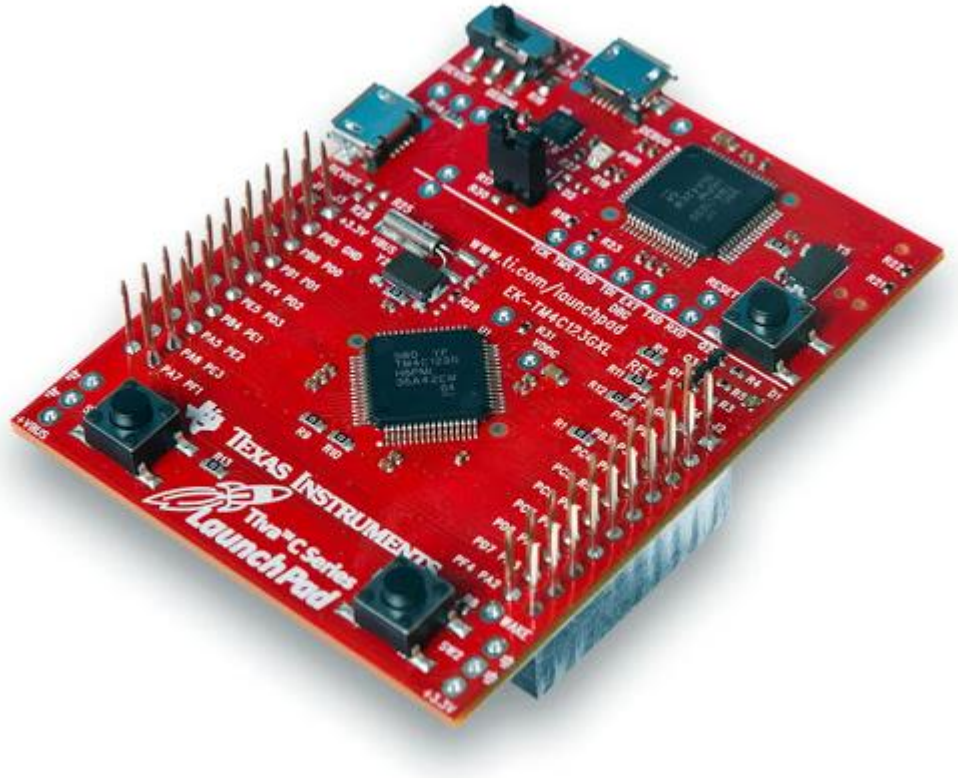
# Embedded Programming and the ARM Cortex-M4



# Motivation

What is embedded programming?



SDU

# Motivation

What is embedded programming?

Programming of embedded systems.

What are embedded systems?



SDU

# Motivation

What is embedded programming?

Programming of embedded systems.

What are embedded systems?

An embedded system is a computer system—a combination of a computer processor, computer memory, and input/output peripheral devices—that has a dedicated function within a larger mechanical or electrical system.

# Motivation

Why is embedded programming important?



SDU

# Motivation

What devices contain embedded systems?



SDU

# ARM architecture

- What is it?
- Acorn RISC Machine, later Advanced RISC Machine
- RISC – reduced instruction set computer



SDU

# RISC – reduced instruction set computer

- Small set of simple and general instructions
- An improvement over CISC (complex instruction set computer)
- Advantages
  - Instructions take one cycle time
  - Performance is better due to simplified instruction set
  - Less chip space is used due to reduced instruction set
  - Can be easily designed as compared to CISC
  - Reduced per-chip cost, as it uses smaller chips
- Disadvantages of RISC Architecture:
  - Performance of the processor will vary according to the code being executed.
  - RISC processors require very fast memory systems to feed various instructions.



SDU

# RISC vs CISC examples

- CISC – complex instructions set computer
  - Intel x86 – in every PC

- RISC – reduced instruction set computer
  - Qualcomm Snapdragon – in every (?) smartphone

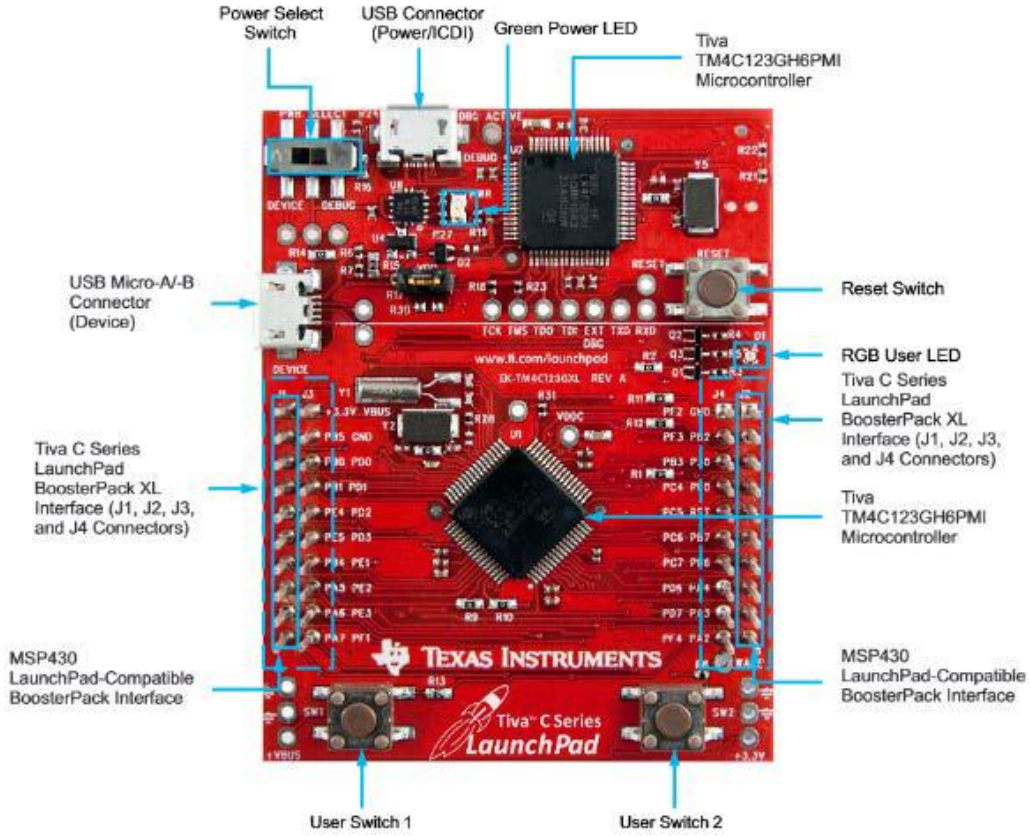


SDU

# ARM

- Good performance per Watt
  - Energy efficient – ideal for mobile/embedded use
- Good price for performance
- Made successful by Apple with their early handheld devices (PDAs)

# Tiva C Series TM4C123G LaunchPad Evaluation Board



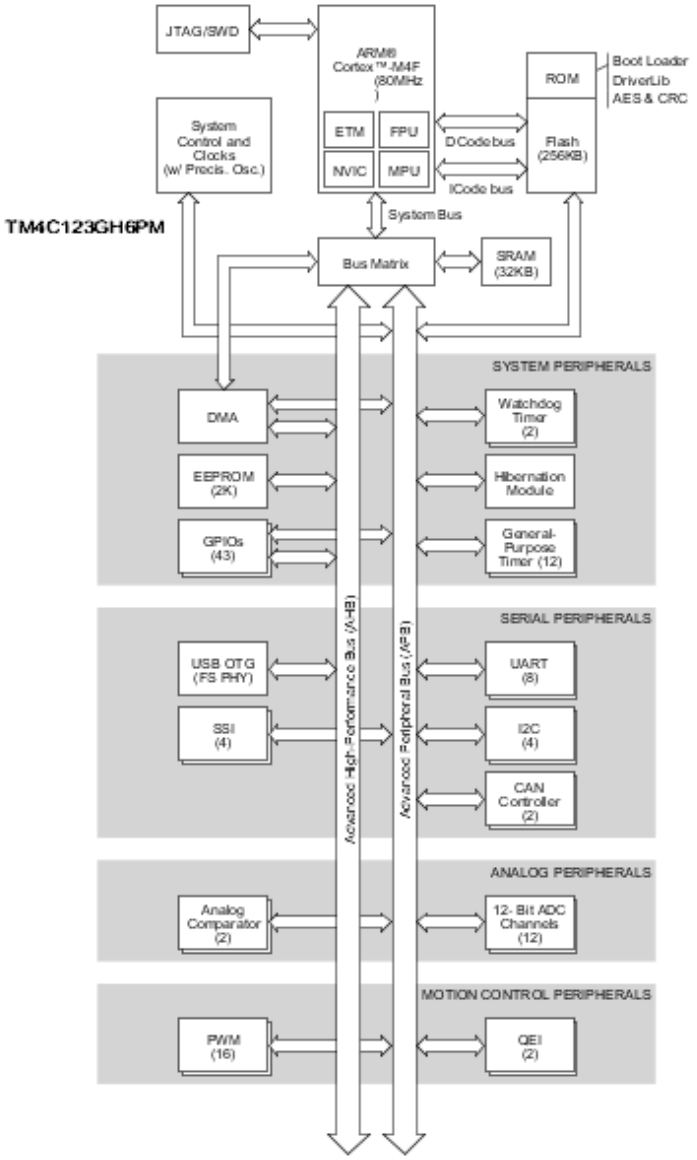
# Tiva C Series TM4C123G LaunchPad Evaluation Board

- the Tiva C Series Launch Pad includes the following features:
  - Tiva TM4C123GH6PMI microcontroller
  - Motion control PWM
  - USB micro-A and micro-B connector for USB device, host, and on-the-go (OTG) connectivity
  - RGB user LEDs
  - Two user switches (application/wake)
  - Available I/O brought out to headers on a 0.1-in (2.54-mm) grid
  - On-board ICDI (in-circuit debugger interface)
  - Switch-selectable power sources:
    - ICDI
    - USB device
  - Reset switch



SDU

# TM4C123GH6PM Microcontroller

# Features of the TM4C123GH6PM

- 32-bit ARM® Cortex™-M4 80-MHz processor core.
- On-chip memory, featuring 256 KB flash, 32 KB SRAM, ROM, 2KB EEPROM.
- Two Controller Area Network (CAN) modules.
- Universal Serial Bus (USB 2.0), and USB OTG/Host/Device mode.
- Advanced serial integration, featuring: eight UARTs
- ARM PrimeCell® 32-channel configurable µDMA (direct memory access) controller.
- Analog support, featuring: two 12-bit ADC (analog to digital converter)
- Advanced motion control, featuring: eight PWM (pulse width modulation) generator blocks.
- Six 32-bit, six 64-bit general-purpose timers and two watchdog timers.
- Up to 43 GPIOs (general purpose input output pins).
- Lower-power battery-backed Hibernation module with Real-Time Clock
- Multiple clock sources for microcontroller system clock.
- Full-featured debug solution.
- Industrial-range (-40°C to 85°C)
  RoHS-compliant 64-pin LQFP (low profile quad flat package)



SDU

# ARM Cortex-M4

- great processing performance combined with fast interrupt handling
- enhanced system debug with extensive breakpoint and trace capabilities
- efficient processor core, system and memories
- ultra-low power consumption with integrated sleep mode and an optional deep sleep mode
- platform security robustness, with optional integrated Memory Protection Unit (MPU).

# ARM Cortex-M4 peripherals

- Nested Vectored Interrupt Controller -The NVIC is an embedded interrupt controller that supports low latency interrupt processing.
- System Control Block - is the programmers model interface to the processor. It provides system implementation information and system control, including configuration, control, and reporting of system exceptions.
- System timer - SysTick, is a 24-bit count-down timer. Use this as a Real Time Operating System (RTOS) tick timer or as a simple counter.
- Memory Protection Unit (MPU) - improves system reliability by defining the memory attributes for different memory regions. It provides up to eight different regions, and an optional predefined background region.
- Floating-point Unit (FPU) provides IEEE754-compliant operations on single-precision, 32-bit, floating-point values.



SDU

# Registers - reminder

- A processor register (CPU register) is one of a small set of data holding places that are part of the computer processor.
- A register may hold an instruction, a storage address, or any kind of data (such as a bit sequence or individual characters).
- Some instructions specify registers as part of the instruction.

Intel x86 architecture

```
add AX,BX
inc   BX            ; increment BX by one
cmp BX, 1000        ; compare BX with a constant (1000 decimal)
jbe  again          ; jump to label if below or equal
```

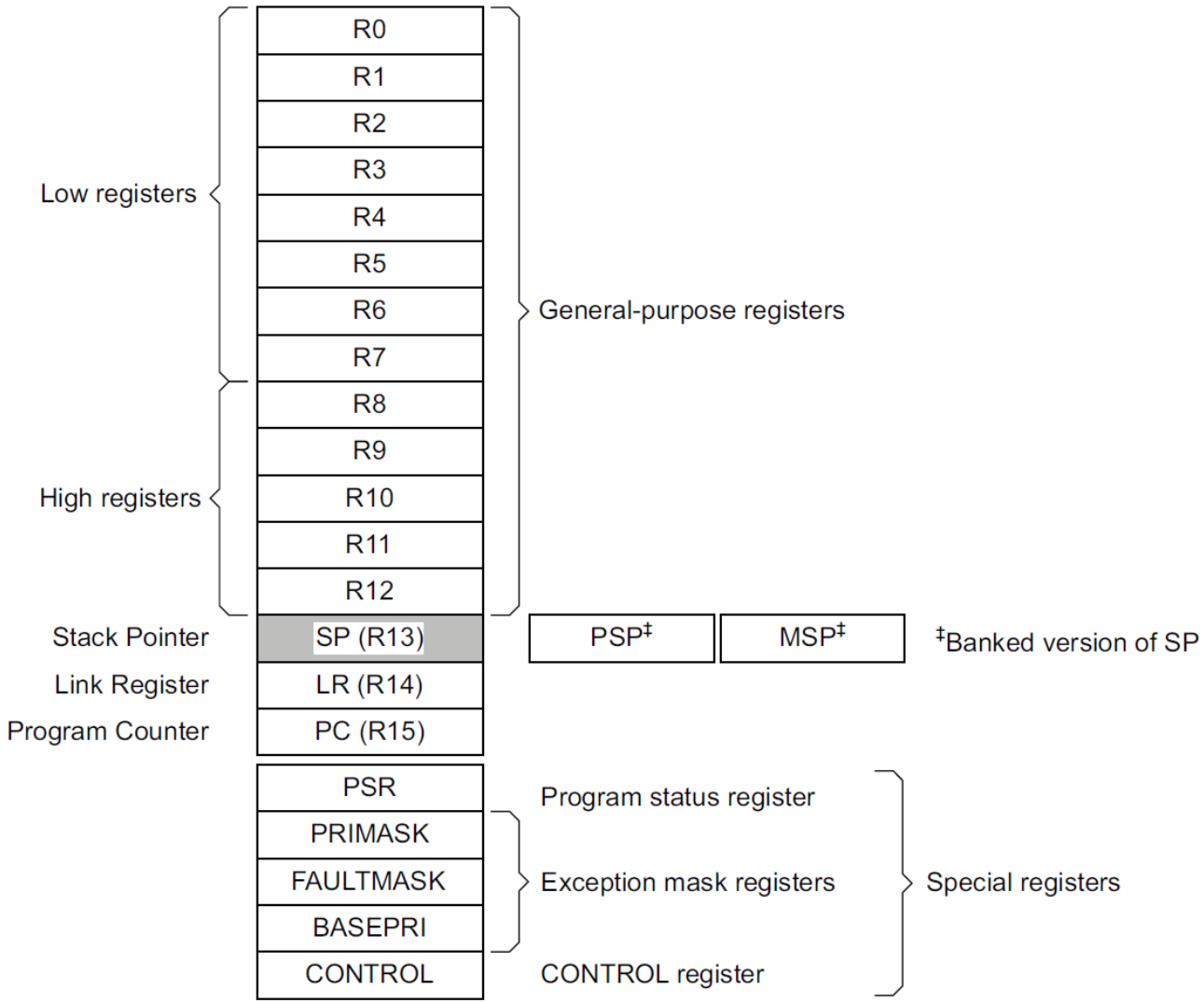
ARM Cortex architecture

```
ldr r0, =222
ldr r1, =111
add r2, r0, r1
```



SDU

# ARM Cortex-M4 core registers



# ARM Cortex-M4 data types

- supports the following data types:
  - 32-bit words
  - 16-bit halfwords
  - 8-bit bytes
- manages all data memory accesses as little-endian or big-endian. Instruction memory and Private Peripheral Bus (PPB) accesses are always performed as little-endian

# Interrupts



SDU

# Interrupts - reminder

- an interrupt is an input signal to the processor indicating an event that needs immediate attention.
- An interrupt signal alerts the processor and serves as a request for the processor to interrupt the currently executing code, so that the event can be processed in a timely manner.
- If the request is accepted, the processor responds by suspending its current activities, saving its state, and executing a function called an interrupt handler (or an interrupt service routine, ISR) to deal with the event.
- This interruption is temporary, and, unless the interrupt indicates a fatal error, the processor resumes normal activities after the interrupt handler finishes

# Interrupt vector table

| Vector number | Interrupt number | Description |
|---|---|---|
| 0-15 | - | Processor exceptions |
| 16 | 0 | GPIO Port A |
| 17 | 1 | GPIO Port B |
| .... | ..... | |
| 35 | 19 | Timer0 A |
| 36 | 20 | Timer0 B |
| .... | .... | |



SDU

# Counters/timers - reminder

# Counters/timers - reminder

- Counter/timer hardware is a crucial component of most embedded systems.
- In some cases a timer is needed to measure elapsed time; in others we want to count or time some external events
- When a timer reaches zero it can trigger an interrupt

# TIMER0 Initialization and configuration

```
// Enable timer peripheral clock
SYSCTL_RCGC1_R |= SYSCTL_RCGC1_TIMER0;
// Disable timer
TIMER0_CTL_R &= ~(TIMER_CTL_TAEN);
// Enable 32-bit configuration
TIMER0_CFG_R = 0x00000000;
// Set to Periodic Timer mode, first clear then set
TIMER0_TAMR_R &= ~(TIMER_TAMR_TAMR_M);
TIMER0_TAMR_R |= TIMER_TAMR_TAMR_PERIOD;
// Set Reload value. timeout = 1sek @ F_CPU = 16 Mhz.
TIMER0_TAILR_R = 16000000;
// Enable timer 0 interrupt
TIMER0_IMR_R |= TIMER_IMR_TATOIM;
```



SDU

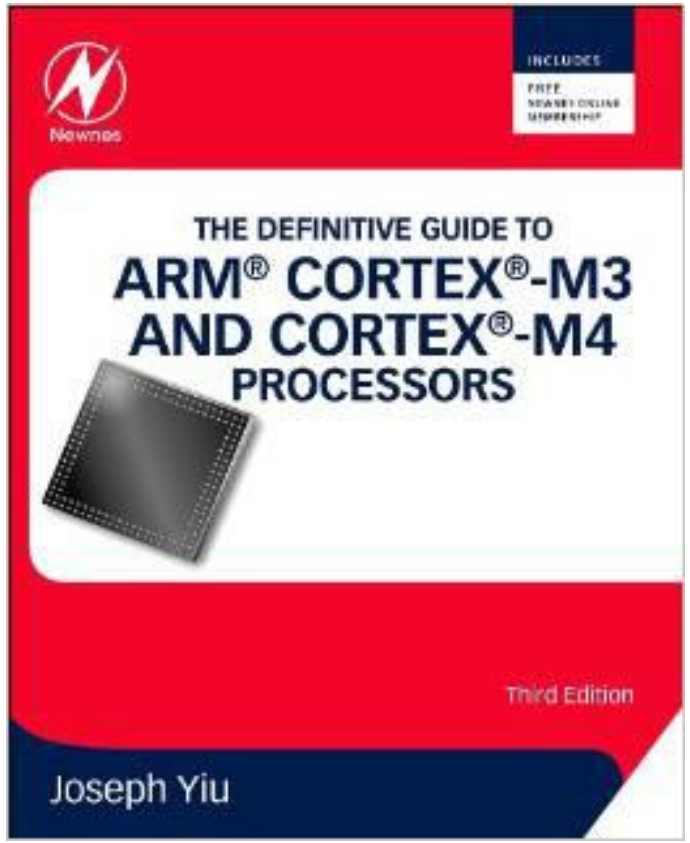
# NVIC Initialization and configuration

```
// Vector number 35, Interrupt Number = 19
// Clear pending interrupt
NVIC_UNPEND0_R |= NVIC_UNPEND0_INT19;
// Set Priority to 0x10, first clear then set.
NVIC_PRI4_R &= ~(NVIC_PRI4_INT19_M);
NVIC_PRI4_R |= (NVIC_PRI4_INT19_M & (0x10<<NVIC_PRI4_INT19_S));
// Enable NVIC interrupt
NVIC_EN0_R |= NVIC_EN0_INT19;
// Enable and start timer
TIMER0_CTL_R |= TIMER_CTL_TAEN;
```



SDU

# Additional literature






SDU