嵌入式微處理機
Embedded Microprocessors

Text Book: TI's Teaching ROMs

References: MSP 430 family data sheets and user's guides

http://www.ti.com/ww/eu/university/roms.html

Info

- **Time：Mon. 6, 7, 8 (EE 501)**
- **Evaluation：**
  - Lab Experiments 50%
  - Midterm (Principle) 20%
  - Final (Operation) 30%
- **Website：**
  http://ares.ee.nchu.edu.tw/course/epr102/

Outline

- Introductory Overview
- MSP430 Architecture
- Device Systems and Operating Modes
- General purpose Input/Output
- Timers
- Data Acquisition
- Digital-to-Analogue Converter
- Direct Memory Access (DMA)
- Hardware Multiplier
- Flash Programming
- Communications

>> Contents

Lecture 1
Introductory Overview

>> Contents

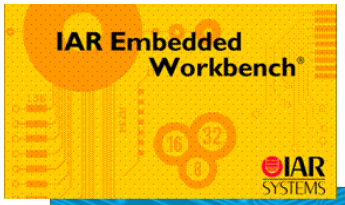
Contents

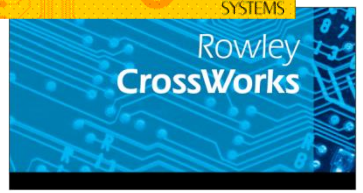
MSP430 IDEs (1/2)

- **Main characteristics of the MSP430 Integrated Development Environments (IDEs);**
- **Available both from TI and third parties;**
- **Special attention will be given to:**
  - Code Composer Essentials v3;
  - IAR Embedded Workbench (EWB) IDE.
- **Using an IDE:**
  - Basic functions;
  - Step by step project development;
    - Structure and management (source files; compiling, assembling and linking operations) of projects developed both in C and/or Assembly language.

MSP430 IDEs (2/2)

- **A range of software tools are available for the generation of MSP430 source code:**





  - **Code Composer Essentials (TI)**
  - **IAR EWB - Kickstart ed. (IAR Systems);**
  - CrossStudio (Rowley Associates);
  - MSPGCC (*open-source comunity)*...
  - SwiftX (Forth, Inc.);
  - HI-TECH (HI-TECH software);
  - ANSI C (ImageCraft);
  - Project-430 (Phyton, Inc.);
  - AQ430 (Quadravox);
  - …

Code Composer Essentials v3 (1/2)

- **TI's MSP430 IDE:**

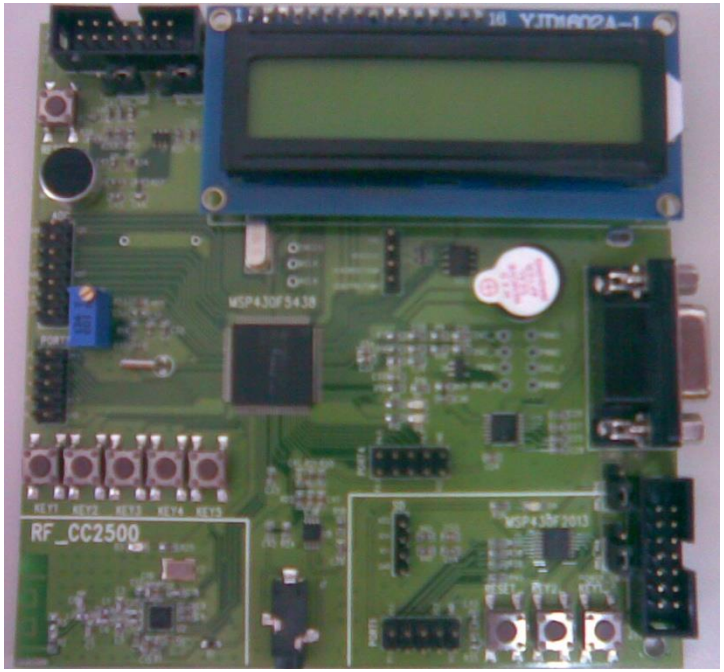
- **It is available as:**
  - A free upgrade for existing v2 users;
  - Professional version ($499), the main features being:
    - Unlimited code size;
    - Can be ordered from the MSP430 web page;
    - Supported by TI Software Support.
  - Evaluation Version (Free):
    - 16 kB Limit on C/ASM code size;
    - Download from MSP430 web page;
    - Supported by TI Software Support.

Code Composer Essentials v3 (2/2)

- **New features include:**
  - Free 16 kB code-limited version;
  - Support for large memory model (Place data >64k);
  - Enhanced Compatibility with IAR C-code:
    `#pragma` (ISR declarations), most intrinsics;
  - GDB Debugger replaced by TI proprietary debugger that allows faster single stepping;
  - Hardware Multiplier libraries (16 bits and 32 bits);
  - CCE v2 project support (auto convert);
  - Breakpoints:
    - EEM support via unified breakpoint manager;
    - Use of Enhanced Emulation Module (EEM), with predefined Use Cases;
    - Unlimited Breakpoints.

Starter kit: Experimenter's board

- **Features:**
  - MSP430F2013;
  - MSP430F5438;
  - Compatible with TI's wireless evaluation modules.



- **Combining 2 MCUs provides nearly every MSP430 peripherals available.**

- **Needs a MSP-FET430 for development**

Starter kit: eZ430-F2013

- **All 14 input/output pins on the MSP430F2013 are accessible on the MSP-EZ430D target board for easy debugging and interfacing to peripherals;**

- **One of these input/output pins is connected to an LED for visual feedback;**

- **Device features and integrated peripherals:**
  - 16-MIPS performance;
  - 16-bit Sigma Delta ADC;
  - 16-bit timer;
  - Watchdog timer;
  - Brownout detector;
  - USI module supporting SPI and I2C;
  - 5 low power modes (0.5 μA standby).

Starter kit: MSP-EXP430F5438 (1/5)

- **MSP-EXP430F5438:**
  - MSP430F5438.
- **New features:**
  - Power Management Module (PMM);
  - Unified Clock System (UCS);
  - System (SYS) modules;
  - Expanded memory/peripheral mapping
  - Peripheral module enhancements.
- **Enhanced performance**
  - 20 bit address capability;
  - 32 bit Hardware Multiplier.
- **This board provides the wide range of F5438 peripherals.**

Starter kit: MSP-EXP430F5438 (2/5)

- **Device features and integrated peripherals**
  - Device: MSP430F5438:
    - 256 kB + 512 kB flash memory; 16 kB RAM.
  - Integrated peripherals:
    - Three 16-bit timers;
    - 12-bit SAR Analogue-to-Digital Converter;
    - Direct Memory Access (DMA);
    - Hardware multiplier (supporting 32-Bit operations);
    - Universal Serial Communication Interfaces (USCI): Enhanced UART Supporting Auto-Baudrate; IrDA Encoder and Decoder; Synchronous SPI; I2C™;
    - Real time clock module with alarm capabilities;
    - Temperature sensor;
    - Up to 87 I/O pins.

Starter kit: MSP-EXP430F5438 (3/5)

- **The MSP430F5438 supports I2C and SPI protocols using the USCI and the USI peripherals;**
  - This protocol is used for inter-processor communication;
  - The link can be disconnected in hardware allowing these peripherals to be used for other communication purposes.

- **Programming and Debugging:**
  - Can be programmed using any MSP430 Flash Emulation Tool (MSP-FET430xIF);

- **Wireless expansion:**
  - Compatible with TI Wireless CCxxxXEMK Evaluation Modules, such as the CC2500EMK.
  - Compatible with TI eZ-RF2500.

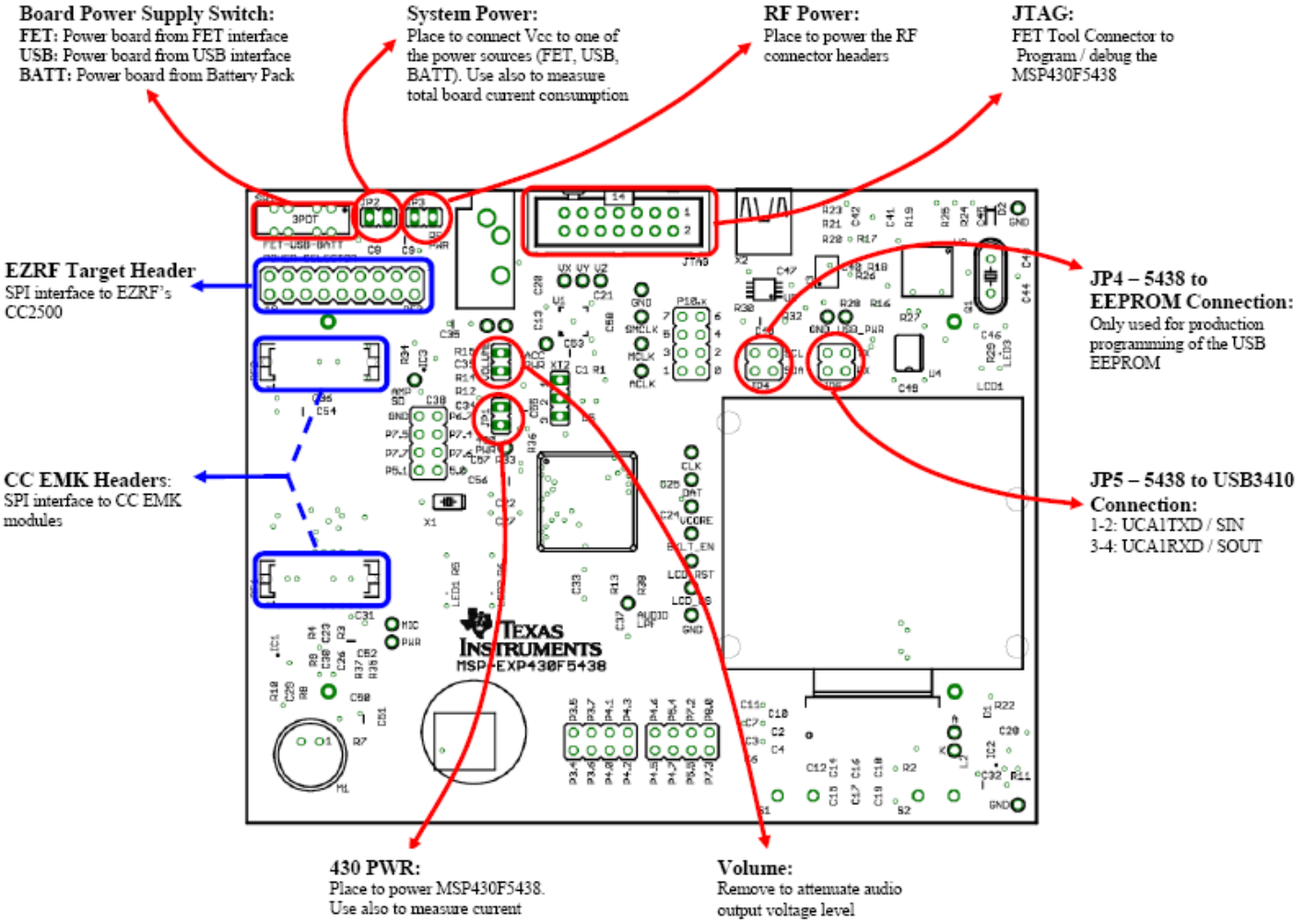
Starter kit: MSP-EXP430F5438 (4/5)

- **The demo board has various system clock options that support low and high frequencies.**

- **The MSP430F5438 has integrated an Unified Clock System that provides different clock sources:**
  - Three low-frequency sources:
    - LFXT1;
    - Internal Very Low Power/Low Frequency Oscillator (VLO);
    - Internal Reference Oscillator (REFO).
  - Internal Digitally Controlled Oscillator (DCO) / Frequency Locked Loop (FLL) for highspeed operation:
    - FLL reference selectable from LFXT1, REFO, or XT2.
  - ACLK/SMCLK/MCLK can all be driven from any source;
  - Dedicated MODOSC (internal) used for modules like Flash controller, ADC, among others.

Starter kit: MSP-EXP430F5438 (5/5)

- **MSP-EXP430F5438 demo board jumper and connectors locations:**



Wireless expansion (Chipcon's RF transceiver chip)

- **CC2500EMK Evaluation Module 2.4 GHz:**
  - The CC2500EM evaluation modules are provided with antennas;
  - These evaluation modules are add-on daughter boards that require a CC2500 development kit for evaluation and development;
  - It allows to do range testing (PER testing) and transfer data from one PC to another using the SmartRF®04DK, in order to evaluate how well the SmartRF®04 products fit the intended application;
  - It allows performing RF measurements.

MSP-FET430 Flash Emulation Tool (1/2)

- **The flash emulation tool (FET) allow the application development on the MSP430 MCU;**

- **There are available two debugging interfaces:**
  - USB port: MSP-FET430UIF;
  - Parallel port: MSP-FET430PIF.

- **MSP-FET430UIF flash emulation tool:**

MSP-FET430 Flash Emulation Tool (2/2)

- **Are used to program and debug the MSP430 in-system through the:**
  - 4-wire JTAG interface: MSP-FET430PIF and MSP-FET430UIF;
  - 2-wire JTAG interface (Spy Bi-Wire): MSP-FET430UIF.

- **These debugging tools interface the previously presented MSP430 hardware development tools to the included integrated software environment (CCE or IAR) and include code to start an application.**

- **Both MSP-FET430 supports development with all MSP430 flash devices.**

How to Read Datasheets (1/6)

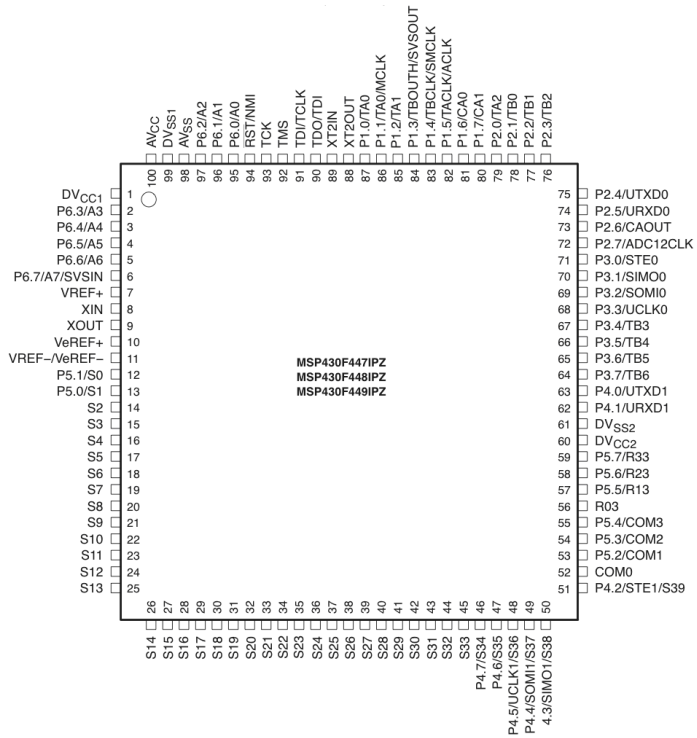
- **Manufacturers of electronic components provide datasheets containing the specifications detailing the part/device characteristics;**
- **Datasheets give the electrical characteristics of the device and the pin-out functions, but without detailing the internal operation;**
- **More complex devices are provided with documents that aid the development of applications, such as:**
  - Application notes;
  - User's guides;
  - Designer's guides;
  - Package drawings, etc…

How to Read Datasheets (2/6)

- **Datasheets include information:**
  - Concerning the part number of the device or device series;
  - Electrical characteristics:
    - Maximum and minimum operating voltages;
    - Operating temperature range e.g. 0 to 70 degrees C;
    - Output drive capacity for each output pin, as well as an overall limit for the entire chip;
    - Clock frequencies;
    - Pin out electrical characteristics (capacitance, inductance, resistance);
    - Noise tolerance or the noise generated by the device itself;
    - Physical tolerances…

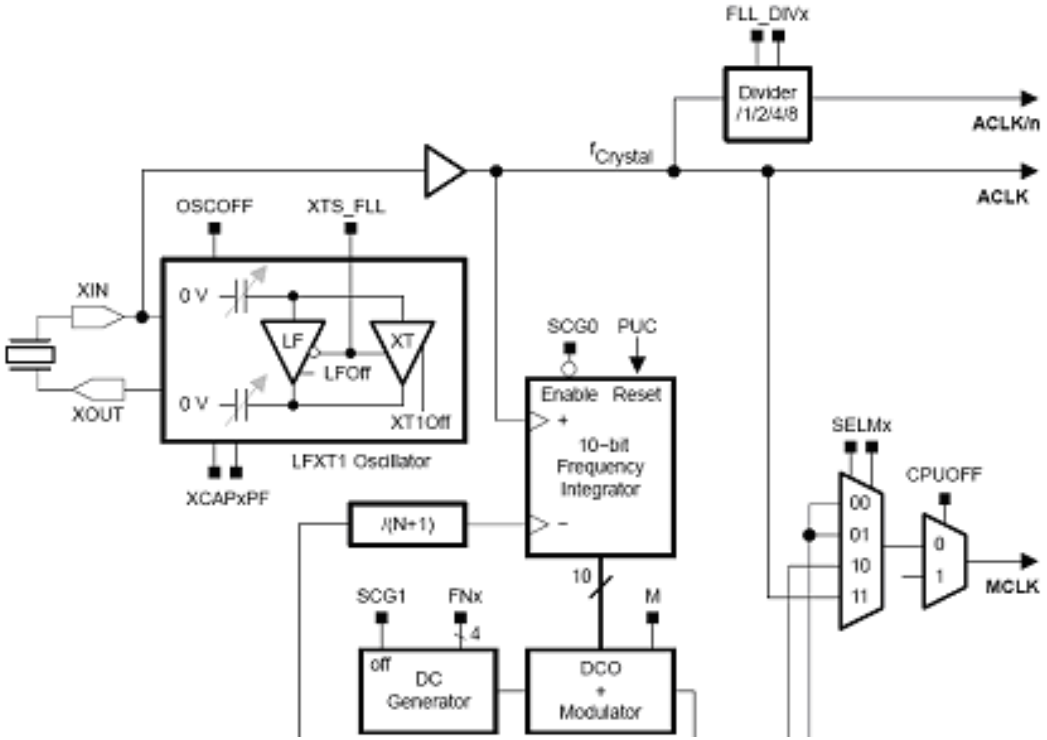
How to Read Datasheets (3/6)

- **MSP430 device datasheet:**
  - Device has a large number of peripherals;
  - Each input/output pin usually has more than one function;
  - It has a table with the description of each pin function;
  - Example, Pin number 2 = P6.3/A3;
    - Digital Input/Output Port 6 bit 3;
    - 3rd analogue input.



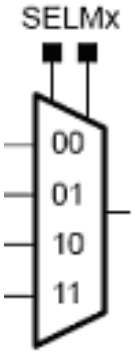
How to Read Datasheets (4/6)

- **MSP430 User's Guide:**
  - Most peripherals are represented by Block Diagrams.
  - Example: Part of the MSP430F44x clock module block diagram:



How to Read Datasheets (5/6)

- **MSP430 User's Guide:**
  - Example: Part of the MSP430F44x clock module block diagram: Detail SELMx;



  - Multiplexed block: (4 inputs and 1 output):
    - SELMx = 00: Output routed to the 1st multiplexer output;
    - SELMx = 01: Output routed to the second one and so on;
    - SELMx is a 2-bit *mnemonic*: SELM1 (MSB), SELM0 (LSB).
  - To use the peripheral, it is necessary to find out how the register(s) need to be configured:
    - SELMx is in the FLL_CTL1 register.

How to Read Datasheets (6/6)

❑ MSP430 User's Guide:

- SELMx are the 3rd and 4th bits of FLL_CTL1 control register.

FLL_CTL1, FLL+ control register 1

7	6	5	4	3	2	1	0
-	SMCLKOFF	XT2OFF	SELMx		SELS	FLL_DIVx	

Bit		Description
6	SMCLKOFF	Disable the submain clock signal (SMCLK): SMCLKOFF = 0 ⇒ SMCLK active SMCLKOFF = 1 ⇒ SMCLK inactive
5	XT2OFF	Disable the second crystal oscillator (XT2): XT2OFF = 0 ⇒ XT2 active XT2OFF = 1 ⇒ XT2 inactive
4-3	SELMx	Select the master clock (MCLK) source: SELM1 SELM0 = 0 0 ⇒ DCO SELM1 SELM0 = 0 1 ⇒ DCO SELM1 SELM0 = 1 0 ⇒ XT2 SELM1 SELM0 = 1 1 ⇒ LFXT1
2	SELS	Select the submain clock (SMCLK) source: SELS = 0 ⇒ DCO SELS = 1 ⇒ XT2
1-0	FLL_DIVx	Select the auxiliary clock (ACLK) signal divider: FLL_DIV_0 = 0 0 ⇒ Divider factor: /1 FLL_DIV_1 = 0 1 ⇒ Divider factor: /2 FLL_DIV_2 = 1 0 ⇒ Divider factor: /4 FLL_DIV_3 = 1 1 ⇒ Divider factor: /8

Good software practices for low power consumption (1/2)

- **C coding tips:**
  - Use local variable as much as possible (Local variables use CPU registers whereas global variables use RAM);
  - Use unsigned data types where possible;
  - Use pointers to access structures and unions;
  - Use “static const” class to avoid run-time copying of structures, unions, and arrays;
  - Avoid modulo;
  - Count down “for” loops.

Good software practices for low power consumption (2/2)

- **Principles for low power applications:**
  - Maximize the time in standby;
  - Use interrupts to control program flow;
  - Replace software functions with peripheral hardware;
  - Manage the power of internal peripherals;
  - Manage the power of external devices;
  - Device choice can make a difference;
  - Effective code is a must. Every unnecessary instruction executed is a portion of the battery wasted that will never return.

Analogue and Digital Signals (1/5)

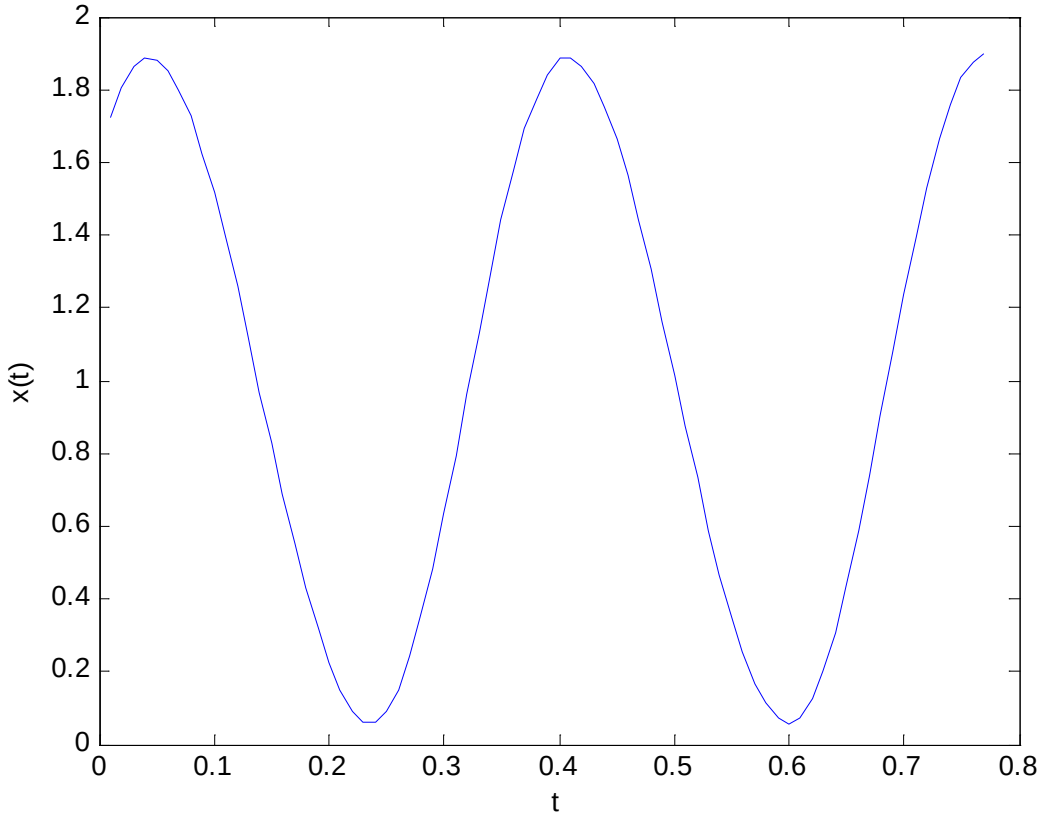
- **In nature, any measurable quantity in time or in space can be considered a signal;**
  - Example: The velocity of a body, as function of time or as function of position, can be represented by a signal.

- **Analogue signal can:**
  - Represent every value possible;
  - Swing through a "continuum" of values;
  - Provide information at every instant of time.

- **A digital signal assumes only a finite number of values at only specific points in time.**

Analogue and Digital Signals (2/5)

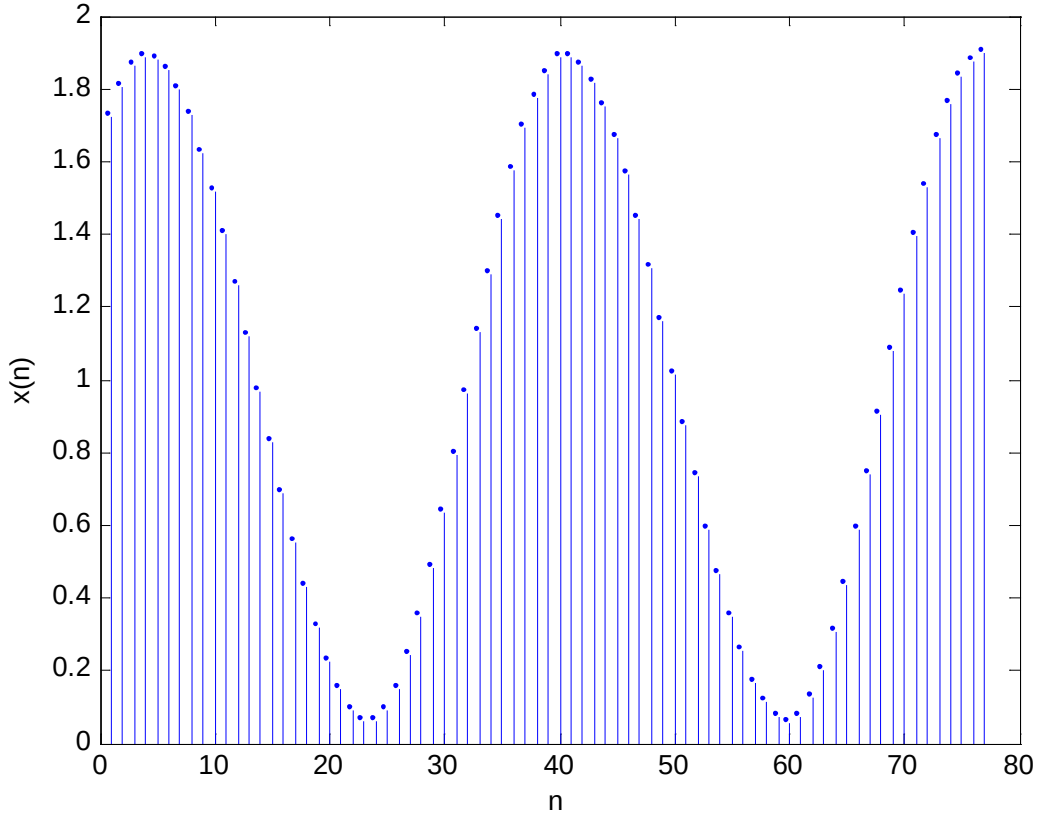
- **Example: analogue signal:**



>> Contents

Analogue and Digital Signals (3/5)

- **Example: discrete analogue signal:**
  - This analogue signal can be discretized in time, using a sampling period, *T*:



Analogue and Digital Signals (4/5)

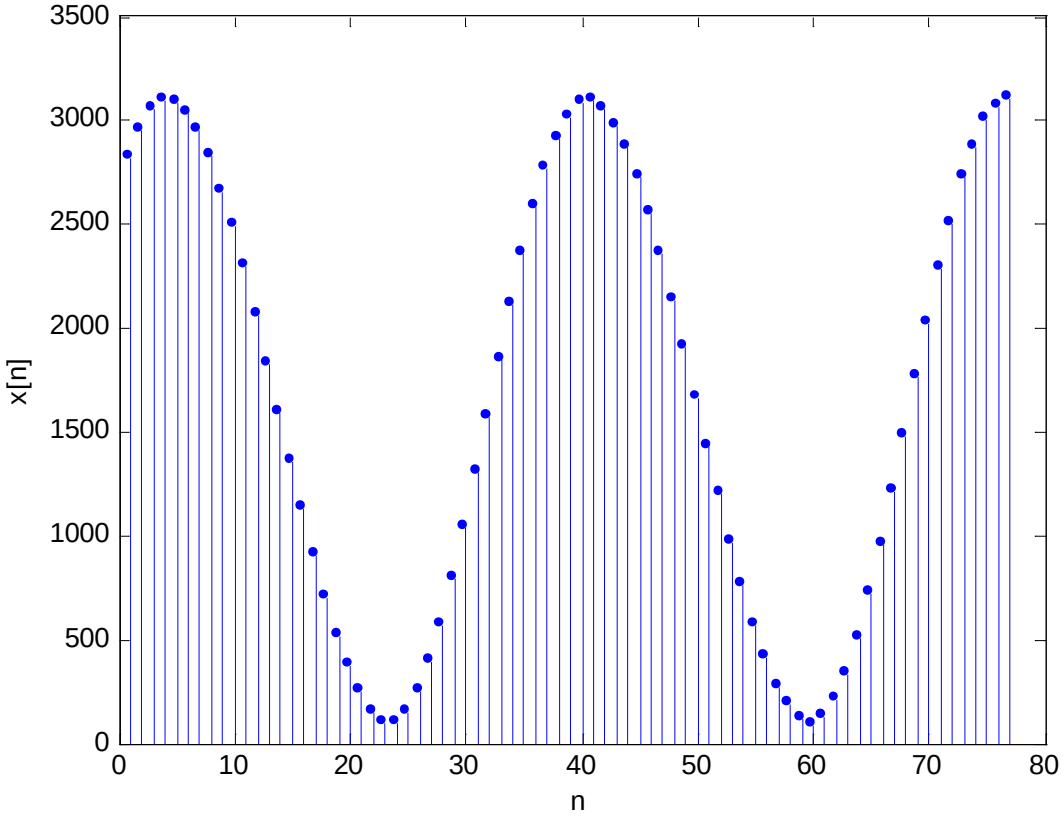
- **The information contained in the signal only can be used by a computer if an amplitude conversion from the analogue domain to the digital domains is performed;**

- **Digital signal:**
  - Rounds off all values to a certain precision or a certain number of digits.

- **Digital signal advantage:**
  - Easiness of data storage and manipulation.

Analogue and Digital Signals (5/5)

- **Example: quantized digital signal:**
  - Result of the analogue signal conversion, performed by a 12-bit Analogue-to-Digital Converter (ADC) with 2.5 V full-scale:



>> Contents

Mathematical Notations (1/16)

- **A base number is a notation that through symbols represents a consistent numerical value.**

- **Depending on the numerical basis, the representation 11 may correspond to:**
  - Eleven for the base decimal (or radix decimal);
  - Three for the base binary;
  - Or some other number, depending on the base used.

Mathematical Notations (2/16)

Decimal numerical base representation review:

$$1234_{10} = 1x10^3 + 2x10^2 + 3x10^1 + 4x10^0$$

- Each of the symbols is multiplied by a weight of base 10;
- As it moves from right to left the weight is increased.

General form for a number in base *b*:

$$a_{n-1}b^{n-1} + a_{n-2}b^{n-2} + \ldots + a_0b^0$$

- *n* is the length of the numerical representation;
- $\{a_{n-1}, a_{n-2}, \ldots, a_0\}$ is the numerical representation ordered increasingly (natural numbers of the set $\{0, \ldots, b-1\}$).

Mathematical Notations (3/16)

Binary Number Base System:

- Computers can only take two states, so they use the base binary (base 2);
- The states allowed are “open” or “closed”, “high” or “low”, “1” or “0 ”;
- Example: 010100_2

$$= 0x2^5 + 1x2^4 + 0x2^3 + 1x2^2 + 0x2^1 + 0x2^0 = 20_{10}$$

- Representation:

Position	5	4	3	2	1	0
Weight	2^5	2^4	2^3	2^2	2^1	2^0
	32	16	8	4	2	1
Base 2	0	1	0	1	0	0

Mathematical Notations (4/16)

- **Hexadecimal Number Base System:**
  - A difficulty that arises from the binary system is that it is verbose;
  - To represent the value 202_{10} in base 2 (binary) requires eight binary digits;
  - The decimal version requires only three decimal digits and so represents numbers much more compactly;
  - The hexadecimal (base 16) numbering system solves these problems.
    - Hexadecimal numbers are very compact;
    - It is easy to convert from hexadecimal to binary and binary to hexadecimal.

Mathematical Notations (5/16)

- **Hexadecimal Number Base System:**
  - Set: {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F};
  - Representation comes directly from the binary representation.
  - Example: 010100_2
    - Split the value 010100_2 in groups of 4 bits from right to left.
    - Pad out the short group with zeros at the front of the second group:

      $0001\ 0100_2 = 0x14$ or (14_{16})
  - Example: 0x0ABCD
    - Convert each hexadecimal digit to its binary counterpart:

      $= 0000\ 1010\ 1011\ 1100\ 1101_2$

Mathematical Notations (6/16)

- **Decimal to Binary Conversion:**
  - **Example 1**: Decimal number 721_{10} to binary conversion:

$721_{10} = 700 + 20 + 1 = 7x10^2 + 2x10^1 + 1x10^0$

$2^{10} = 1024_{10}$ (Higher than 3rd power decimal value, 10^3);
$2^9 = 512_{10}$ (MSB: $1x2^9$): $721 - 512 = 209$;
$2^8 = 256_{10}$ (Higher than 209, the 2nd bit: $0x2^8$);
$2^7 = 128_{10}$ (3rd bit is: $1x2^7$): $209 - 128 = 81$;
$2^6 = 64_{10}$ (4th bit is: $1x2^6$): $81 - 64 = 17$;
$2^5 = 32_{10}$ (Higher than 17, the 5th bit is: $0x2^5$);
$2^4 = 16_{10}$ (6th bit is: $1x2^4$): $17 - 16 = 1$;
$2^3 = 8_{10}$ (Higher than 1, the 7th bit: $0x2^3$)
$2^2 = 4_{10}$ (Higher than 1, the 8th bit: $0x2^2$);
$2^1 = 2_{10}$ (Higher than 1, the 9th bit: $0x2^1$);
$2^0 = 1_{10}$ (10th bit - LSB): $1x2^0$;

Result: $721_{10} = 1011010001_2$

Mathematical Notations (7/16)

- This conversion method consists of continuously subtracting the highest power of 2 not higher than the remainder. If it can be subtracted, mark the corresponding position with '1', else with '0'.

- **Decimal to Binary Conversion:**
  - Other method: Continuously divide by 2. The division remainder value is the corresponding position binary digit.

$103_{10} = 100 + 3 = 1x10^2 + 0x10^1 + 3x10^0$

103/2 = 51	Remainder: 1 (LSB)
51/2 = 25	Remainder: 1
25/2 = 12	Remainder: 1
12/2 = 6	Remainder: 0
6/2 = 3	Remainder: 0
3/2 = 1	Remainder: 1
1/2 = 0	Remainder: 1 (MSB)

Result: $103_{10} = 1100111_2$

Mathematical Notations (8/16)

- **Binary to Decimal Conversion:**
  - **Example 2**: Convert the binary number 1010111_2 to decimal notation.
  - This method is the inversion of method presented in example 1.

The binary number was 7 bits:

MSB:	1	$= 1 \times 2^6 =$		64
6th bit:	0	$= 0 \times 2^5 =$		0
5th bit:	1	$= 1 \times 2^4 =$		16
4th bit:	0	$= 0 \times 2^3 =$		0
3rd bit:	1	$= 1 \times 2^2 =$		4
2nd bit:	1	$= 1 \times 2^1 =$		2
LSB:	1	$= 1 \times 2^0 =$	+	1
				87_{10}

Result: $1010111_2 = 87_{10}$

Mathematical Notations (9/16)

Binary to Hexadecimal Conversion:

- The conversion from an integer binary number to hexadecimal is accomplished by:
  - Breaking the binary number into 4-bit sections from the LSB to MSB;
  - Converting the 4-bit binary number to its hexadecimal equivalent.

- **Example 3**: Convert the binary value 1010111110110010_2 to hexadecimal notation.

```
1010   1111   1011   0010
A      F      B      2
```

Result: $1010111110110010_2 = 0xAFB2$

>> Contents

Mathematical Notations (10/16)

Hexadecimal to Binary Conversion:

- The conversion from an integer hexadecimal number to binary is accomplished by:
  - Converting the hexadecimal number to its 4-bit binary equivalent;
  - Combining the 4-bit sections by removing the spaces.

- **Example 4**: Convert the hexadecimal value `0x0AFB2` to binary notation.

Hexadecimal:	A	F	B	2
Binary:	1010	1111	1011	0010

Result: `0xAFB2` = 1010111110110010_2

Mathematical Notations (11/16)

Hexadecimal to Decimal Conversion:

- To convert from Hexadecimal to Decimal, multiply the value in each position by its hexadecimal weight and add each value;

- **Example 5**: Convert the hexadecimal number `0x0AFB2` to decimal notation.

$A*16^3$		$F*16^2$		$B*16^1$		$2*16^0$		
10*4096		15*256		11*16		2*1		
40960	+	3840	+	176	+	2	=	44978_{10}

Result: $0x0AFB2 = 44978_{10}$

Mathematical Notations (12/16)

- **Decimal to Hexadecimal Conversion (Repeated Division by 16):**
  - **Example 6**: Convert the decimal number 44978_{10} to hexadecimal notation.
  -

Division	Quotient	Remainder	Hexadecimal
44978/16	2811	2	2
2811/16	175	11	B2
175/16	10	15	FB2
10/16	0	10	0AFB2

Result: 44978_{10} = 0x0AFB2

Mathematical Notations (13/16)

- **Example:**
  - Sum of two binary numbers:

$$\begin{array}{r} 01011101_2 \\ +\ 00010110_2 \\ \hline 01110011_2 \end{array}$$

  - Multiplication of two binary numbers:

$$\begin{array}{r} 01011101_2 \\ \times\ \ 0110_2 \\ \hline 00000000 \\ 01011101 \\ 01011101 \\ +\ 00000000 \\ \hline 010001011110_2 \end{array}$$

Mathematical Notations (14/16)

- **Negative numbers representation:**
  - One's complement notation;
  - Two's complement notation.

- **One's complement:**
  - Complement each bit of the number represented;
  - Example: $0001\ 0100_2$;
  - One's complement → $1110\ 1011_2$;
  - MSB = 1, means that the representation corresponds to a negative number;
  - However, this representation cannot be used directly to carry out mathematical operations – it gives wrong results.

Mathematical Notations (15/16)

- **Two's complement:**
  - Perform one's complement and then add 1 to the result;
  - The task "subtract two numbers" has been transformed into "add the complement" - so a total different operation 'ADD' instead of 'SUB' is used.
  - **Example:** $0001\ 0100_2$;
  - Two's complement → $1110\ 1011 + 1 = 1110\ 1100_2$;
  - Example: Subtraction operation:

$0101\ 1101_2 - 0001\ 0100_2$

$$\begin{array}{rl} & 0101\ 1101_2 \\ + & 1110\ 1100_2 \quad \text{<- Two's complement representation} \\ \hline & 0100\ 1001_2 \quad \text{(Positive result: MSB =0)} \end{array}$$

Mathematical Notations (16/16)

- **Two's complement:**
  - Eliminates the double representation of the zero value;
  - Allows addition and subtraction operations to be carried out without need to take the sign of the operands into account;
  - Example: Subtraction operation:

$$1010\ 0011_2 - 0001\ 0110_2$$

$$\begin{array}{rl} & 1010\ 0011_2 \\ + & \underline{1110\ 1010_2} \quad \text{<- Two's complement} \\ & 1000\ 1101_2 \quad \text{(Negative result: MSB = 1)} \end{array}$$

Floating and Fixed-point Arithmetic (1/15)

- **Representation of fractional numbers:**

- **Two different ways:**
  - Fixed-point;
  - Floating-point.

- **Fixed-point representation:**
  - Allows us to represent the fractional part;
  - Consumes less processing resources than floating-point;

Floating and Fixed-point Arithmetic (2/15)

- **Representation of fractional numbers (continued):**
  - **Binary base signed numbers representation:**
  - In the sign and absolute value notation:
    - MSB is reserved to the sign (0: positive; 1: negative);
    - Magnitude of the value is represented by the others bits;
    - Value of zero has two representations.
  - Fixed point representation notation: signed numbers:

Binary value	000	001	010	011	100	101	110	111
Sign and absolute value	+0	+1	+2	+3	-0	-1	-2	-3
One's complement	+0	+1	+2	+3	-3	-2	-1	-0
Two's complement	+0	+1	+2	+3	-4	-3	-2	-1

Floating and Fixed-point Arithmetic (3/15)

Representation of fractional numbers (continued):

- **Fixed-point numerical representation:**
  - Easily implemented within a short memory space;
  - Rapid to implement;
  - Suited to real-time applications that do not have a floating point coprocessor.

- A fixed point integer value, A, with n bits, is given by the notation (UQp.q):
  - U: unsigned (no sign bit)
  - p + q = n identifies an unsigned number
  - p bits: integer component;
  - q bits: fractional component.

$$a_u = \frac{A}{2^q}$$

  - Limits: $0 \leq a_u \leq \frac{2^{p+q} - 1}{2^q}$
  - Resolution: $r = 2^{-q}$

Floating and Fixed-point Arithmetic (4/15)

- **Representation of fractional numbers (continued):**
  - **Fixed point numerical representation:**

Format	Minimum	Maximum	*r*	*n*	*p*	*q*
(UQ16.)	0	$2^{16}-1$	1	16	16	0
(UQ.16)	0	$1-2^{-16}$	2^{-16}	16	0	16
(Q15.)	-2^{15}	$2^{15}-1$	1	16	15	0
(Q.15)	-1	$1-2^{-15}$	2^{-15}	16	0	15
(UQ16.16)	0	$2^{16}-1$	2^{-16}	32	16	16
(Q15.16)	-2^{15}	$2^{15}-2^{-16}$	2^{-16}	32	15	16

Floating and Fixed-point Arithmetic (5/15)

Representation of fractional numbers: Fixed-point

- **Example 1:**
- Convert the decimal number 123.045_{10} into hexadecimal.

  - Decimal representation: 123.045_{10}
  - Integer component: $123_{10} = 7B_{16}$
  - Fractional component: $0.045_{10} = 0.0B8_{16}$
  - Result: $123.045_{10} = 7B.0B8_{16}$

>> Contents

Floating and Fixed-point Arithmetic (6/15)

Representation of fractional numbers: Fixed-point (cont.)

- **Example 2:**
- Convert the decimal number 8751.135_{10} into a hexadecimal representation.

  - Integer component: $8751_{10} = 222F_{16}$
  - Fractional component: $0.135_{10} = 0.228_{16}$
  - Result: $8751.135_{10} = 222F.228_{16}$

Floating and Fixed-point Arithmetic (7/15)

Representation of fractional numbers: Floating-point

- In floating-point number representation, the radix point position can float relatively to the significant digits of the number;
- This detail allows the support of a much wider range of values when compared with the fixed point number representation;
- Before 1958 IEEE 754 standard, the floating-point representation of numerical values was made through different formats and word widths;
- Nowadays, the IEEE 754 standard is accepted almost by all kind of computers.

Floating and Fixed-point Arithmetic (8/15)

Representation of fractional numbers: Floating-point (cont.)

- From the set of formats supported by IEEE 754 standard, the most widely used are the single-precision (32-bit) and double-precision (64-bit);
- Moreover, the standard also establishes others kinds of formats: "quad" binary and decimal floating-point (128-bit); and double decimal floating-point (64-bit);
- Less used are the extended precision (80-bit) and half-precision (16-bit) formats. The last one appears in the IEEE 754r proposed revision;
- Standard current version is IEEE 754-2008, which was published in August 2008.

Floating and Fixed-point Arithmetic (9/15)

Representation of fractional numbers: Floating-point (cont.)

- In the data structure of floating-point numbers, the most significant bit is the sign bit (S), followed by the exponent that is biased;
- The mantissa without the most significant bit is stored after the exponent in the fraction field:

- The exponent is biased by the $(2^{e-1}) - 1$, where e is the exponent number of bits;
- This solution was adopted because exponents must be signed, and the two's complement representation will require extra computational work if it was used.

Floating and Fixed-point Arithmetic (10/15)

Representation of fractional numbers: Floating-point (cont.)

- The value of the biased exponent and mantissa determines the data meaning:
  - The number is said to be *normalized* (most significant bit of the mantissa is 1), if exponent ranges between 0 and $2^e - 1$;
  - The number is said to be *de-normalized* (most significant bit of the mantissa is 0), if the exponent is 0 and fraction is not 0.

Floating and Fixed-point Arithmetic (11/15)

Representation of fractional numbers: Floating-point (cont.)

- The value of the biased exponent and the mantissa determines the data meaning:
  - The number is ±0 (depending on the sign bit), if exponent is 0 and fraction is 0;
    - `+0.0: 0x0000;`
      `-0.0: 0x8000;`
  - The number is ±infinity (depending on the sign bit), if exponent = $2^e - 1$ and fraction is 0;
  - The number being represented is not a number (NaN), if exponent = $2^e - 1$ and fraction is not 0.

Floating and Fixed-point Arithmetic (12/15)

Representation of fractional numbers: Floating-point (cont.)

- The IEEE 754 binary formats have the following organization:

Format	Sign	Exponent	Exponent bias	Mantissa	Total number of bits
Half	1	5	15	10	16
Single	1	8	127	23	32
Double	1	11	1023	52	64
Quad	1	15	16383	112	128

- A number X has the value:

$$X = (-1)^S \times 2^{\text{exponent} - \text{exponent bias}} \times \text{mantissa}$$

Floating and Fixed-point Arithmetic (13/15)

- **Representation of fractional numbers: Floating-point (cont.)**
  - **Example 3:** Consider the decimal number 14.2352_{10}.
  - The number integer part can be converted to the binary value 1110_2, while the fractional number part is converted to the binary value 0.00111100001010001111_2.
  - The number can be represented in the binary base as:

$$1110.00111100001010001111_2$$

Floating and Fixed-point Arithmetic (14/15)

Representation of fractional numbers: Floating-point (cont.)

- The number normalization requires that the radix is moved to the left:

$$1.11000111100001010001111_2 \times 2^3$$

- We can now code the floating-point representation in the single format (32-bit) as:
  - Sign (1-bit): `0` (the number is positive);
  - Exponent (8-bit): `127 + 3 = 130 = 10000010`;
  - Fraction (23-bit): `11000111100001010001111` (fractional part of the mantissa).

Floating and Fixed-point Arithmetic (15/15)

Representation of fractional numbers: Floating-point (cont.)

- Finally, the floating-point code:

Sign	Exponent	Fraction
0	10000010	11000111100001010001111

- The floating-point code in hexadecimal: `0x4163C28F`