Introduction to ARM Processor and Tiva C-Series Microcontroller

ENCE361 Embedded Systems 1

Course Coordinator: Ciaran Moore (ciaran.moore@canterbury.ac.nz)

Lecturer: Le Yang (le.yang@canterbury.ac.nz)

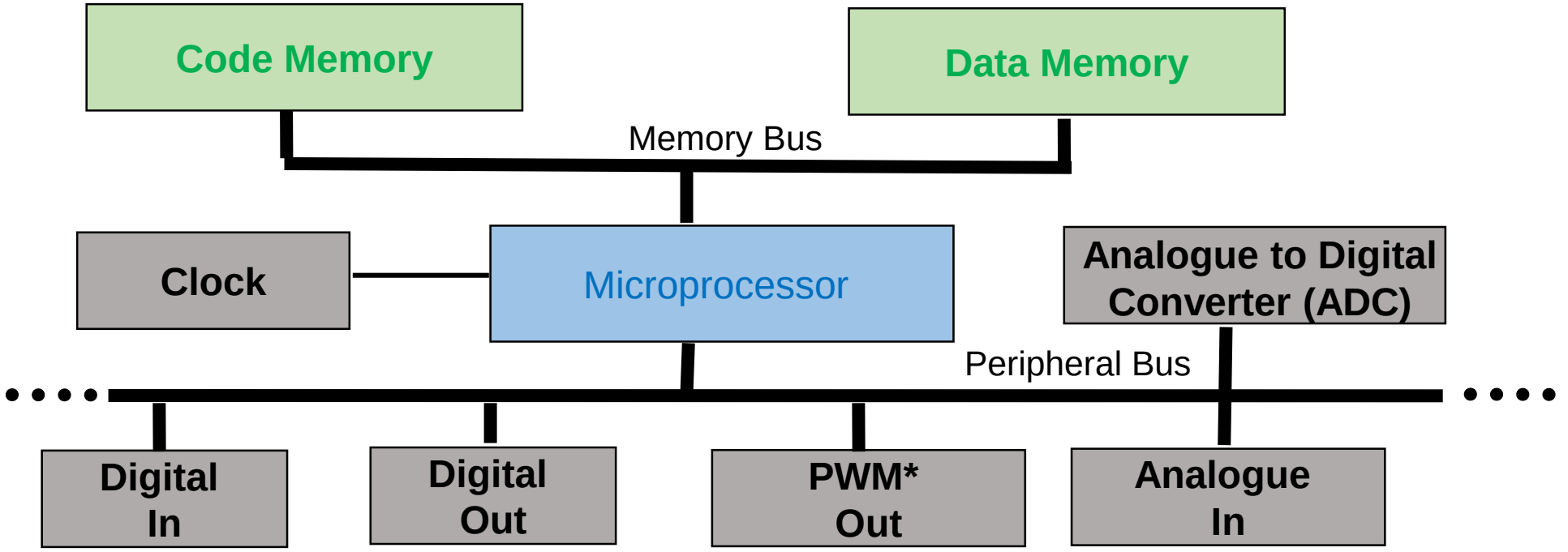
Department of Electrical and Computer Engineering

Where we're going today

- **Microcontroller and its application**
- ARM Cortex-M4 processor and Tiva C-series TM4C123x MCU
- ARM instruction set Not testable
- Example program in C

Microcontroller (MCU) vs. Microcomputer

- Microcontroller = Microprocessor (MPU) + Memory + Peripherals
- Microcomputer = microcontroller on a single silicon chip
  - Of which 99% are



* PWM: Pulse Width Modulation

Progress of MPU and MCU

MIPS: Million Instructions Per Second

Processor	Bus Width	Largest memory ROM/SRAM	Internal Peripherals	Speed (MIPS)	Type	Year of Manuf.
Intel 8051 family	8	64 KB program + 64 KB data	3 Timers + 1 serial port	1	MPU	1980
Freescale MC9S08QB8	8	8 KB Flash + 512 B data	3 Timers + ADC, SPI	20	MCU	2010
Intel 8086 family	16	32KB program + 512 B data	All external	5	MPU	1978
Texas Instr. MSP430x3	16	8 KB Flash + 512 B data	3 Timers + ADC, SPI	16	MCU	2000
Motorola 68000 family	32	4GB	Various	10	MPU	1978
Texas Instr. Tiva C-Series	32	4GB +32KB	Timer/PWM ADC, SPI, +	80	MCU	2015

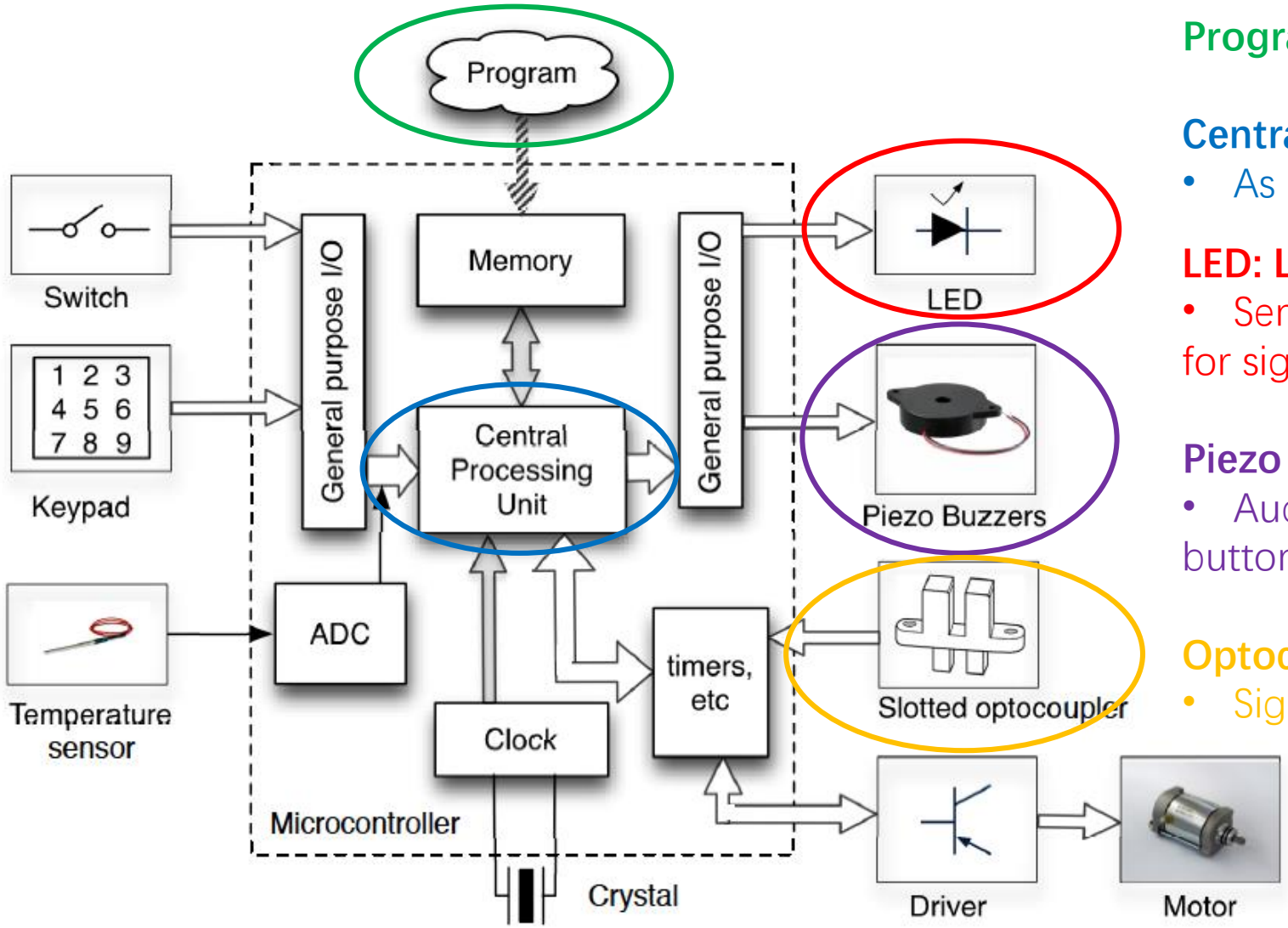
ROM: Read-Only Memory
- Store firmware

SRAM: Static Random-Access Memory
- CPU cache

SPI: Serial Peripheral Interface
- Master-slave synchronous short-distance communications

We are using it

Application of Microcontroller



Program: Code + Data

Central Processing Unit (CPU)

- As microprocessor (MPU)

LED: Lights Emitting Diode

- Semiconducter light source for signalling

Piezo buzzer

- Audio signalling device to indicate button pressing (click, ring or beep)

Optocoupler: light emitter + receiver

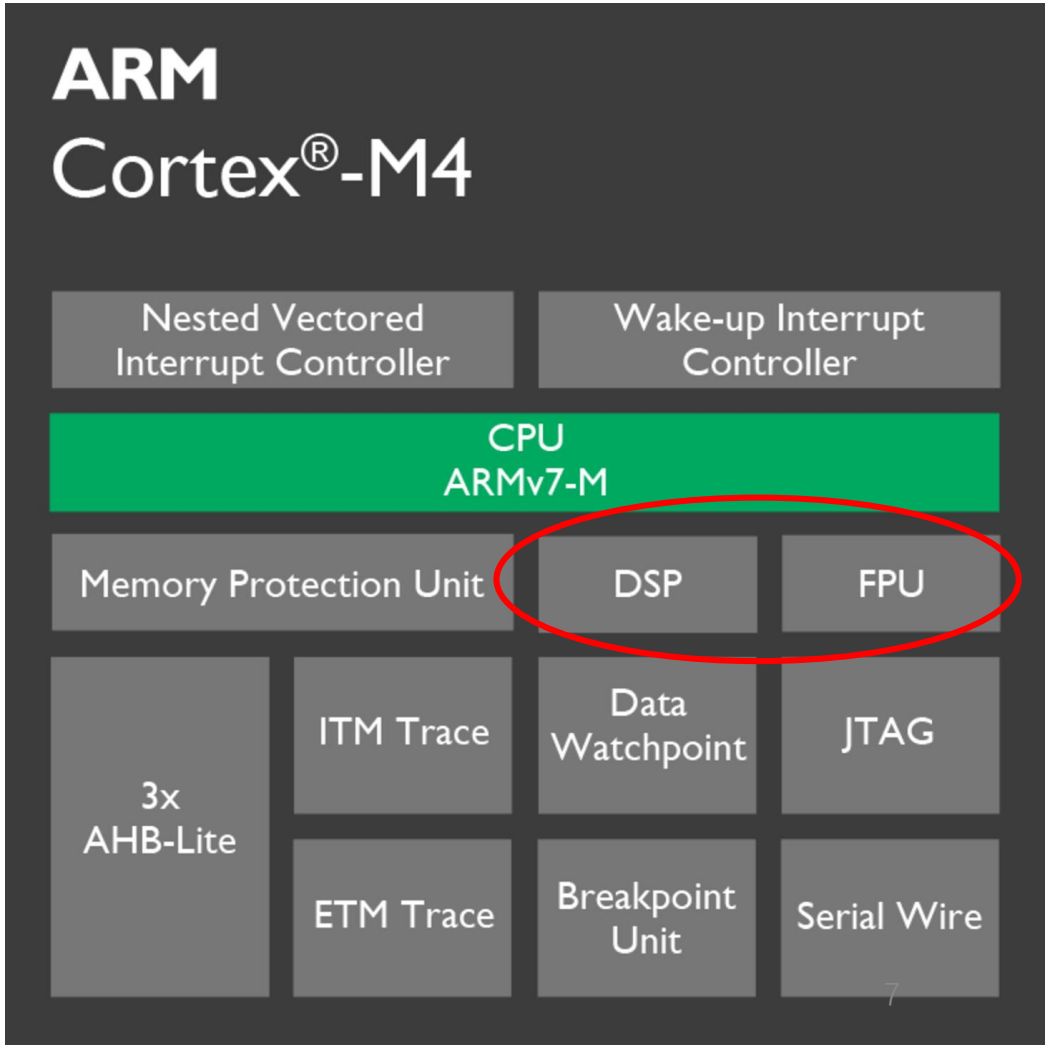
- Signal detection in electrical noise …

Where we're going today

- Microcontroller and its application
- **ARM Cortex-M4 processor and Tiva C-series TM4C123x MCU**
- ARM instructions
- Example program in C

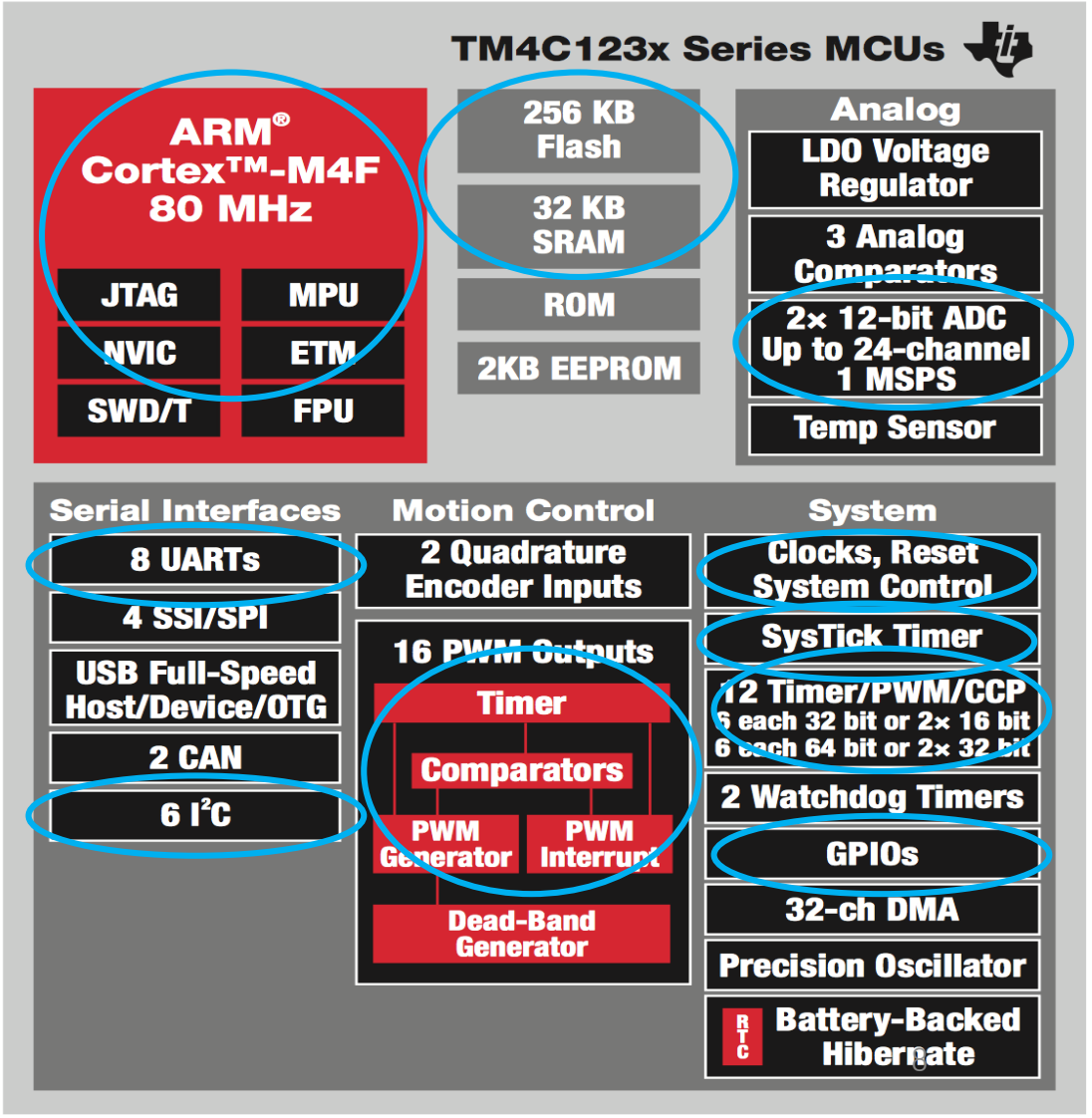
ARM Cortex-M4 Processor

- ARMv7-M CPU
  - Not a new CPU, also used in ARM Cortex-M3
- Digital Signal Processing (DSP) Support
- Float Point Unit (FPU)
- Faster float point calculations and more efficient in MIPS/mW than ARM Cortex-M3
  - MIPS: Million Instructions Per Second
  - mW: milliwatt



Tiva C-Series TM4C123x MCU

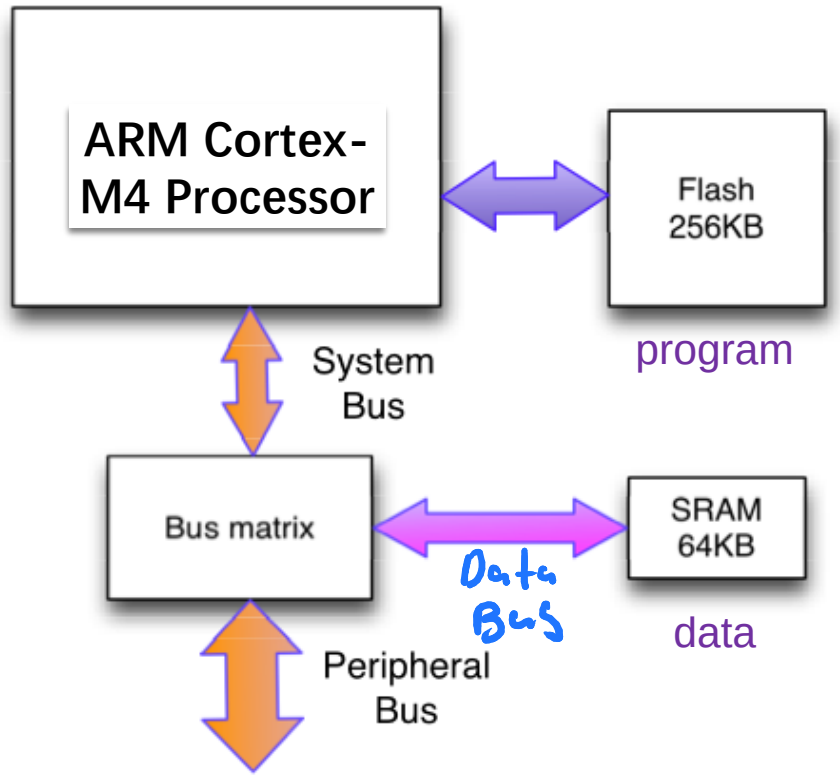
- Based on ARM Cortex-M4
- Circles indicate modules relevant to our course
- Homework: find out the definition of each acronym in the diagram



MCU Architecture

- Harvard structure
  - Separate storage and signal pathways for program and data
  - Read an instruction and access data memory at the same time

  - Mitigate von Neumann bottleneck
    - Von Neumann structure
      - shared storage and signal pathway for program and data
      - CPU needs to wait for data: pathway competition



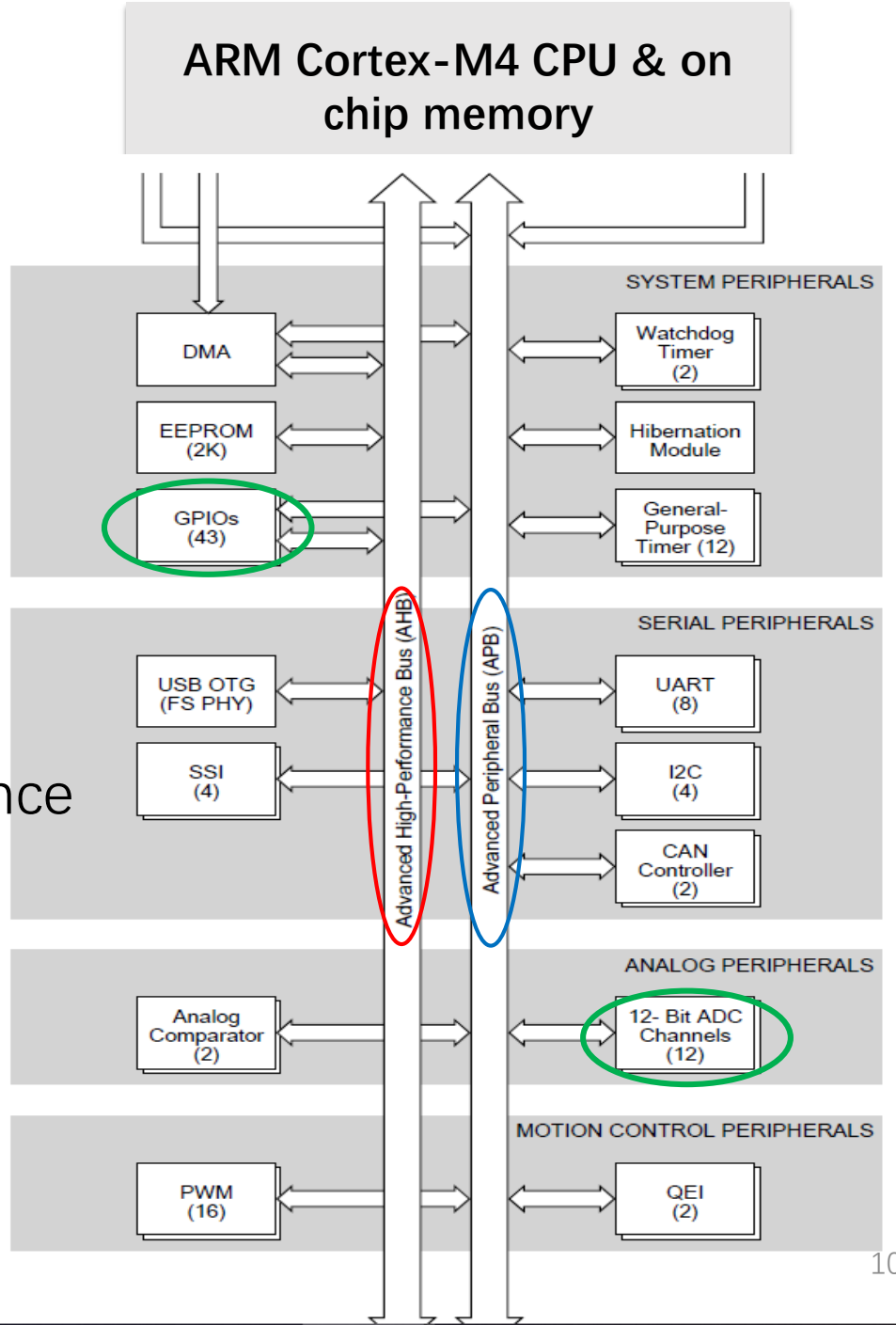
MCU Peripherals

From: Figure 1-1, P48

- Tiva TM4C123GH6PM Data Sheet

Peripheral buses

- AHB: Advanced High-Performance Bus
  - Better back-to-back access performance
- APB: Advanced Peripheral Bus
  - Legacy bus



Where we're going today

- Microcontroller and its application
- ARM Cortex-M4 processor and Tiva C-series TM4C123x MCU
- **ARM instruction set** ← Not testable
- Example program in C

ARM Instruction Set

- ARM: Advanced RISC Machine
  - First developed by Acorn Computers in the mid 80's based on RISC concept originated at Stanford & Berkeley
  - Now a separate company licensing ARM cores to Texas Instruments (TI), Atmel, Motorola …
- RISC vs. CISC
  - RISC: Reduced Instruction Set Computer
    - One instruction for one operation executed within one clock cycle
    - Larger code size but less complicated hardware (Advantage)
  - CISC: Complex Instruction Set Computer
    - One instruction for multiple operations run over several clock cycles
    - Shorter code length but complex hardware

ARM Instruction Set Architecture (ISA)

- Standard ARM instructions
  - Fixed length of 32 bits
  - Most are executed in a single clock cycle (RISC)
- ARM Cortex-M4 has Thumb-2 instruction set
  - ARM (32-bit instructions)/Thumb (16-bit instructions, introduced in 1994) blend
  - Decode Thumb instructions into 32-bit ARM instructions by hardware
- Load-Store architecture
  - Separate instructions for memory-register and between-register operations
- Instructions can optionally set condition flags (suffix 'S')
- Conditional execution of instructions

Example of ARM Instructions

Table 3-1 Cortex-M4 instructions

Mnemonic	Operands	Brief description	Flags	Page
ADC, ADCS	{Rd,} Rn, Op2	Add with Carry	N,Z,C,V	page 3-41
ADD, ADDS	{Rd,} Rn, Op2	Add	N,Z,C,V	page 3-41
ADD, ADDW	{Rd,} Rn, #imm12	Add	-	page 3-41
ADR	Rd, label	Load PC-relative Address	-	page 3-23
AND, ANDS	{Rd,} Rn, Op2	Logical AND	N,Z,C	page 3-44
ASR, ASRS	Rd, Rm, <Rs|#n>	Arithmetic Shift Right	N,Z,C	page 3-46
B	label	Branch	-	page 3-119
BFC	Rd, #lsb, #width	Bit Field Clear	-	page 3-115
BFI	Rd, Rn, #lsb, #width	Bit Field Insert	-	page 3-115
BIC, BICS	{Rd,} Rn, Op2	Bit Clear	N,Z,C	page 3-44
BKPT	#imm	Breakpoint	-	page 3-158
BL	label	Branch with Link	-	page 3-119
BLX	Rm	Branch indirect with Link	-	page 3-119
BX	Rm	Branch indirect	-	page 3-119

From Page 3-2 Cortex-M4 Devices_Generic Users Guide.pdf

Conditional Execution of Instructions

- Condition flags in Application Program Status Register (APSR)

Table 2-4 APSR bit assignments

Bits	Name	Function
[31]	N	Negative flag
[30]	Z	Zero flag
[29]	C	Carry or borrow flag
[28]	V	Overflow flag
[27]	Q	DSP overflow and saturation flag
[26:20]	-	Reserved
[19:16]	GE[3:0]	Greater than or Equal flags. See *SEL* on page 3-70 for more information.
[15:0]	-	Reserved

From Page 2-5 Cortex-M4 Devices_Generic Users Guide.pdf

- Relationship between condition code suffix *{cond}* and flags

Table 1-2. Condition Code Suffixes

Suffix	Flags	Meaning
EQ	Z = 1	Equal
NE	Z = 0	Not equal
CS or HS	C = 1	Higher or same, unsigned ≥
CC or LO	C = 0	Lower, unsigned <
MI	N = 1	Negative
PL	N = 0	Positive or zero
VS	V = 1	Overflow
VC	V = 0	No overflow
HI	C = 1 and Z = 0	Higher, unsigned >
LS	C = 0 or Z = 1	Lower or same, unsigned ≤
GE	N = V	Greater than or equal, signed ≥
LT	N != V	Less than, signed <
GT	Z = 0 and N = V	Greater than, signed >
LE	Z = 1 and N != V	Less than or equal, signed ≤
AL	Can have any value	Always. This is the default when no suffix is specified.

Conditional Execution of Instructions

- BEQ label ; Branch to label if previous operation results in
  ; equal status (Z=1)
- ADDEQ R0, R1, R2 ; Carry out the add operation if previous
  ; operation results in equal status (Z=1)
- Condition execution requires a preceding IT instruction
- Homework: What does the following code do? (see also slide 22)

```
CMP R0, #9;
  ITE GT;
  ADDGT R1, R0, #55; #65
  ADDLE R1, R0, #48;
```

Where we're going today

- Microcontroller and its application
- ARM Cortex-M4 processor and Tiva C-series TM4C123x MCU
- ARM instruction set
- **Example program in C**

C Program to Run on Tiva (1 of 3)

```
//*****************************************************
//
// ADCdemo1.c - Simple interrupt driven program that
// samples an analogue voltage with AIN0
//
// Author:  P. J. Bones, UC ECE
// Last modified:         18/02/2020 by Le Yang, UC ECE
//
//*****************************************************

#include <stdint.h>
#include <stdbool.h>

#include "driverlib/adc.h"
#include "driverlib/gpio.h"
#include "driverlib/sysctl.h"
#include "driverlib/systick.h"
#include "driverlib/interrupt.h“

#define BUF_SIZE 10
#define SAMPLE_RATE_HZ 10
```

Data type definition, uint8_t, uint32_t, int32_t … as in ENCE 260

More #include to access API, *TivaWare*, …

Definition of constants

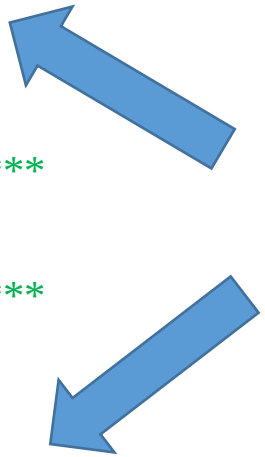
C Program to Run on Tiva (2 of 3)

```
//*****************************************************************
// Handler for SysTick interrupt
//*****************************************************************
void SysTickIntHandler(void)
{
  ADCProcessorTrigger(ADC0_BASE, 3);    // Start a conversion
}

//*****************************************************************
// Handler for ADC conversion complete interrupt
//*****************************************************************
void ADCIntHandler(void)
{
        unsigned long ulValue;

        ADCSequenceDataGet(ADC0_BASE, 3, &ulValue);  // Get sample
        writeCircBuf (&g_inBuffer, (uint32_t) ulValue);       // Store it

        // Clean up by clearing the interrupt
        ADCIntClear(ADC0_BASE, 3);
}
```



Both functions are interrupt handlers

C Program to Run on Tiva (3 of 3)

```
Int main(void)
{
    uint16_t i;
    int32_t sum;

    initClock ();
    initADC ();
    initDisplay ();
    initCircBuf (&g_inBuffer, BUF_SIZE);

    IntMasterEnable();                    // Enable interrupts to the processor
    while (1)
    {
        // Background task: calculate and display approximate mean value
        for (sum = 0, i = 0; i < BUF_SIZE; i++)
            sum = sum + readCircBuf (&g_inBuffer);

        displayMeanVal (sum / BUF_SIZE);          // Display mean value
        SysCtlDelay (SysCtlClockGet() / 6);       // Update at ~ 2 Hz
    }
}
```

Variables definitions in the main program

API calls to set up peripherals and clocks

Fundamental Data Types

Type Class	Machine Type	Byte size	Byte alignment	Note
Integral	Unsigned byte	1	1	Character
	Signed byte	1	1	
	Unsigned half-word	2	2	
	Signed half-word	2	2	
	Unsigned word	4	4	
	Signed word	4	4	
	Unsigned double-word	8	8	
	Signed double-word	8	8	
Floating Point	Single precision (IEEE 754)	4	4	The encoding of floating point numbers is described in [ARM ARM] chapter C2, *VFP Programmer's Model*, *§2.1.1 Single-precision format*, and *§2.1.2 Double-precision format*.
	Double precision (IEEE 754)	8	8	
Pointer	Data pointer	4	4	Pointer arithmetic should be unsigned.
	Code pointer	4	4	Bit 0 of a code pointer indicates the target instruction set type (0 ARM, 1 Thumb).

Table 1, Byte size and byte alignment of fundamental data types

Example of ARM Instructions

ADD, ADC, SUB, SBC, and RSB

Add, Add with carry, Subtract, Subtract with carry, and Reverse Subtract.

Syntax

```
op{S}{cond} {Rd,} Rn, Operand2
op{cond} {Rd,} Rn, #imm12                ; ADD and SUB only
```

where:

op Is one of:

ADD	Add.
ADC	Add with Carry.
SUB	Subtract.
SBC	Subtract with Carry.
RSB	Reverse Subtract.

S	Is an optional suffix. If S is specified, the condition code flags are updated on the result of the operation, see *Conditional execution* on page 3-18.
cond	Is an optional condition code, see *Conditional execution* on page 3-18.
Rd	Specifies the destination register. If *Rd* is omitted, the destination register is *Rn*.
Rn	Specifies the register holding the first operand.
Operand2	Is a flexible second operand. See *Flexible second operand* on page 3-12 for details of the options.
imm12	Is any value in the range 0-4095.

Operation

The ADD instruction adds the value of *Operand2* or *imm12* to the value in *Rn*.

The ADC instruction adds the values in *Rn* and *Operand2*, together with the carry flag.

The SUB instruction subtracts the value of *Operand2* or *imm12* from the value in *Rn*.

The SBC instruction subtracts the value of *Operand2* from the value in *Rn*. If the carry flag is clear, the result is reduced by one.

The RSB instruction subtracts the value in *Rn* from the value of *Operand2*. This is useful because of the wide range of options for *Operand2*.

From Page 3-41 Cortex-M4 Devices_Generic Users Guide.pdf