Queen Mary
University of London
Science and Engineering

School of Electronic Engineering and Computer Science
QMUL-BUPT Joint Programme

EBU6475 Microprocessor System Design
EBU5476 Microprocessors for Embedded Computing

Introduction to Embedded Systems Design

arm

Last updated: 15 February, 2020

University Program Education Kits

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

	Implementation	Design Cost	Unit Cost	Upgrades & Bug Fixes	Size	Weight	Power	System Speed
Dedicated Hardware	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
Software Running on Generic Hardware	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

Example: Motor Control Unit

- Functions
  - Motor control
  - System communications
  - Current monitoring
  - Rotation speed detection
- Constraints
  - Reliability in harsh environment
  - Cost
  - Weight
- Many Inputs and Outputs
  - Discrete sensors & actuators
  - Network interface to rest of car
- High Performance MCU
  - 32-bit, 256 KB flash memory, 80 MHz

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

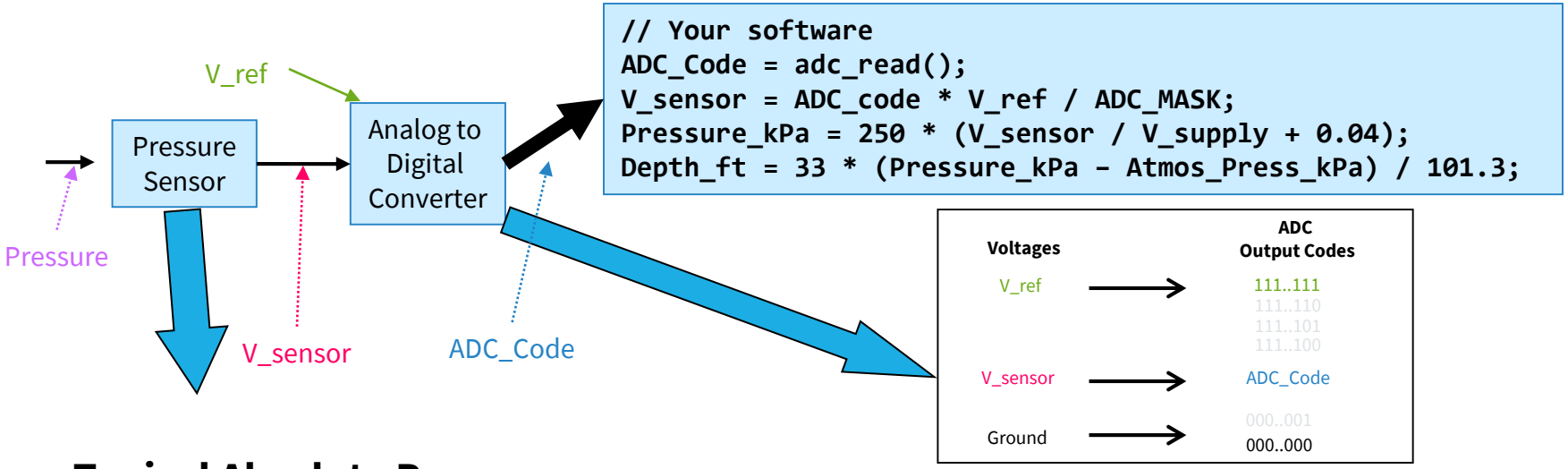
Embedded System Functions

- 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

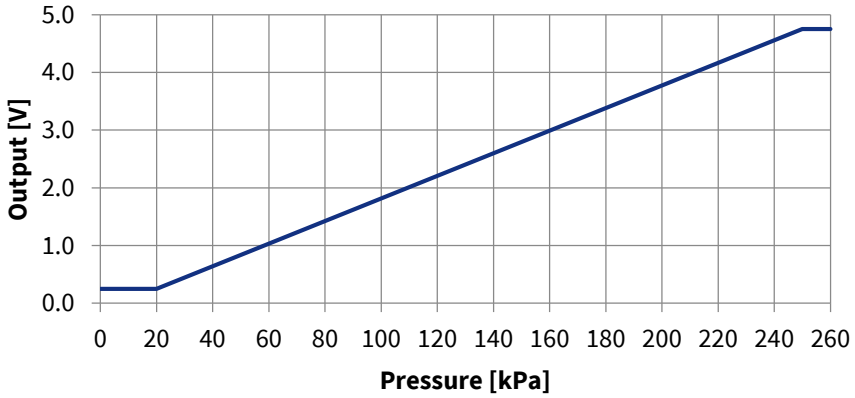
Attributes of Embedded Systems

- 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

Example: Analog Sensor - Depth Gauge



Typical Absolute Pressure vs. Ouput



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

Microcontroller vs. Microprocessor

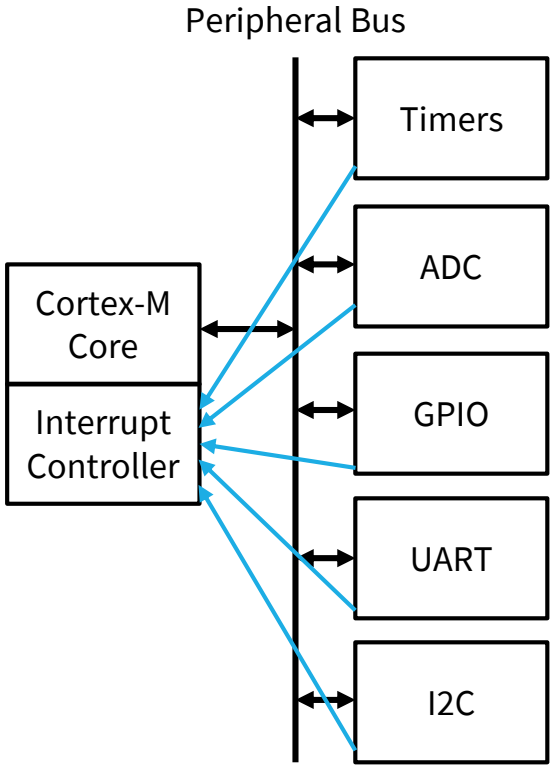
- Both have a CPU core to execute instructions
- Microcontroller has on-chip peripherals for concurrent embedded interfacing and control
  - Analog
  - Non-logic level signals
  - Timing
  - Clock generators
  - Communications
  - Reliability and safety

Attributes of Embedded Systems

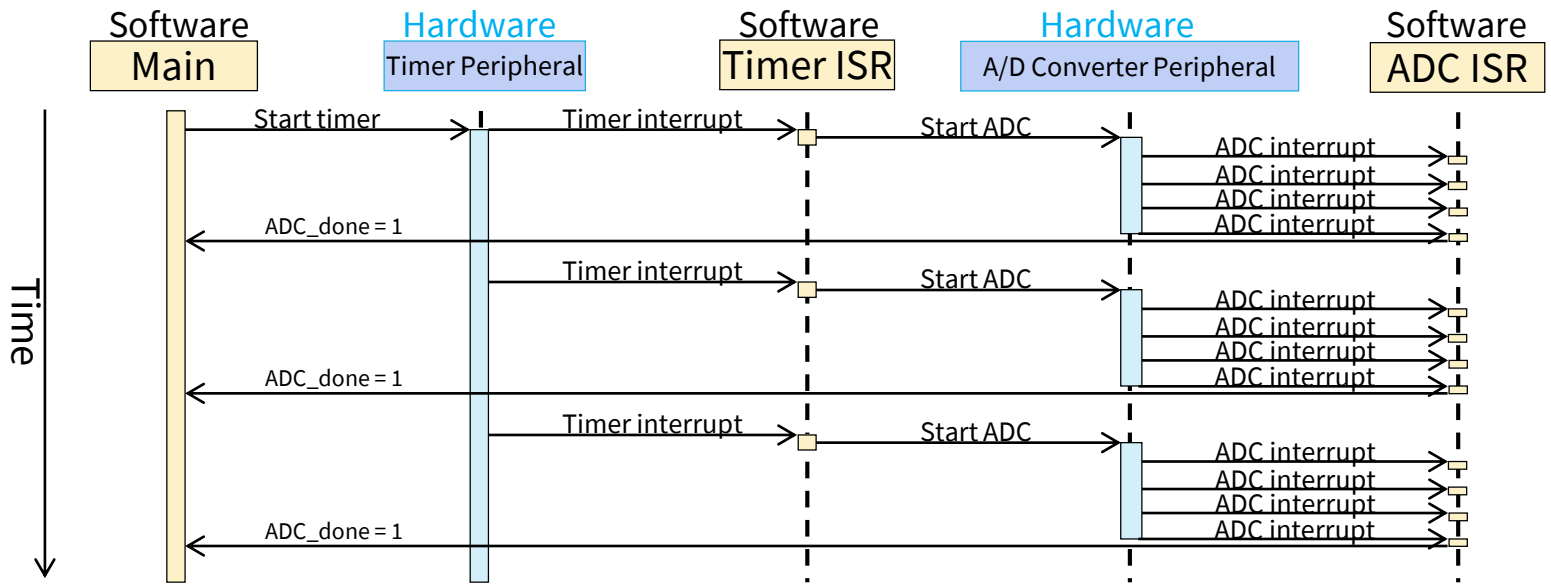
- Concurrent, reactive behaviors
  - Continuously react to inputs from the environment by generating corresponding outputs
  - Must respond to sequences and combinations of events
  - Real-time systems have deadlines on responses
  - Typically must perform multiple separate activities concurrently
  - Typical applications include vehicle control, consumer electronics, remote sensing, smart household aplliances etc.

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
  - LCD driver
- 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

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
- Environment
  - Temperatures may range from -40°C to 125°C, or even more
  - Cooling limits

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

Overview of this module

- Introductory Course:

- Building an Embedded System with an MCU
  - Microcontroller concepts
  - Processor core architecture and interrupt system
  - C as implemented in assembly language
  - Peripherals and interfacing

Why Are We…?

- Using C instead of Java (or Python, or your other favourite 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 needed

- 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.

Short Quiz (1)

- Which of the following includes on-chip peripherals?
  (a) Microcontroller (MCU)
  (b) Microprocessor (MPU)
- When building a system, which hardware is the most suitable if targeting a specific application for optimization in size, footprint, and power consumption?
  (a) ASIC
  (b) Embedded PC
  (c) Programmable logic (FPGA, PLD)
  (d) Microcontroller (internal memory and peripherals)
- Which programming language is most commonly used for programming microcontrollers?
  (a) Java/Swift 3
  (b) Assembly language
  (c) C and assembly language
  (d) All of the above

Short Quiz (2)

- How does an embedded system usually represent logic values physically?
  (a) Different current levels
  (b) Different voltage levels
  (c) None of these
  (d) Both (a) and (b) are correct
- In which of the following applications can the MCU not be used as the main processor?
  (a) Automotive (brakes, speed control, etc)
  (b) Smart Watch
  (c) Electronic home devices (washing machines, microwaves, etc)
  (d) None of the above
- How can a peripheral notify a CPU of events?
  (a) Using exceptions
  (b) Using interrupts
  (c) Using polling
  (d) None of the above