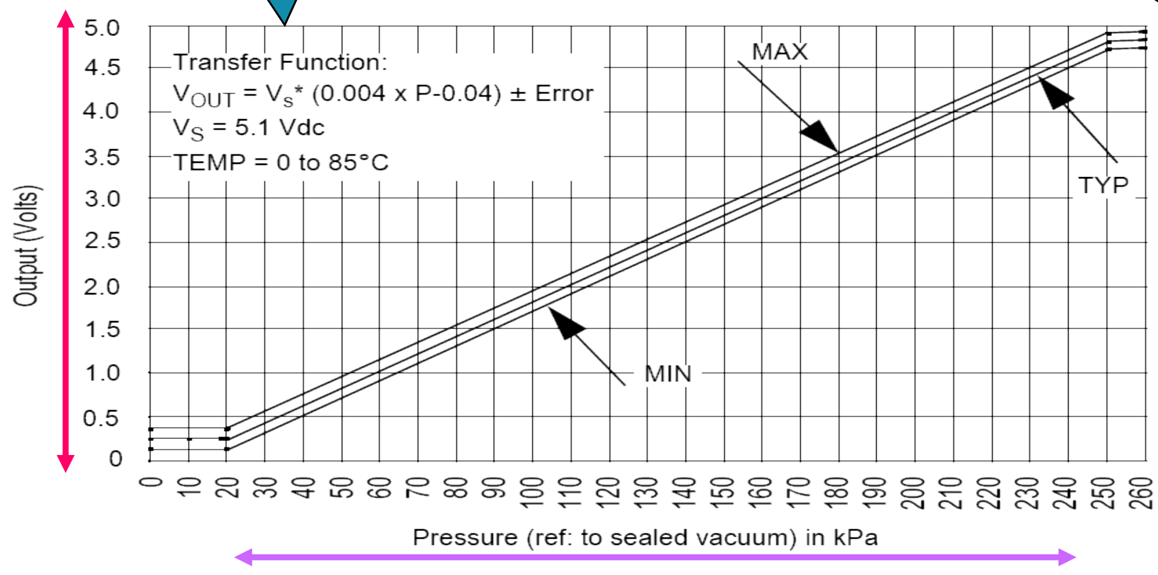
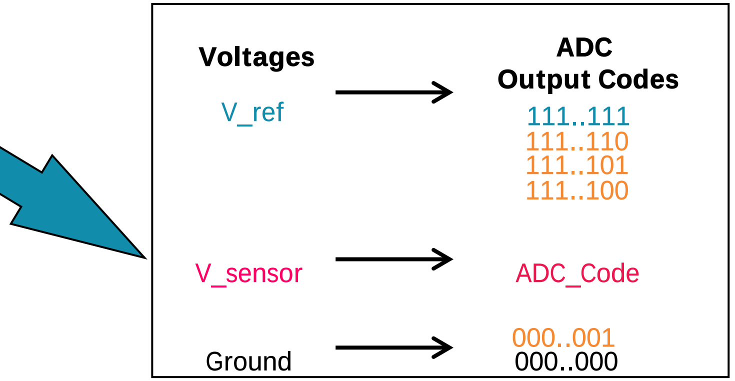


Figure 4. Output vs. Absolute Pressure

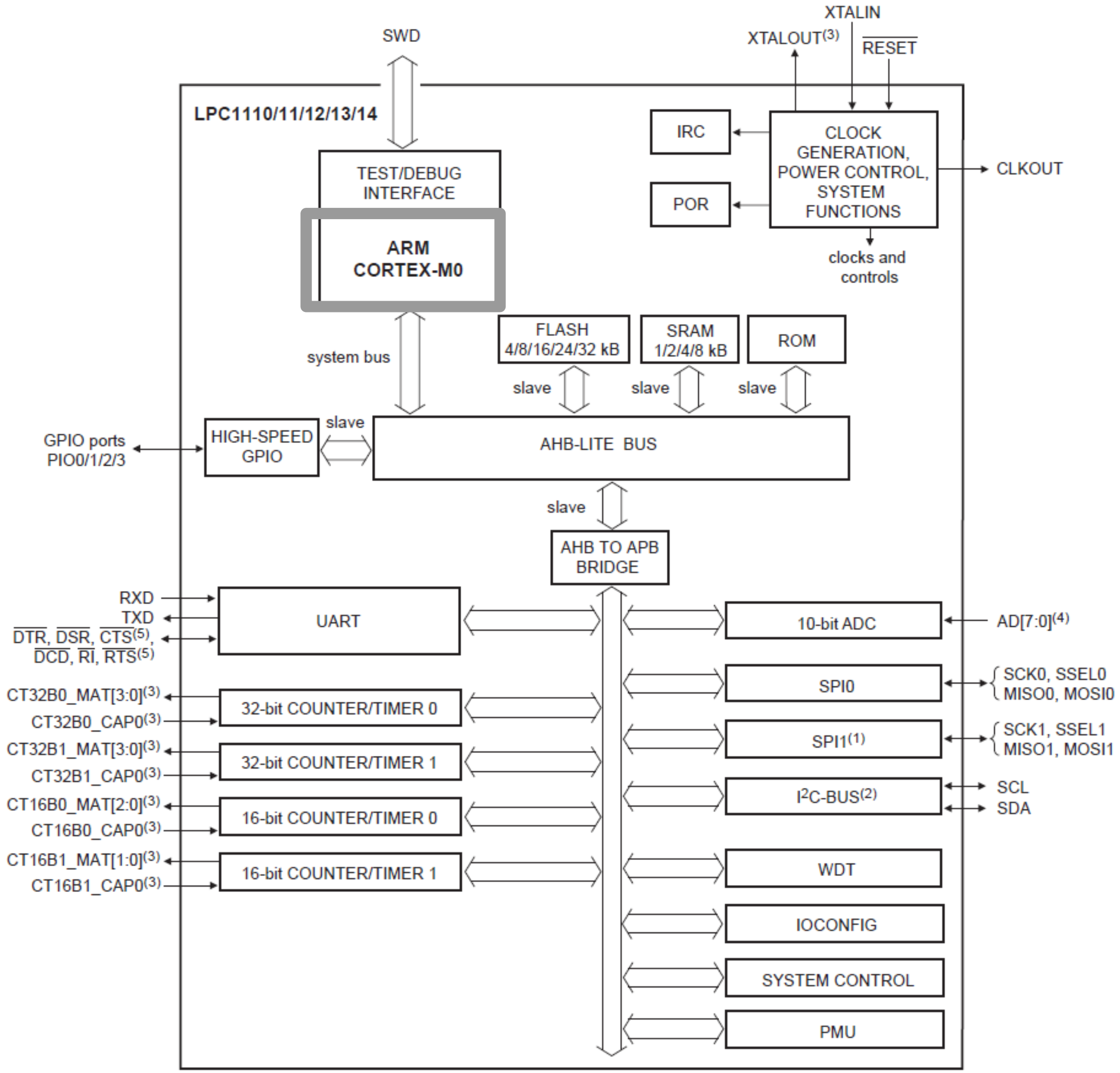


1. Sensor detects *pressure* and generates a proportional *output voltage* V_sensor
2. ADC generates a proportional digital *integer* (code) based on V_sensor and V_ref
3. Code can convert that integer to something more useful
   1. first a float representing the *voltage*,
   2. then another float representing *pressure*,
   3. finally another float representing *depth*

ARM

Microcontroller vs. Microprocessor

- Both have a CPU core to execute instructions
- Microcontroller has peripherals for concurrent embedded interfacing and control
  - Analog
  - Non-logic level signals
  - Timing
  - Clock generators
  - Communications
    - point to point
    - network
  - Reliability and safety



Microcontroller vs. Microprocessor

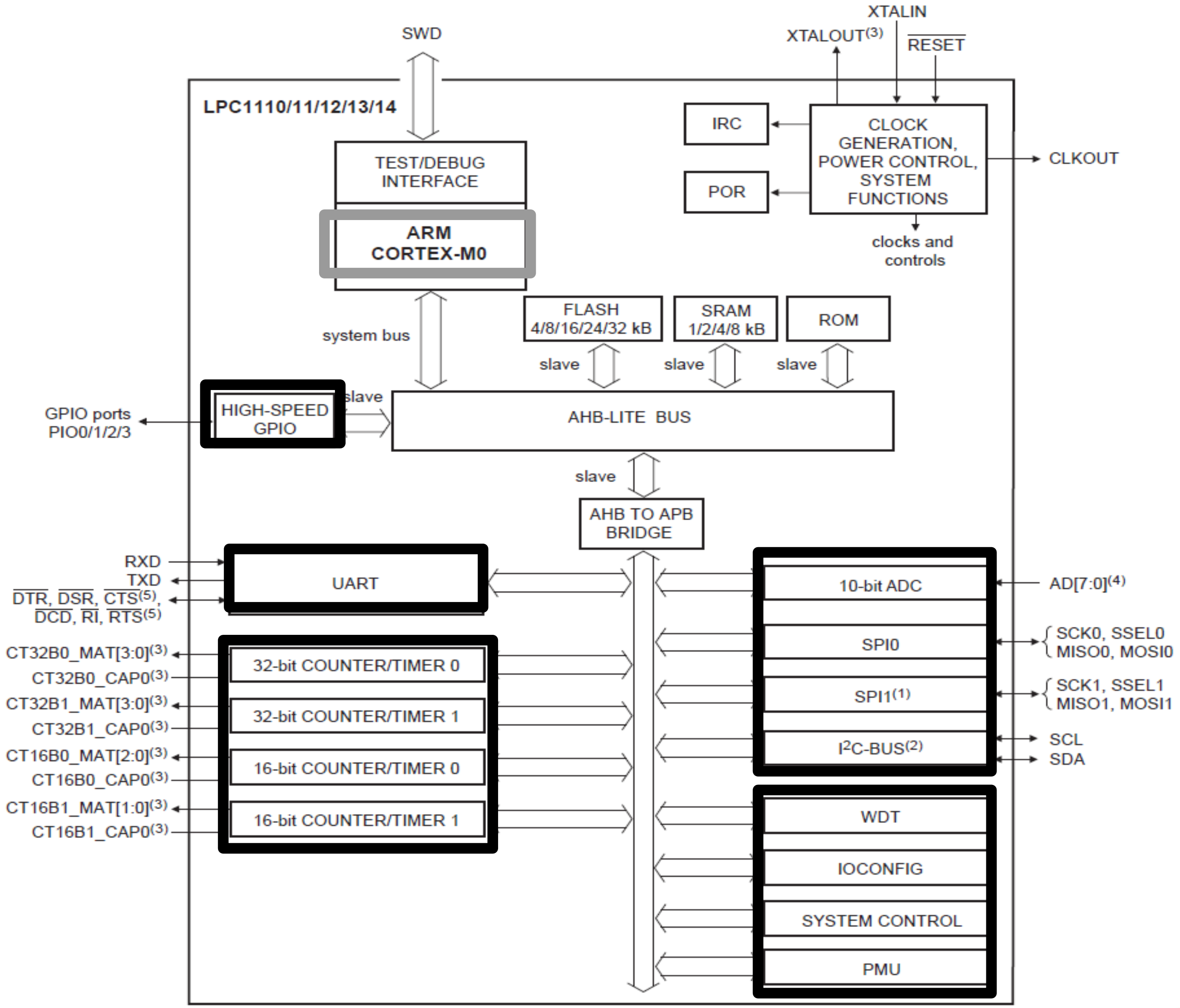
- Roughly speaking: MCU= CPU + peripherals (e.g. memory, programmable input/output peripherals)
- ARM provides ARM IPs like Cores, internal bus, interrupt controllers, etc.
- But MCUs are not created equal! MCUs from different vendors really vary due to different design decisions:
    - Architecture
    - Implementation
    - Processing optimization
    - Peripherals
    - Power management
    - Preferred tool chains
    - ……

Attributes of Embedded Systems

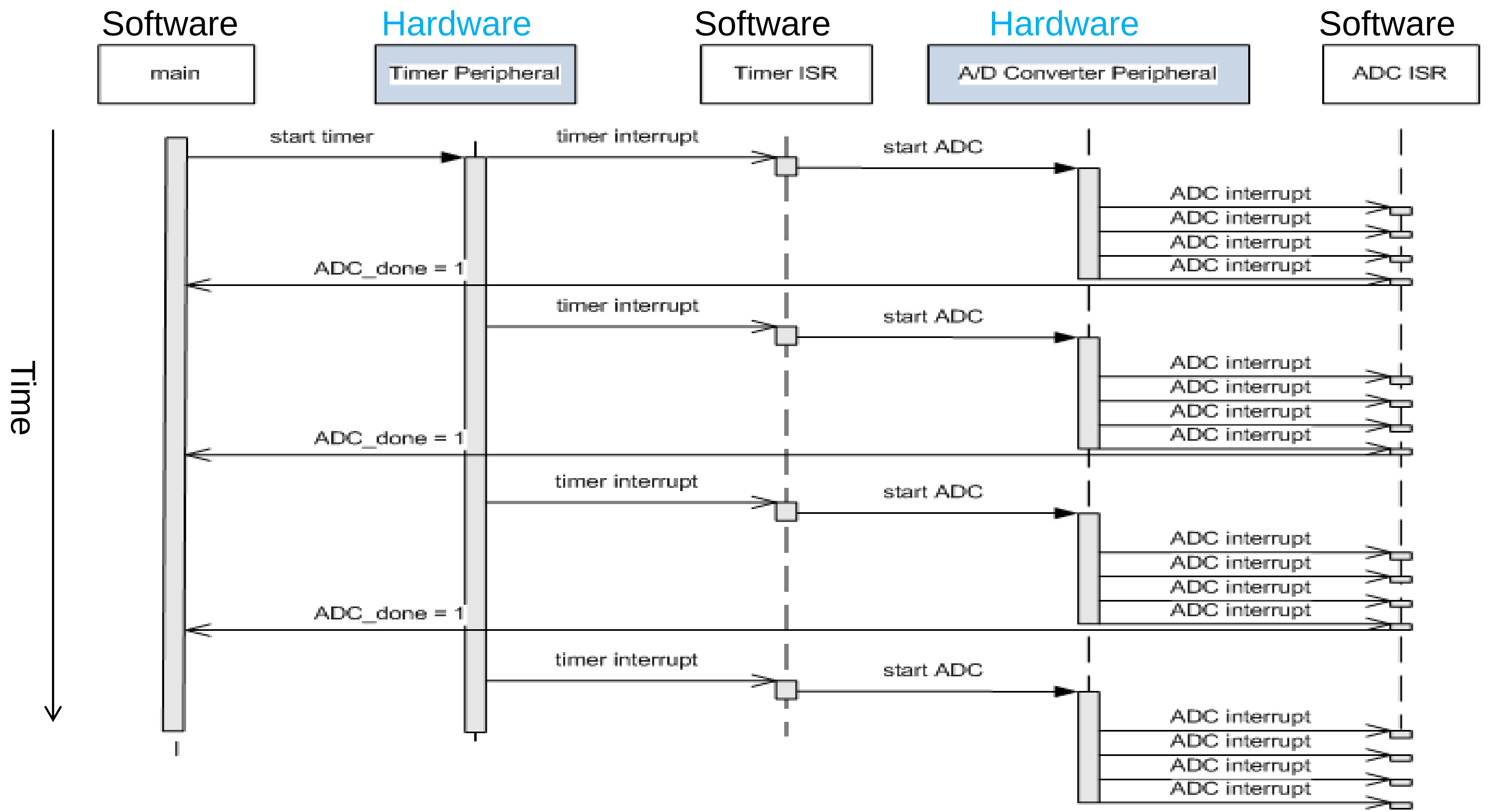
- Concurrent, reactive behaviors
  - Must respond to sequences and combinations of events
  - Real-time systems have deadlines on responses
  - Typically must perform multiple separate activities concurrently

MCU Hardware & Software for Concurrency

- CPU executes instructions from one or more thread of execution
- Specialized hardware peripherals add dedicated concurrent processing
  - Watchdog timer
  - Analog interfacing
  - Timers
  - Communications with other devices
  - Detecting external signal events
  - Power management
- Peripherals use ***interrupts*** to notify CPU of events



Concurrent Hardware & Software Operation



- Embedded systems rely on both MCU ***hardware peripherals*** and ***software*** to get everything done on time

Attributes of Embedded Systems

- Fault handling
  - Many systems must operate independently for long periods of time, requiring system to handle likely faults without crashing
  - Often fault-handling code is larger and more complex than the normal-case code
- Diagnostics
  - Help service personnel determine problem quickly

ARM

Constraints

- Cost
  - Competitive markets penalize products which don't deliver adequate value for the cost
- Size and weight limits
  - Mobile (aviation, automotive) and portable (e.g. handheld) systems
- Power and energy limits
  - Battery capacity
  - Cooling limits
- Environment
  - Temperatures may range from -40°C to 125°C, or even more

ARM

Impact of Constraints

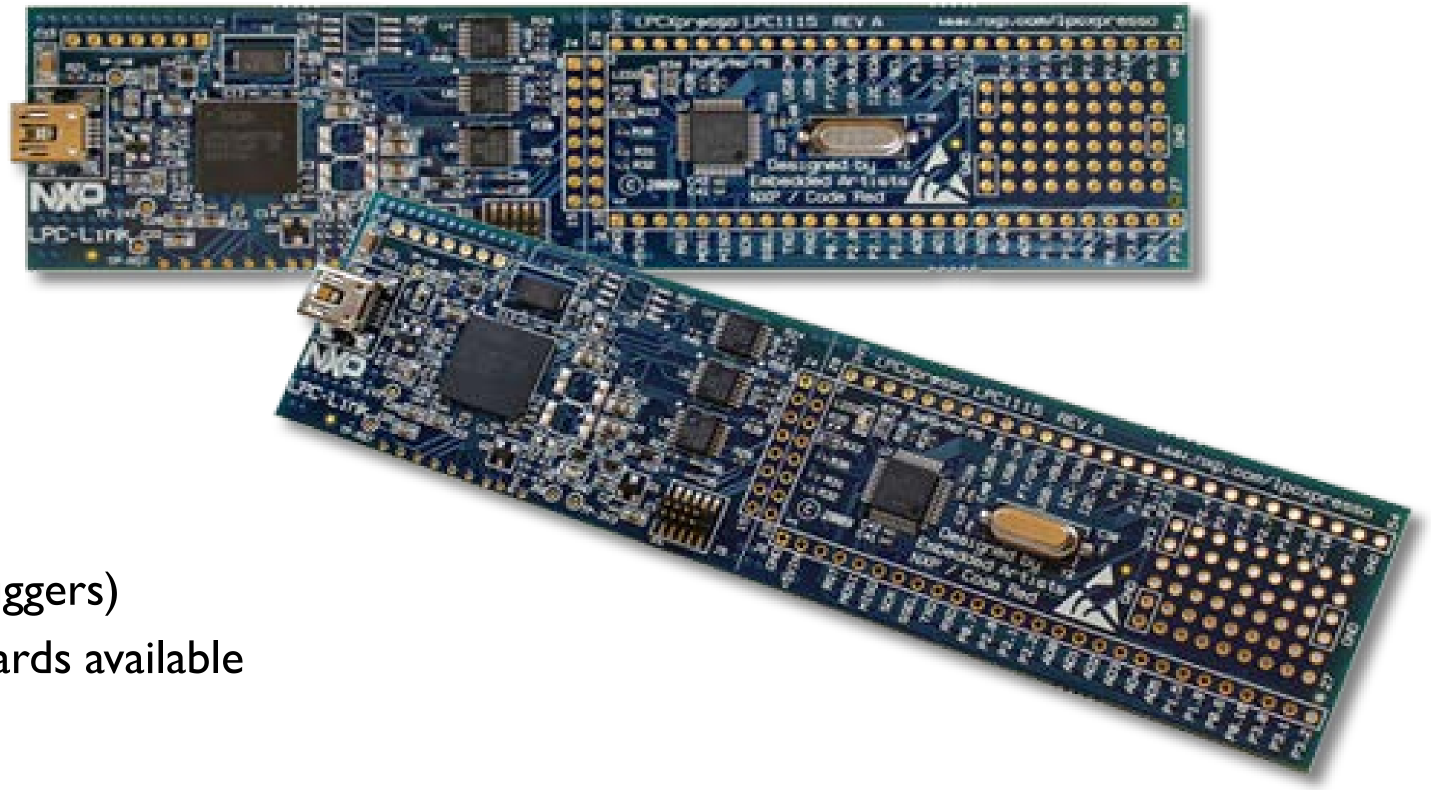
- Microcontrollers used (rather than microprocessors)
  - Include peripherals to interface with other devices, respond efficiently
  - On-chip RAM, ROM reduce circuit board complexity and cost
- Programming language
  - Programmed in C rather than Java (smaller and faster code, so less expensive MCU)
  - Some performance-critical code may be in assembly language
- Operating system
  - Typically no OS, but instead simple scheduler (or even just interrupts + main code (foreground/background system)
  - If OS is used, likely to be a lean RTOS

Curriculum Overview

- Introductory Course: Building an Embedded System with an MCU
  - Microcontroller concepts
  - Software design basics
  - ARM Cortex M0 architecture and interrupt system
  - C as implemented in assembly language
  - Peripherals and interfacing
- Advanced Course: Performance Analysis and Optimizations
  - Creating responsive systems
  - Creating fast systems
  - Optimizing system power and energy

Target Boards – EA LPC1115 LPCXpresso Board or STM Nucleo

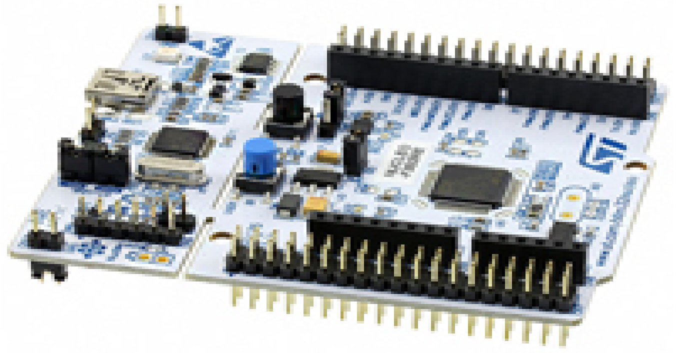
- 32-bit Cortex-M0 Processor Core
- LPC1115 in LQFP48 package
  - 50 MHz max clock
  - 64KB Flash/ 8KB RAM
  - Wide range of peripherals
- LPC1115 LPCXpresso Board
  - $25 (USD)
  - Peripherals: SSP,I2C,UART,ADC,etc.
  - Quick and easy breadboard prototyping
  - Supports various tool chains (with suitable debuggers)
  - Rich examples, libraries and extra expansion boards available from Embedded Artists and other third parties



*It will also be suitable to use LPC1114, a previous version of LPC1115, whose major difference to LPC1115 is its smaller flash. See EA’s pages for more details.

ARM

Target Boards –STM Nucleo



- Includes one STM32 microcontroller
- • On-board ST-LINK debugger/ programmer: Virtual COM port
- Mass storage
- • Wide extension capabilities with specialized shields: Arduino Uno rev3 connectors on Nucleo-64 and Nucleo-144
- Access to a wider range of peripherals through Zio connectors on Nucleo-144
- Access to all MCU pins through ST morpho connectors on Nucleo-64 and Nucleo-144
- Arduino Nano connectors on Nucleo-32

- • Direct access to Mbed online resources for most boards
- • Supported by IAR, Arm Keil, Arm® Mbed™ online,

ARM

Why Are We…?

- Using C instead of Java (or Python, or your other favorite language)?
  - C is the de facto standard for embedded systems because of:
    - Precise control over what the processor is doing.
    - Modest requirements for ROM, RAM, and MIPS, so much cheaper system
    - Predictable behavior, no OS (e.g. Garbage Collection) preemption
- Learning assembly language?
  - The compiler translates C into assembly language. To understand whether the compiler is doing a reasonable job, you need to understand what it has produced.
  - Sometimes we may need to improve performance by writing assembly versions of functions.
- Required to have a microcontroller board?
  - The best way to learn is hands-on.
  - You will keep these boards after the semester ends for possible use in other projects (e.g. Senior Design, Advanced Embedded System Design, Mechatronics, etc.)

ARM

References

- ARMv6-M Architecture Reference Manual
- The Definitive Guide to ARM® Cortex®-M0
- Cortex-M0 Technical Reference Manual
- Cortex-M0 Devices Generic User Guide
- LPC111x Product Data Sheet
- LPC111x User Manual
- LPC1114 Board Schematics
- Application Note 237
  - This introduces how to set up the development environment with ARM-MDK and LPC-Link 2.