Data Interoperability and Semantics

< Part 1. Encoding base data types >

ICM – Toolbox Engineering and Interoperability of Software Systems – Course unit on Data Interoperability and Semantics
M1 Cyber Physical and Social Systems – Course unit on Data Interoperability and Semantics
Maxime Lefrançois https://maxime-lefrancois.info
Course unit URL: https://ci.mines-stetienne.fr/cps2/course/data

Data Interoperability and Semantics
Outline

- < Part 1. Encoding base data types >
  - Part 1.1. Reminders: binary and hexadecimal strings
  - Part 1.2. Endianness
  - Example: MCF88 LoRa temperature, humidity and pressure sensor payload
  - Part 1.3. Computer number formats
  - Part 1.4. Character encoding
  - Part 1.5. Base32 and Base64 encoding
  - Part 1.6. Date and time
  - Part 1.7. XML Schema Datatypes
  - Part 1.8. Codes: countries, languages, …
  - Part 1.9. Quantities and Units of measure
  - Part 1.10. Colors

Data Interoperability and Semantics

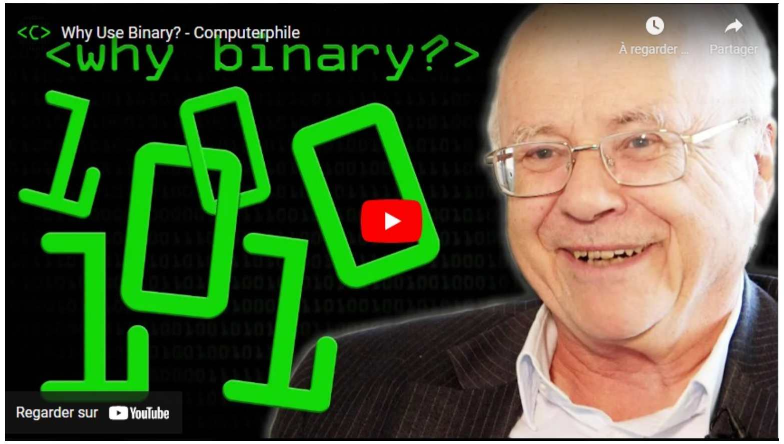
Part 1. Encoding base data types

Part 1.1. Reminders: binary and hexadecimal strings

ICM – Toolbox Engineering and Interoperability of Software Systems – Course unit on Data Interoperability and Semantics
M1 Cyber Physical and Social Systems – Course unit on Data Interoperability and Semantics
Maxime Lefrançois https://maxime-lefrancois.info
Course unit URL: https://ci.mines-stetienne.fr/cps2/course/data

Numbering systems for computers

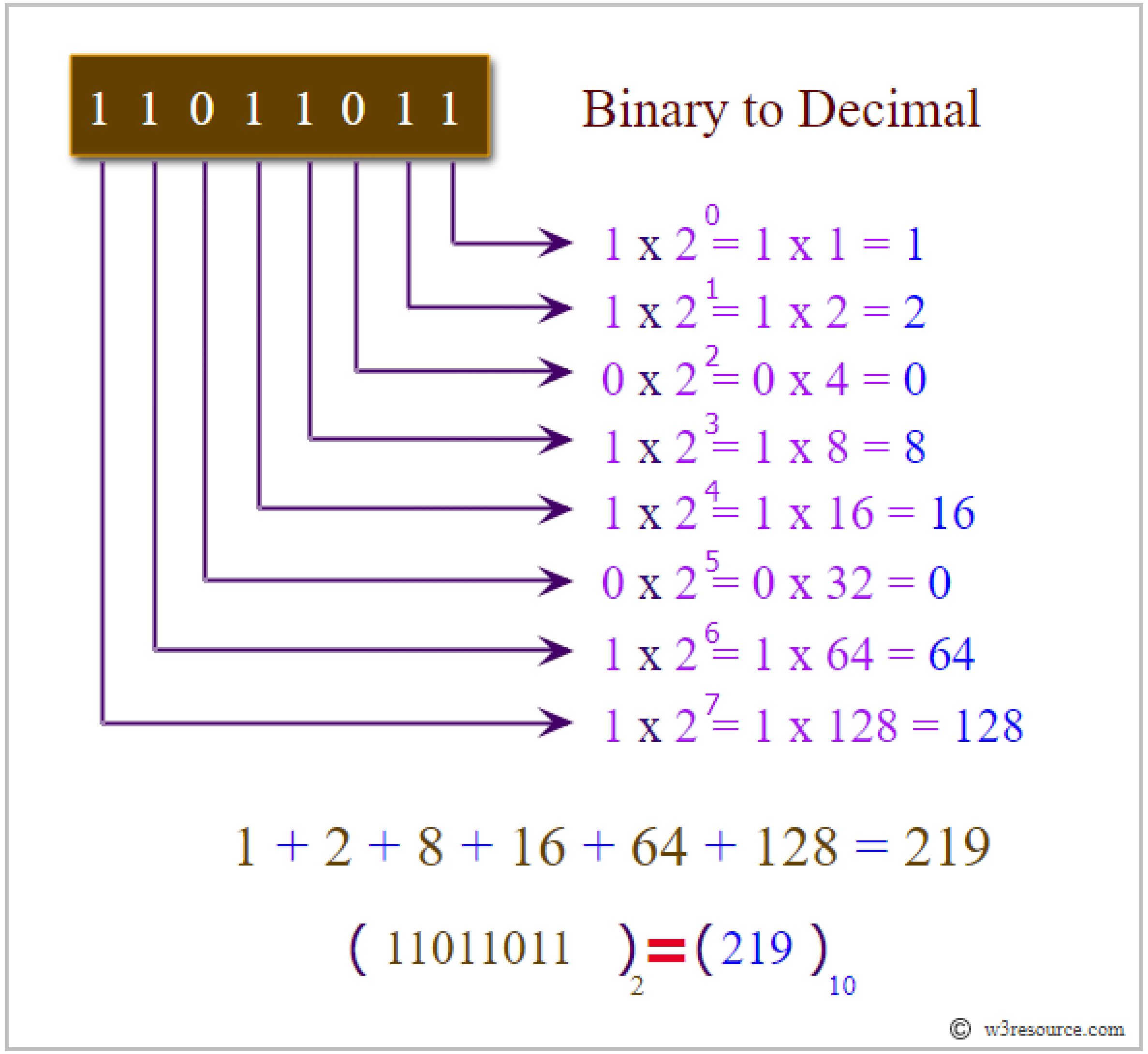
System	Base	Digits	ex python
Binary	2	0,1	0b"01111011"
Octal	8	0,1,2,3,4,5,6,7	0o"173"
Decimal	10	0,1,2,3,4,5,6,7,8,9	123
Hexadecimal	16	0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F	0x"7B"



It has been debated a lot at the beginning

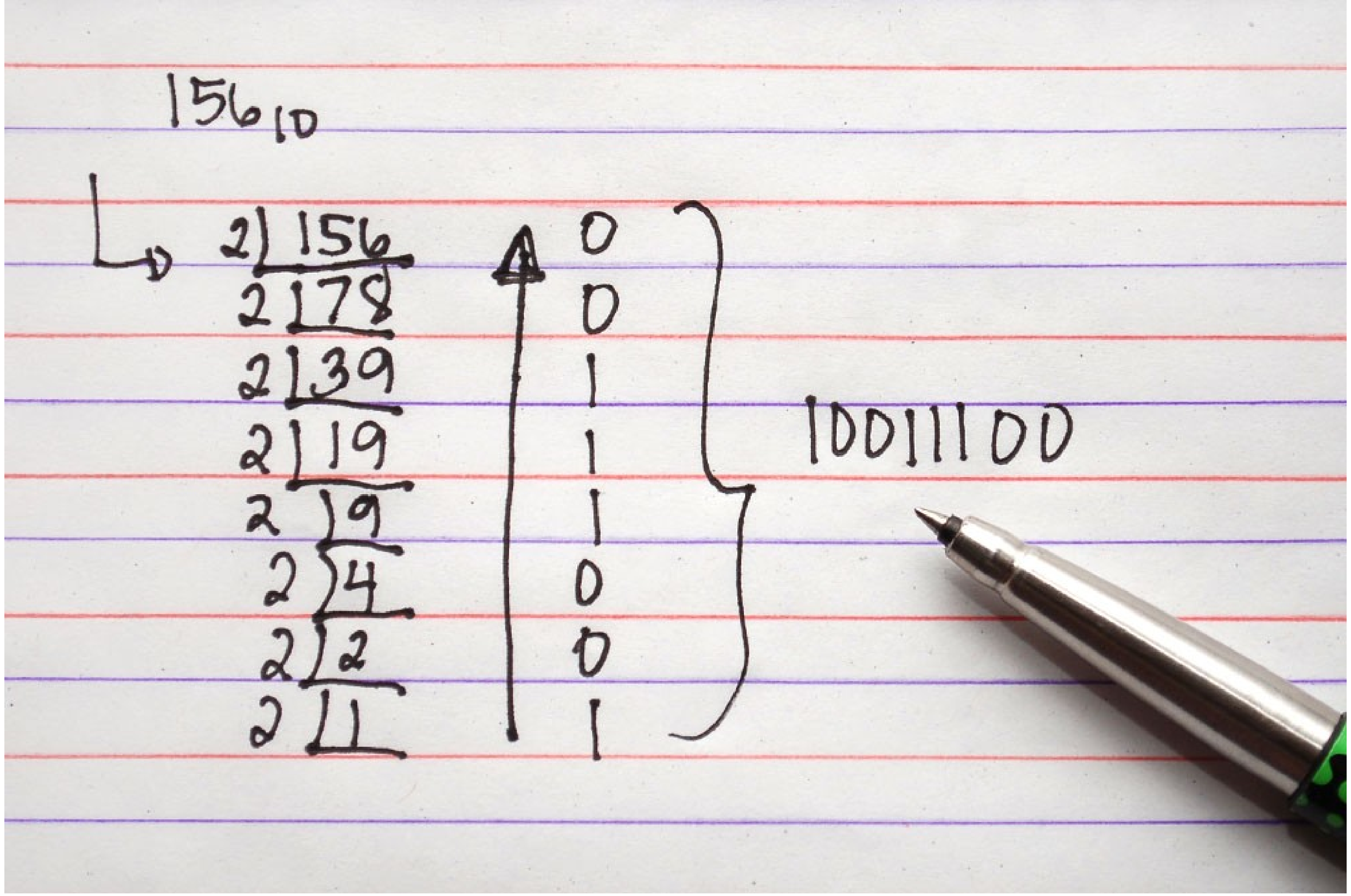
https://www.youtube.com/watch?v=thrx3SBEpL8

Tips: binary to decimal



https://www.w3resource.com/javascript-exercises/javascript-math-exercise-2.php

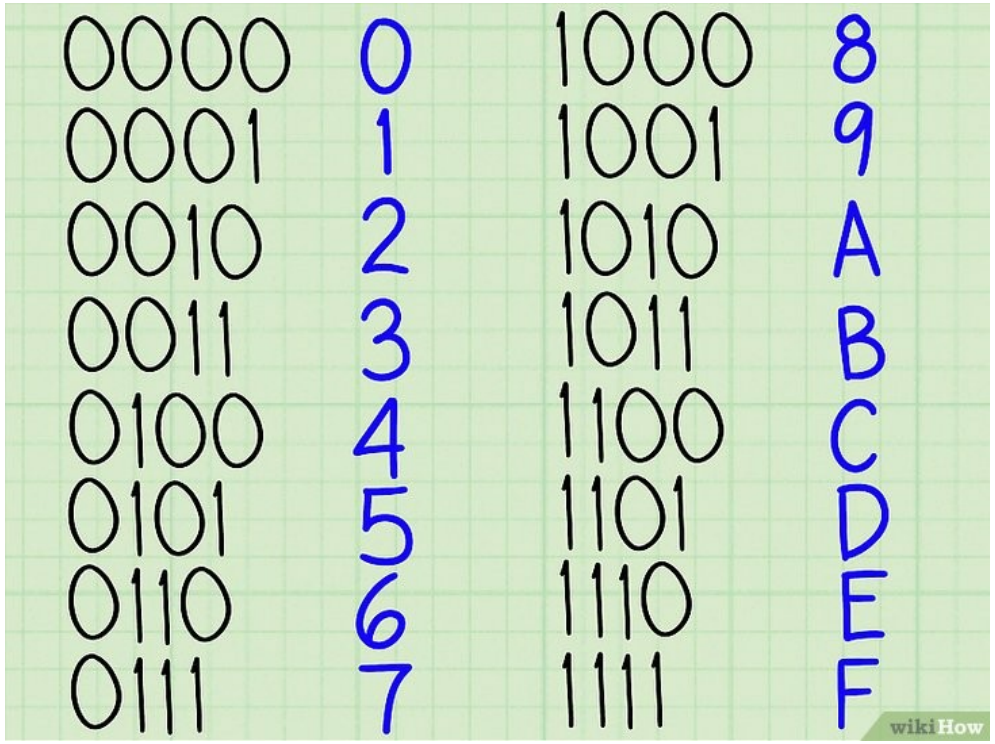
Tips: decimal to binary



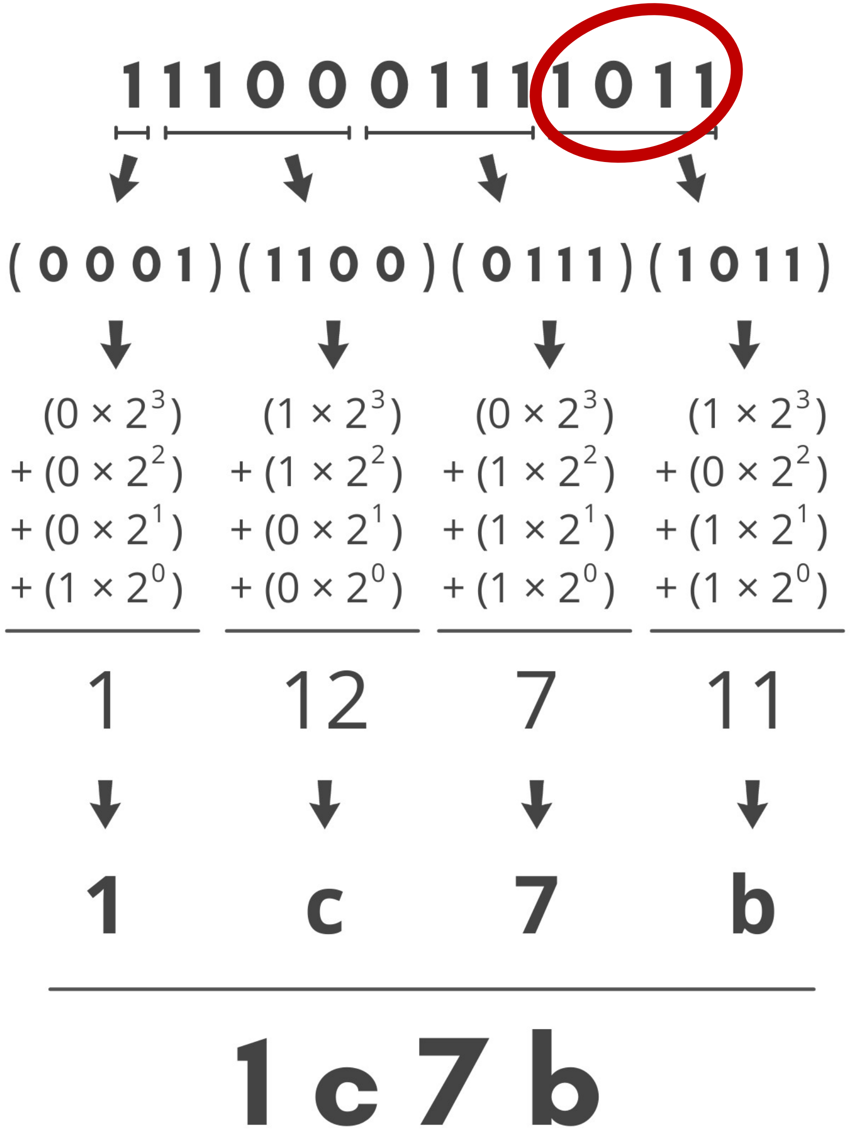
https://tex.stackexchange.com/questions/96399/how-can-i-illustrate-decimal-to-binary-conversion

Tips: binary to hexadecimal

Nibble - *In computing, a nibble is a four-bit aggregation, or half an octet. It is also known as half-byte or tetrade. In a networking or telecommunication context, the nibble is often called a semi-octet, quadbit, or quartet.*



Binary nibble to hexadecimal digit



https://www.inchcalculator.com/binary-to-hex-converter/

Programming with binary strings

```c
#include <stdio.h>
int main()
{
    unsigned char a = 0x65;
    unsigned char b = 0x09;

    // a = 0x65(0110 0101), b = 0x09(0000 1001)
    printf("a = %#.2X, b = %#.2x\n", a, b);

    // & bitwise AND operator
    // the result is 0x01(0000 0001)
    printf("a&b = %#.2X\n", a&b);

    // | bitwise OR operator
    // the result is 0x6D(0110 1101)
    printf("a|b = %#.2X\n", a|b);

    // ^ bitwise exclusive OR operator
    // the result is 0x6C(0110 1100)
    printf("a^b = %#.2X\n", a^b);

    // << Left shift operator
    // the result is 0xCA(1100 1010)
    printf("a << 1 = %#.2X\n", a << 1);

    // >> Right shift operator
    // the result is 0x32(0011 0010)
    printf("a >> 1 = %#.2X\n", a >> 1);

    // ~ bitwise One's Complement operator
    // the result is 0X9A (1001 1010)
    // or more precisely 0XFFFFFF9A (~ promotes to int)
    printf("~a = %#.2X\n", ~a);

    // Get 3 bits starting at position 2 (start index 0)
    // the result is 1(0001)
    printf("(a >> 2) & 0x7 = %#.1X\n", (a>>2)&7);

    return 0;
}
```

```
$ gcc main.c
$ ./a.out
a = 0X65, b = 0x09
a&b = 0X01
a|b = 0X6D
a^b = 0X6C
a << 1 = 0XCA
a >> 1 = 0X32
~a = 0XFFFFFF9A
(a >> 2) & 0x7 = 0X1
```

Data Interoperability and Semantics

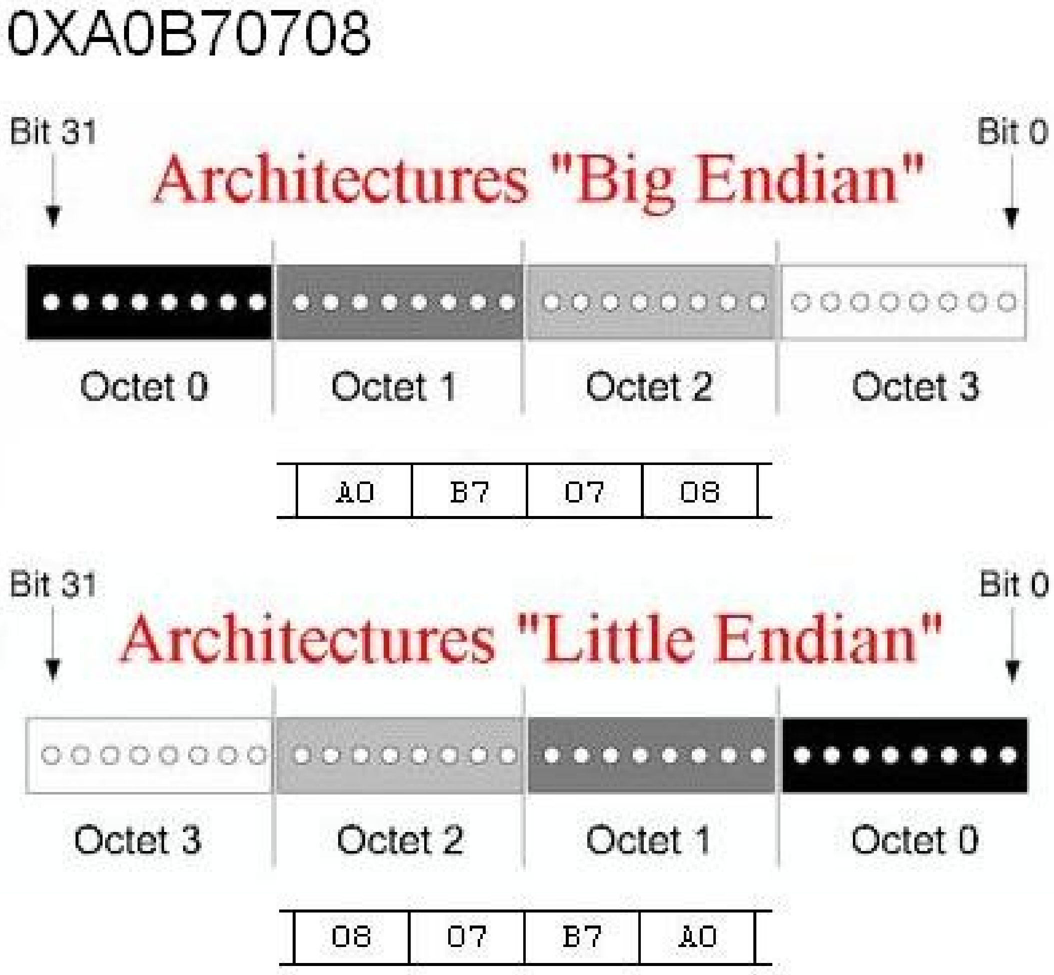
Part 1. Encoding base data types

Part 1.2. Endianness

ICM – Toolbox Engineering and Interoperability of Software Systems – Course unit on Data Interoperability and Semantics
M1 Cyber Physical and Social Systems – Course unit on Data Interoperability and Semantics
Maxime Lefrançois https://maxime-lefrancois.info
Course unit URL: https://ci.mines-stetienne.fr/cps2/course/data

Endianness

In computing, endianness is the order or sequence of bytes of a word of digital data in computer memory. Endianness is primarily expressed as ***big-endian (BE)*** *or* ***little-endian (LE)****.*



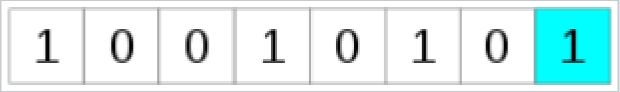
https://www.sqlpac.com/fr/documents/sybase-ase-12.5.3-dump-load-cross-platforms.html

Acronyms
- LSB - Least significant byte
- MSB - Most significant byte

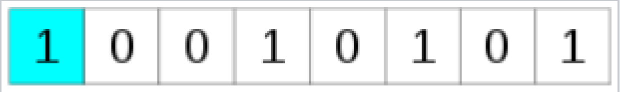
Bit endianness or bit-level endianness

Bit endianness or bit-level endianness refers to the transmission order of bits over a serial medium

See course Programming Connected Devices:
- Least significant bit first: used in RS-232, Ethernet, USB…
- Most significant bit first: used in I²C



The binary representation of decimal 149, with the LSB highlighted. The LSB represents a value of 1.



The unsigned binary representation of decimal 149, with the MSB highlighted. The MSB represents a value of 128.

https://en.wikipedia.org/wiki/Bit_numbering

Example: WS2812B color leds

Worldsemi

WS2812B

Intelligent control LED
integrated light source

Composition of 24bit data:

G7	G6	G5	G4	G3	G2	G1	G0	R7	R6	R5	R4	R3	R2	R1	R0	B7	B6	B5	B4	B3	B2	B1	B0

Note: Follow the order of GRB to sent data and the high bit sent at first.

Data Interoperability and Semantics

Part 1. Encoding base data types

Example: MCF88 LoRa temperature, humidity and pressure sensor payload

ICM – Toolbox Engineering and Interoperability of Software Systems – Course unit on Data Interoperability and Semantics
M1 Cyber Physical and Social Systems – Course unit on Data Interoperability and Semantics
Maxime Lefrançois https://maxime-lefrancois.info
Course unit URL: https://ci.mines-stetienne.fr/cps2/course/data

Author: Colognato Stefano

Created: 29/09/2016

Modified: 30/11/2018

MCF88 DATA FRAME FORMAT 1.17

1.2 TEMPERATURE/PRESSURE/HUMIDITY

HOME

name	size [byte]	hex value	mean
Uplink ID	1 byte	04	Temperature/Pressure/Humidity
Data	10 byte	XX XX …..	Measure 1, refer to Note1
	10 byte	XX XX …..	Measure 2, refer to Note1
	10 byte	XX XX …..	Measure 3, refer to Note1
Batt %	1 byte (optional)	XX	Battery percentage
RFU	4 byte (optional)	XX XX XX XX	Optional RFU byte

Note1:

The 10 bytes for each measurement are divided as follows:

- 4 bytes are for the date and time. The MSB (most significant byte) is on the right so they must be read from the right. The 4 byte in reverse order are as follows:
    - 7 bit for the offset of the year, starting from the year 2000
    - 4 bit per month
    - 5 bit for day of the month
    - 5 bits for hour
    - 6 bits for minutes
    - 5 bits for half the seconds. The seconds range is from 0 to 31, so the result should be multiplied by 2 to find the actual seconds of the measurement.
- 2 bytes for temperature. The temperature is represented by a signed integer with the least significant byte first. The temperature is expressed in hundreds of a °C degree.
- 1 byte for humidity. Relative humidity is an unsigned integer corresponding to twice the percentage of humidity.
- 3 bytes for pressure. Pressure is an unsigned integer with the least significant byte first; it is expressed in Pascal.

Example

Sample payload:

04dc7e3721b40a47608801dd7e3721b10a43608801e07e3721b20a425d8801

Example: MCF88 LoRa sensors

1.2 TEMPERATURE/PRESSURE/HUMIDITY

HOME

name	size [byte]	hex value	mean
Uplink ID	1 byte	04	Temperature/Pressure/Humidity
Data	10 byte	XX XX …..	Measure 1, refer to Note1
	10 byte	XX XX …..	Measure 2, refer to Note1
	10 byte	XX XX …..	Measure 3, refer to Note1
Batt %	1 byte (optional)	XX	Battery percentage
RFU	4 byte (optional)	XX XX XX XX	Optional RFU byte

Note1:

The 10 bytes for each measurement are divided as follows:

- 4 bytes are for the date and time. The MSB (most significant byte) is on the right so they must be read from the right. The 4 byte in reverse order are as follows:
  - 7 bit for the offset of the year, starting from the year 2000
  - 4 bit per month
  - 5 bit for day of the month
  - 5 bits for hour
  - 6 bits for minutes
  - 5 bits for half the seconds. The seconds range is from 0 to 31, so the result should be multiplied by 2 to find the actual seconds of the measurement.
- 2 bytes for temperature. The temperature is represented by a signed integer with the least significant byte first. The temperature is expressed in hundreds of a °C degree.
- 1 byte for humidity. Relative humidity is an unsigned integer corresponding to twice the percentage of humidity.
- 3 bytes for pressure. Pressure is an unsigned integer with the least significant byte first; it is expressed in Pascal.

Example

Sample payload:
04dc7e3721b40a47608801dd7e3721b10a43608801e07e3721b20a425d8801
Remove the first byte and divide the other 30 into 3 parts by 10 byte that correspond to 3 measurements.
The 3 measurements will be:

- *dc7e3721b40a47608801*
- *dd7e3721b10a43608801*
- *e07e3721b20a425d8801*

Author: Colognato Stefano

1.2 TEMPERATURE/PRESSURE/HUMIDITY

HOME

name	size [byte]	hex value	mean
Uplink ID	1 byte	04	Temperature/Pressure/Humidity
Data	10 byte	XX XX …..	Measure 1, refer to Note1
	10 byte	XX XX …..	Measure 2, refer to Note1
	10 byte	XX XX …..	Measure 3, refer to Note1
Batt %	1 byte (optional)	XX	Battery percentage
RFU	4 byte (optional)	XX XX XX XX	Optional RFU byte

Note1:

The 10 bytes for each measurement are divided as follows:

- 4 bytes are for the date and time. The MSB (most significant byte) is on the right so they must be read from the right. The 4 byte in reverse order are as follows:
  - 7 bit for the offset of the year, starting from the year 2000
  - 4 bit per month
  - 5 bit for day of the month
  - 5 bits for hour
  - 6 bits for minutes
  - 5 bits for half the seconds. The seconds range is from 0 to 31, so the result should be multiplied by 2 to find the actual seconds of the measurement.
- 2 bytes for temperature. The temperature is represented by a signed integer with the least significant byte first. The temperature is expressed in hundreds of a °C degree.
- 1 byte for humidity. Relative humidity is an unsigned integer corresponding to twice the percentage of humidity.
- 3 bytes for pressure. Pressure is an unsigned integer with the least significant byte first; it is expressed in Pascal.

Example

Sample payload:
04dc7e3721b40a47608801dd7e3721b10a43608801e07e3721b20a425d8801
Remove the first byte and divide the other 30 into 3 parts by 10 byte that correspond to 3 measurements.
The 3 measurements will be:

- *dc7e3721b40a47608801*
- *dd7e3721b10a43608801*
- *e07e3721b20a425d8801*

Decipher the first measurement dividing it by groups and applying the necessary transformations:

- *Measurement date: dc 7e 37 21*
  - *Byte swapping, result: 21 37 7e dc*

1.2 TEMPERATURE/PRESSURE/HUMIDITY

HOME

name	size [byte]	hex value	mean
Uplink ID	1 byte	04	Temperature/Pressure/Humidity
Data	10 byte	XX XX …..	Measure 1, refer to Note1
	10 byte	XX XX …..	Measure 2, refer to Note1
	10 byte	XX XX …..	Measure 3, refer to Note1
Batt %	1 byte (optional)	XX	Battery percentage
RFU	4 byte (optional)	XX XX XX XX	Optional RFU byte

Note1:

The 10 bytes for each measurement are divided as follows:

- 4 bytes are for the date and time. The MSB (most significant byte) is on the right so they must be read from the right. The 4 byte in reverse order are as follows:
  - 7 bit for the offset of the year, starting from the year 2000
  - 4 bit per month
  - 5 bit for day of the month
  - 5 bits for hour
  - 6 bits for minutes
  - 5 bits for half the seconds. The seconds range is from 0 to 31, so the result should be multiplied by 2 to find the actual seconds of the measurement.
- 2 bytes for temperature. The temperature is represented by a signed integer with the least significant byte first. The temperature is expressed in hundreds of a °C degree.
- 1 byte for humidity. Relative humidity is an unsigned integer corresponding to twice the percentage of humidity.
- 3 bytes for pressure. Pressure is an unsigned integer with the least significant byte first; it is expressed in Pascal.

Example

Sample payload:
04dc7e3721b40a47608801dd7e3721b10a43608801e07e3721b20a425d8801
Remove the first byte and divide the other 30 into 3 parts by 10 byte that correspond to 3 measurements.
The 3 measurements will be:

- *dc7e3721b40a47608801*
- *dd7e3721b10a43608801*
- *e07e3721b20a425d8801*

Decipher the first measurement dividing it by groups and applying the necessary transformations:

- *Measurement date: dc 7e 37 21*
  - *Byte swapping, result: 21 37 7e dc*
  - *The result in bits will be: 00100001 00110111 01111110 11011100*
    - *The bits are divided as explained above*
      - *Year: 0010000*
        - *Result: 16*
          - *2000+16 = 2016*
      - *Month: 1001*
        - *Result: 9*
      - *Day: 10111*
        - *Result: 23*
      - *Hour: 01111*
        - *Result: 15*
      - *Minutes: 110110*
        - *Result: 54*
      - *Seconds: 11100*
        - *Result: 28*
          - *28*2 = 56*
  - *The date of the measurament will be: 23/09/2016 15:54:56.*

Created: 29/09/2016

Modified: 30/11/2018

MCF88 DATA FRAME FORMAT 1.17

Author: Colognato Stefano

1.2 TEMPERATURE/PRESSURE/HUMIDITY

HOME

name	size [byte]	hex value	mean
Uplink ID	1 byte	04	Temperature/Pressure/Humidity
Data	10 byte	XX XX …..	Measure 1, refer to Note1
	10 byte	XX XX …..	Measure 2, refer to Note1
	10 byte	XX XX …..	Measure 3, refer to Note1
Batt %	1 byte (optional)	XX	Battery percentage
RFU	4 byte (optional)	XX XX XX XX	Optional RFU byte

Note1:

The 10 bytes for each measurement are divided as follows:

- 4 bytes are for the date and time. The MSB (most significant byte) is on the right so they must be read from the right. The 4 byte in reverse order are as follows:
  - 7 bit for the offset of the year, starting from the year 2000
  - 4 bit per month
  - 5 bit for day of the month
  - 5 bits for hour
  - 6 bits for minutes
  - 5 bits for half the seconds. The seconds range is from 0 to 31, so the result should be multiplied by 2 to find the actual seconds of the measurement.
- 2 bytes for temperature. The temperature is represented by a signed integer with the least significant byte first. The temperature is expressed in hundreds of a °C degree.
- 1 byte for humidity. Relative humidity is an unsigned integer corresponding to twice the percentage of humidity.
- 3 bytes for pressure. Pressure is an unsigned integer with the least significant byte first; it is expressed in Pascal.

Example

Sample payload:

04dc7e3721b40a47608801dd7e3721b10a43608801e07e3721b20a425d8801

Remove the first byte and divide the other 30 into 3 parts by 10 byte that correspond to 3 measurements.

The 3 measurements will be:

- *dc7e3721b40a47608801*
- *dd7e3721b10a43608801*
- *e07e3721b20a425d8801*

Decipher the first measurement dividing it by groups and applying the necessary transformations:

- *Measurement date: dc 7e 37 21*
  - *Byte swapping, result: 21 37 7e dc*
  - *The result in bits will be: 00100001 00110111 01111110 11011100*
    - *The bits are divided as explained above*
      - *Year: 0010000*
        - *Result: 16*
          - *2000+16 = 2016*
      - *Month: 1001*
        - *Result: 9*
      - *Day: 10111*
        - *Result: 23*
      - *Hour: 01111*
        - *Result: 15*
      - *Minutes: 110110*
        - *Result: 54*
      - *Seconds: 11100*
        - *Result: 28*
          - *28*2 = 56*
  - *The date of the measurament will be: 23/09/2016 15:54:56.*
- *Temperature: b40a*
  - *Byte swapping, result: 0ab4*
    - *The result (with sign) will be +2740 with two decimal places, then + 27.40 °C.*
- *Humidity: 47*
  - *In decimal is 71, the humidity is 71/2 = 35.5% rH.*
- *Pressure: 608801*
  - *Byte swapping, result: 018860*
    - *In decimal, the result is 100448, with two decimal places the pressure is 1004.48 hPa.*

Author: Colognato Stefano

1.2 TEMPERATURE/PRESSURE/HUMIDITY

HOME

name	size [byte]	hex value	mean
Uplink ID	1 byte	04	Temperature/Pressure/Humidity
Data	10 byte	XX XX …..	Measure 1, refer to Note1
	10 byte	XX XX …..	Measure 2, refer to Note1
	10 byte	XX XX …..	Measure 3, refer to Note1
Batt %	1 byte (optional)	XX	Battery percentage
RFU	4 byte (optional)	XX XX XX XX	Optional RFU byte

Note1:

The 10 bytes for each measurement are divided as follows:

- 4 bytes are for the date and time. The MSB (most significant byte) is on the right so they must be read from the right. The 4 byte in reverse order are as follows:
  - 7 bit for the offset of the year, starting from the year 2000
  - 4 bit per month
  - 5 bit for day of the month
  - 5 bits for hour
  - 6 bits for minutes
  - 5 bits for half the seconds. The seconds range is from 0 to 31, so the result should be multiplied by 2 to find the actual seconds of the measurement.
- 2 bytes for temperature. The temperature is represented by a signed integer with the least significant byte first. The temperature is expressed in hundreds of a °C degree.
- 1 byte for humidity. Relative humidity is an unsigned integer corresponding to twice the percentage of humidity.
- 3 bytes for pressure. Pressure is an unsigned integer with the least significant byte first; it is expressed in Pascal.

Example

Sample payload:
04dc7e3721b40a47608801dd7e3721b10a43608801e07e3721b20a425d8801
Remove the first byte and divide the other 30 into 3 parts by 10 byte that correspond to 3 measurements.
The 3 measurements will be:

- *dc7e3721b40a47608801*
- *dd7e3721b10a43608801*
- *e07e3721b20a425d8801*

Decipher the first measurement dividing it by groups and applying the necessary transformations:

- *Measurement date: dc 7e 37 21*
  - *Byte swapping, result: 21 37 7e dc*
  - *The result in bits will be: 00100001 00110111 01111110 11011100*
    - *The bits are divided as explained above*
      - *Year: 0010000*
        - *Result: 16*
          - *2000+16 = 2016*
      - *Month: 1001*
        - *Result: 9*
      - *Day: 10111*
        - *Result: 23*
      - *Hour: 01111*
        - *Result: 15*
      - *Minutes: 110110*
        - *Result: 54*
      - *Seconds: 11100*
        - *Result: 28*
          - *28*2 = 56*
  - *The date of the measurament will be: 23/09/2016 15:54:56.*
- *Temperature: b40a*
  - *Byte swapping, result: 0ab4*
    - *The result (with sign) will be +2740 with two decimal places, then + 27.40 °C.*
- *Humidity: 47*
  - *In decimal is 71, the humidity is 71/2 = 35.5% rH.*
- *Pressure: 608801*
  - *Byte swapping, result: 018860*
    - *In decimal, the result is 100448, with two decimal places the pressure is 1004.48 hPa.*

Data Interoperability and Semantics

Part 1. Encoding base data types

Part 1.3. Computer number formats

ICM – Toolbox Engineering and Interoperability of Software Systems – Course unit on Data Interoperability and Semantics
M1 Cyber Physical and Social Systems – Course unit on Data Interoperability and Semantics
Maxime Lefrançois https://maxime-lefrancois.info
Course unit URL: https://ci.mines-stetienne.fr/cps2/course/data

C number data types

- char (8 bits) – [0, 255]
- short/int (16 bits) - [-32,767, +32,767] or [0, 65,535]
- long (32 bits) - [−2,147,483,647, +2,147,483,647] or [0, 4,294,967,295]
- long long (64 bits) [−9,223,372,036,854,775,807, +9,223,372,036,854,775,807] or <<positive>>
- float - IEEE 754 single-precision binary floating-point format (32 bits)
- double - IEEE 754 double-precision binary floating-point format (64 bits)

Integer encoding

Unsigned

$$B2U(X) = \sum_{i=0}^{w-1} x_i \cdot 2^i$$

Two's Complement

$$B2T(X) = -x_{w-1} \cdot 2^{w-1} + \sum_{i=0}^{w-2} x_i \cdot 2^i$$

Sign
Bit

```
short int x =  15213;
short int y = -15213;
```

- C `short` 2 bytes long

	Decimal	Hex	Binary
x	15213	3B 6D	00111011 01101101
y	-15213	C4 93	11000100 10010011

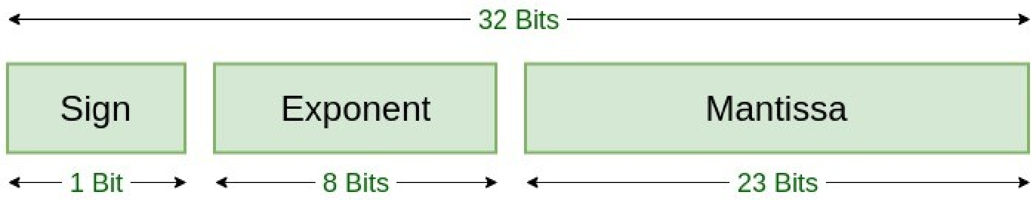
Sign Bit

- For 2's complement, most significant bit indicates sign
  - 0 for nonnegative
  - 1 for negative

https://slideplayer.com/slide/5048952/

IEEE 754 single/double encoding

https://www.geeksforgeeks.org/ieee-standard-754-floating-point-numbers/



Single Precision
IEEE 754 Floating-Point Standard

$$-1^{s} \times 2^{(\text{exp}-127)} \times 1.frac$$

exceptional cases:

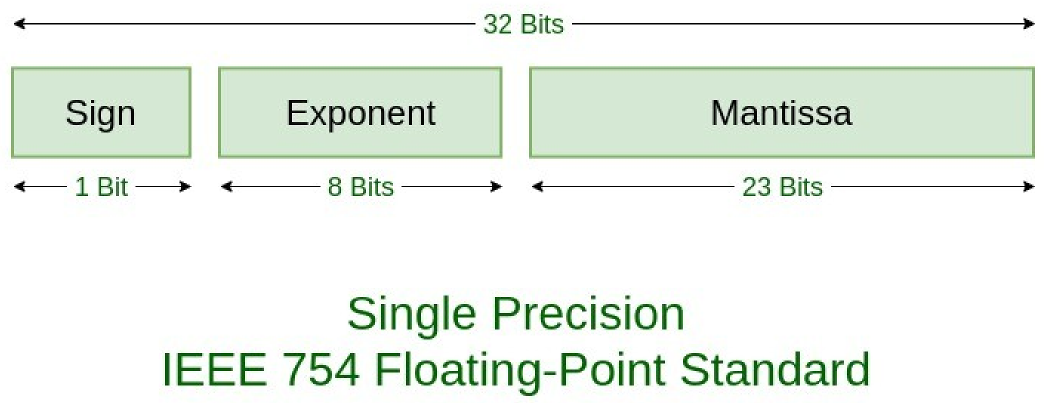
- If $E = 255$ and F is nonzero, then $x = NaN$ (“Not a number”).
- If $E = 255$, F is zero, and S is 1, then $x = -$ Infinity.
- If $E = 255$, F is zero, and S is 0, then $x = +$ Infinity.
- If $0 < E < 255$, then $x = (-1)^s \times (1.F) \times 2^{E-127}$, where $1.F$ represents the binary number created by prefixing F with an implicit leading 1 and a binary point.
- If $E = 0$ and F is nonzero, then $x = (-1)^s \times (0.F) \times 2^{-126}$. This is an “unnormalized” value.
- If $E = 0$, F is zero, and S is 1, then $x = -0$.
- If $E = 0$, F is zero, and S is 0, then $x = 0$.

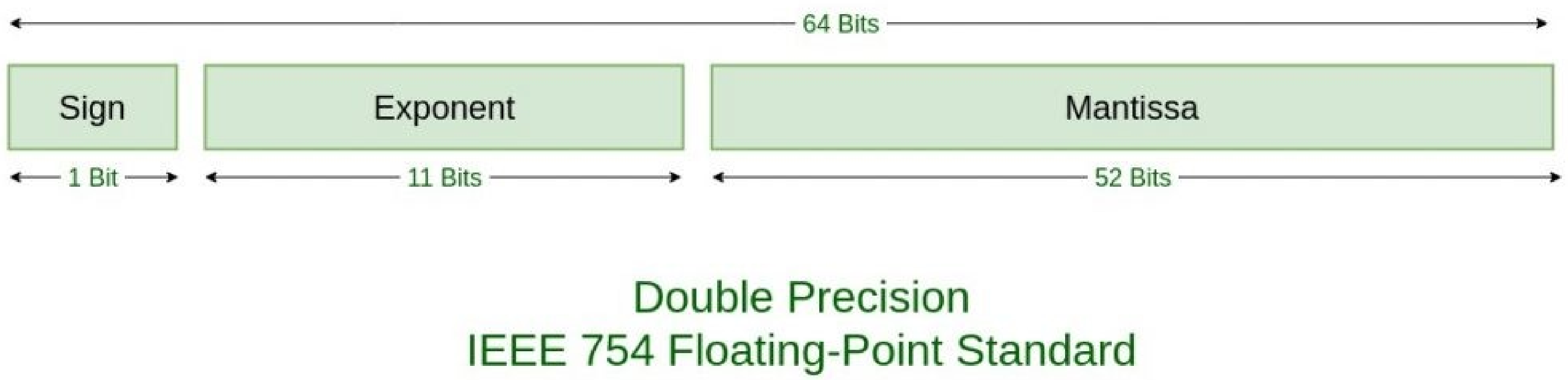
Simple examples of conversions

$$0\ 10000000\ 00000000000000000000000 = (-1)^0 \times (1.0_2) \times 2^{128-127} = 2.0$$
$$0\ 10000001\ 10100000000000000000000 = (-1)^0 \times (1.101_2) \times 2^{129-127} = 6.5$$
$$1\ 10000001\ 10100000000000000000000 = (-1)^1 \times (1.101_2) \times 2^{129-127} = -6.5.$$

IEEE 754 single/double encoding

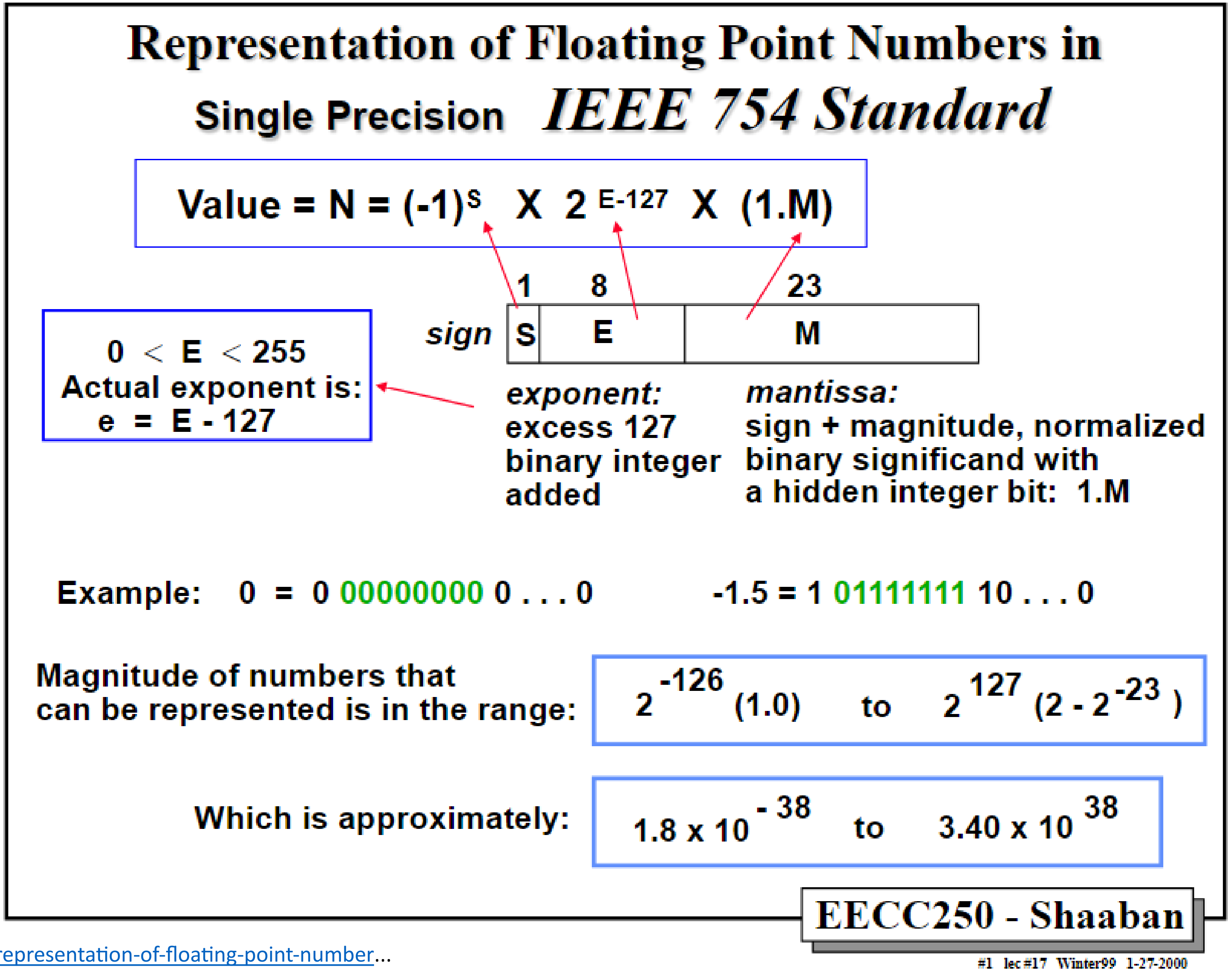
https://www.geeksforgeeks.org/ieee-standard-754-floating-point-numbers/





https://www.geeksforgeeks.org/ieee-standard-754-floating-point-numbers/

IEEE 754 single/double/quadruple encoding



https://studylib.net/doc/17671880/ieee-754-standard-representation-of-floating-point-number...

Data Interoperability and Semantics

Part 1. Encoding base data types

Part 1.4. Character encoding

ICM – Toolbox Engineering and Interoperability of Software Systems – Course unit on Data Interoperability and Semantics
M1 Cyber Physical and Social Systems – Course unit on Data Interoperability and Semantics
Maxime Lefrançois https://maxime-lefrancois.info
Course unit URL: https://ci.mines-stetienne.fr/cps2/course/data

Character encoding

In computing, data storage, and data transmission, character encoding is used to represent a repertoire of characters by some kind of encoding system that assigns a number to each character for digital representation

— https://en.wikipedia.org/wiki/Character_encoding

Common character encodings [edit]

- ISO 646
  - ASCII
- EBCDIC
- ISO 8859:
  - ISO 8859-1 Western Europe
  - ISO 8859-2 Western and Central Europe
  - ISO 8859-3 Western Europe and South European (Turkish, Maltese plus Esperanto)
  - ISO 8859-4 Western Europe and Baltic countries (Lithuania, Estonia, Latvia and Lapp)
  - ISO 8859-5 Cyrillic alphabet
  - ISO 8859-6 Arabic
  - ISO 8859-7 Greek
  - ISO 8859-8 Hebrew
  - ISO 8859-9 Western Europe with amended Turkish character set
  - ISO 8859-10 Western Europe with rationalised character set for Nordic languages, including complete Icelandic set
  - ISO 8859-11 Thai
  - ISO 8859-13 Baltic languages plus Polish
  - ISO 8859-14 Celtic languages (Irish Gaelic, Scottish, Welsh)
  - ISO 8859-15 Added the Euro sign and other rationalisations to ISO 8859-1
  - ISO 8859-16 Central, Eastern and Southern European languages (Albanian, Bosnian, Croatian, Hungarian, Polish, Romanian, Serbian and Slovenian, but also French, German, Italian and Irish Gaelic)
- CP437, CP720, CP737, CP850, CP852, CP855, CP857, CP858, CP860, CP861, CP862, CP863, CP865, CP866, CP869, CP872
- MS-Windows character sets:
  - Windows-1250 for Central European languages that use Latin script, (Polish, Czech, Slovak, Hungarian, Slovene, Serbian, Croatian, Bosnian, Romanian and Albanian)
  - Windows-1251 for Cyrillic alphabets
  - Windows-1252 for Western languages
  - Windows-1253 for Greek
  - Windows-1254 for Turkish
  - Windows-1255 for Hebrew
  - Windows-1256 for Arabic
  - Windows-1257 for Baltic languages
  - Windows-1258 for Vietnamese
- Mac OS Roman
- KOI8-R, KOI8-U, KOI7
- MIK
- ISCII
- TSCII
- VISCII
- JIS X 0208 is a widely deployed standard for Japanese character encoding that has several encoding forms.
  - Shift JIS (Microsoft Code page 932 is a dialect of Shift_JIS)
  - EUC-JP
  - ISO-2022-JP
- JIS X 0213 is an extended version of JIS X 0208.
  - Shift_JIS-2004
  - EUC-JIS-2004
  - ISO-2022-JP-2004
- Chinese Guobiao
  - GB 2312
  - GBK (Microsoft Code page 936)
  - GB 18030
- Taiwan Big5 (a more famous variant is Microsoft Code page 950)
  - Hong Kong HKSCS
- Korean
  - KS X 1001 is a Korean double-byte character encoding standard
  - EUC-KR
  - ISO-2022-KR
- Unicode (and subsets thereof, such as the 16-bit 'Basic Multilingual Plane')
  - UTF-8
  - UTF-16
  - UTF-32
- ANSEL or ISO/IEC 6937

Let’s focus on the main standards

- ASCII
- UTF-8

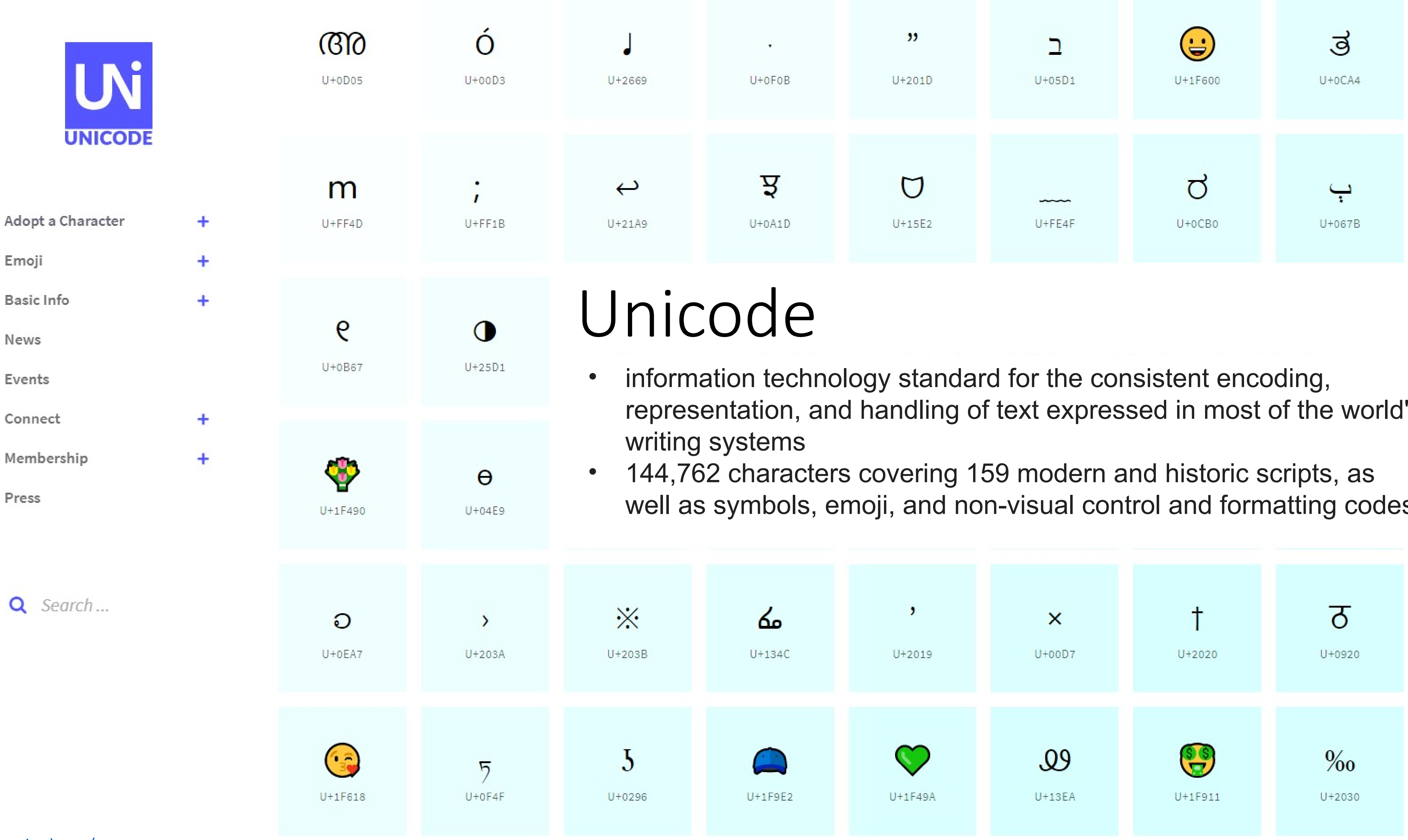
Common character encodings [edit]

- ISO 646
  - ASCII
- EBCDIC
- ISO 8859:
  - ISO 8859-1 Western Europe
  - ISO 8859-2 Western and Central Europe
  - ISO 8859-3 Western Europe and South European (Turkish, Maltese plus Esperanto)
  - ISO 8859-4 Western Europe and Baltic countries (Lithuania, Estonia, Latvia and Lapp)
  - ISO 8859-5 Cyrillic alphabet
  - ISO 8859-6 Arabic
  - ISO 8859-7 Greek
  - ISO 8859-8 Hebrew
  - ISO 8859-9 Western Europe with amended Turkish character set
  - ISO 8859-10 Western Europe with rationalised character set for Nordic languages, including complete Icelandic set
  - ISO 8859-11 Thai
  - ISO 8859-13 Baltic languages plus Polish
  - ISO 8859-14 Celtic languages (Irish Gaelic, Scottish, Welsh)
  - ISO 8859-15 Added the Euro sign and other rationalisations to ISO 8859-1
  - ISO 8859-16 Central, Eastern and Southern European languages (Albanian, Bosnian, Croatian, Hungarian, Polish, Romanian, Serbian and Slovenian, but also French, German, Italian and Irish Gaelic)
- CP437, CP720, CP737, CP850, CP852, CP855, CP857, CP858, CP860, CP861, CP862, CP863, CP865, CP866, CP869, CP872
- MS-Windows character sets:
  - Windows-1250 for Central European languages that use Latin script, (Polish, Czech, Slovak, Hungarian, Slovene, Serbian, Croatian, Bosnian, Romanian and Albanian)
  - Windows-1251 for Cyrillic alphabets
  - Windows-1252 for Western languages
  - Windows-1253 for Greek
  - Windows-1254 for Turkish
  - Windows-1255 for Hebrew
  - Windows-1256 for Arabic
  - Windows-1257 for Baltic languages
  - Windows-1258 for Vietnamese
- Mac OS Roman
- KOI8-R, KOI8-U, KOI7
- MIK
- ISCII
- TSCII
- VISCII
- JIS X 0208 is a widely deployed standard for Japanese character encoding that has several encoding forms.
  - Shift JIS (Microsoft Code page 932 is a dialect of Shift_JIS)
  - EUC-JP
  - ISO-2022-JP
- JIS X 0213 is an extended version of JIS X 0208.
  - Shift_JIS-2004
  - EUC-JIS-2004
  - ISO-2022-JP-2004
- Chinese Guobiao
  - GB 2312
  - GBK (Microsoft Code page 936)
  - GB 18030
- Taiwan Big5 (a more famous variant is Microsoft Code page 950)
  - Hong Kong HKSCS
- Korean
  - KS X 1001 is a Korean double-byte character encoding standard
  - EUC-KR
  - ISO-2022-KR
- Unicode (and subsets thereof, such as the 16-bit 'Basic Multilingual Plane')
  - UTF-8
  - UTF-16
  - UTF-32
- ANSEL or ISO/IEC 6937

You will be given this document in appendix of the written exam !

ASCII (7 bits)

Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char	Decimal	Hex	Char
0	0	*[NULL]*	32	20	*[SPACE]*	64	40	**@**	96	60	`
1	1	*[START OF HEADING]*	33	21	**!**	65	41	**A**	97	61	**a**
2	2	*[START OF TEXT]*	34	22	**"**	66	42	**B**	98	62	**b**
3	3	*[END OF TEXT]*	35	23	**#**	67	43	**C**	99	63	**c**
4	4	*[END OF TRANSMISSION]*	36	24	**$**	68	44	**D**	100	64	**d**
5	5	*[ENQUIRY]*	37	25	**%**	69	45	**E**	101	65	**e**
6	6	*[ACKNOWLEDGE]*	38	26	**&**	70	46	**F**	102	66	**f**
7	7	*[BELL]*	39	27	**'**	71	47	**G**	103	67	**g**
8	8	*[BACKSPACE]*	40	28	**(**	72	48	**H**	104	68	**h**
9	9	*[HORIZONTAL TAB]*	41	29	**)**	73	49	**I**	105	69	**i**
10	A	*[LINE FEED]*	42	2A	*****	74	4A	**J**	106	6A	**j**
11	B	*[VERTICAL TAB]*	43	2B	**+**	75	4B	**K**	107	6B	**k**
12	C	*[FORM FEED]*	44	2C	**,**	76	4C	**L**	108	6C	**l**
13	D	*[CARRIAGE RETURN]*	45	2D	**-**	77	4D	**M**	109	6D	**m**
14	E	*[SHIFT OUT]*	46	2E	**.**	78	4E	**N**	110	6E	**n**
15	F	*[SHIFT IN]*	47	2F	**/**	79	4F	**O**	111	6F	**o**
16	10	*[DATA LINK ESCAPE]*	48	30	**0**	80	50	**P**	112	70	**p**
17	11	*[DEVICE CONTROL 1]*	49	31	**1**	81	51	**Q**	113	71	**q**
18	12	*[DEVICE CONTROL 2]*	50	32	**2**	82	52	**R**	114	72	**r**
19	13	*[DEVICE CONTROL 3]*	51	33	**3**	83	53	**S**	115	73	**s**
20	14	*[DEVICE CONTROL 4]*	52	34	**4**	84	54	**T**	116	74	**t**
21	15	*[NEGATIVE ACKNOWLEDGE]*	53	35	**5**	85	55	**U**	117	75	**u**
22	16	*[SYNCHRONOUS IDLE]*	54	36	**6**	86	56	**V**	118	76	**v**
23	17	*[ENG OF TRANS. BLOCK]*	55	37	**7**	87	57	**W**	119	77	**w**
24	18	*[CANCEL]*	56	38	**8**	88	58	**X**	120	78	**x**
25	19	*[END OF MEDIUM]*	57	39	**9**	89	59	**Y**	121	79	**y**
26	1A	*[SUBSTITUTE]*	58	3A	**:**	90	5A	**Z**	122	7A	**z**
27	1B	*[ESCAPE]*	59	3B	**;**	91	5B	**[**	123	7B	**{**
28	1C	*[FILE SEPARATOR]*	60	3C	**<**	92	5C	****	124	7C	**|**
29	1D	*[GROUP SEPARATOR]*	61	3D	**=**	93	5D	**]**	125	7D	**}**
30	1E	*[RECORD SEPARATOR]*	62	3E	**>**	94	5E	**^**	126	7E	**~**
31	1F	*[UNIT SEPARATOR]*	63	3F	**?**	95	5F	**_**	127	7F	*[DEL]*



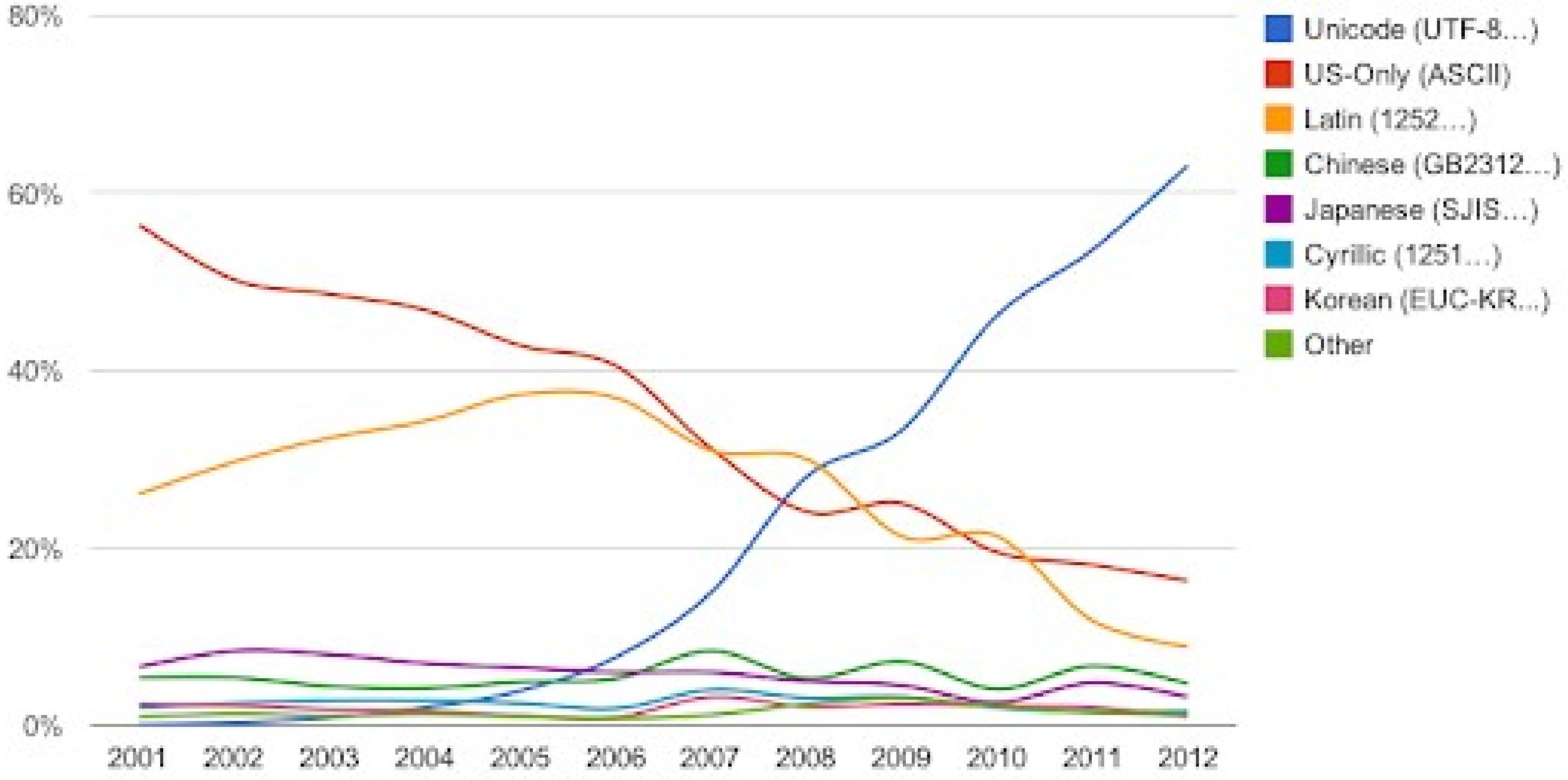
Unicode

- information technology standard for the consistent encoding, representation, and handling of text expressed in most of the world's writing systems
- 144,762 characters covering 159 modern and historic scripts, as well as symbols, emoji, and non-visual control and formatting codes.

https://home.unicode.org/
https://unicode.org/charts/

UTF-8

- Implementation of Unicode on 1-4 bytes (as little as needed)



share of web pages with different encodings

Inserting characters

Inserting Characters

There are many ways character é ("e" with an acute accent, character code 233 (decimal) in Latin-1 and Unicode), can be inserted into a document:

- On Windows, I hold down the Alt key and type 0233 on the numeric keyboard and release the Alt key. I could use the charmap program, too. Or I could copy and paste it (e.g., é). But entering the code directly is risky because, if the character encoding changes, e.g., from Latin-1 to UTF-8, then the meaning of code 233 changes.
- In an HTML document, I can enter these magical incantations, which are displayed correctly regardless of encoding:
    - é (decimal) ⇒ é
    - é (hex) ⇒ é
    - é (mnemonic) ⇒ é

  Note: HTML/XHTML validation programs might not be acquainted with these and complain.
- In Microsoft Word, I type an accent code followed by the accented letter. On Windows, Ctrl+quote, then 'e'. On Mac, Option+quote, then 'e'. Accent codes include: grave=backquote, acute=quote, circumflex=hat, colon=umlaut, comma=cedilla, tilde=tilde, slash=slash, and perhaps others.

http://hcibib.org/multilingual/badchars.htm

Encoding errors

What Could Possibly Go Wrong?

If é is UTF-8 encoded, but displayed without decoding, it looks like this:

Ã©

The first 128 characters in the Latin-1 character set (same as ASCII), are simply represented as themselves in UTF-8. The second half of Latin-1 characters are split. The first half of the non-ASCII Latin-1 characters are represented by themselves, preceded by code 194 decimal or C2 hex, so the UTF-8 encoding for character code 191 (decimal), ¿, is

Â¿

The second half of the non-ASCII Latin-1 characters are represented by a different character, preceded by code 195 decimal or C3 hex. So, when looking at UTF-8 encodings of Latin-1 characters, if you see Â or Ã where you do not expect it, there are probably too many UTF-8 encodings. Multiple extra encodings have a pattern to them:

0 é
1 Ã©
2 ÃƒÂ©
3 ÃƒÂƒÃ‚Â©
4 ÃƒÆ’Ã‚Æ’Ãƒâ€šÃ‚Â©
5 *you get the idea*

Note: If you see boxes in the characters above, it is because the font used is missing that character. There is no way to fix it other than getting a new font or by changing the font. Often, the fonts used in a window title or status bar or JavaScript are more limited than those used elsewhere, so the "alert", "title", and "status" buttons in the Character Conversion Corner can be used to test characters in those contexts.

Too few encodings can have a bad effect that looks different. When é is not UTF-8 encoded, it can appear like this very high numbered character:

�

Progressive under-encoding can result in a question mark being displayed.

http://hcibib.org/multilingual/badchars.htm

Encoding errors

Diagnostic Reference

You are now ready to diagnose UTF-8 encoding problems (e.g., with é):

Symptom	Diagnosis
é	no problems
Ã©	too much UTF-8 encoding, or viewing UTF-8 encoded text with Latin-1 encoding
ÃƒÂ©	much too much UTF-8 encoding
�	too little UTF-8 encoding
?	something bad happened to this character
	wild animals have eaten this character
□	if you see a box, the font in use is missing this character. Firefox 3's boxes contain the hexadecimal value for the missing character, but it's still just a missing character.

http://hcibib.org/multilingual/badchars.htm

Data Interoperability and Semantics

Part 1. Encoding base data types

Part 1.5. Base32 and Base64 encoding

ICM – Toolbox Engineering and Interoperability of Software Systems – Course unit on Data Interoperability and Semantics
M1 Cyber Physical and Social Systems – Course unit on Data Interoperability and Semantics
Maxime Lefrançois https://maxime-lefrancois.info
Course unit URL: https://ci.mines-stetienne.fr/cps2/course/data

Binary to text encoding

- Base64

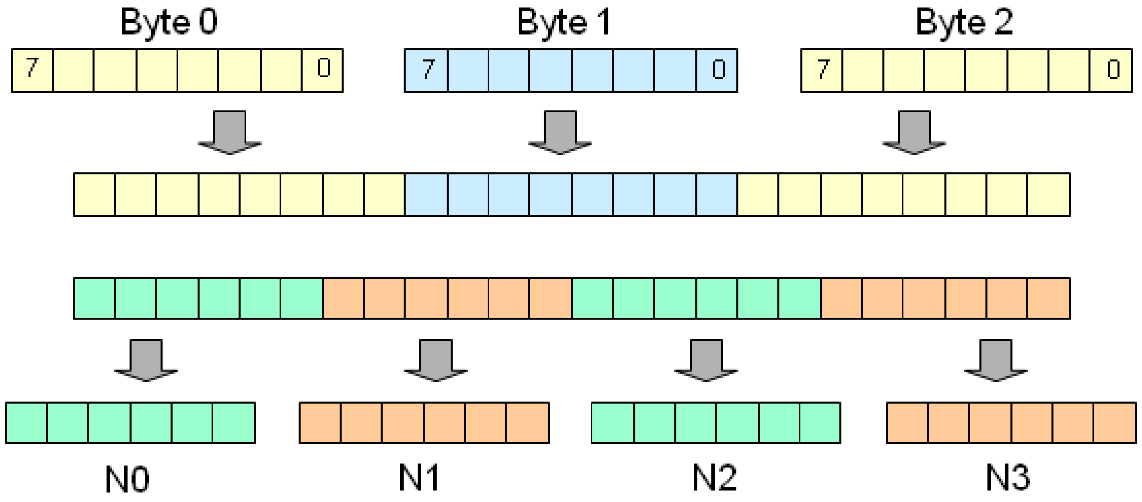


Table 1: The Base 64 Alphabet

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	17	R	34	i	51	z
1	B	18	S	35	j	52	0
2	C	19	T	36	k	53	1
3	D	20	U	37	l	54	2
4	E	21	V	38	m	55	3
5	F	22	W	39	n	56	4
6	G	23	X	40	o	57	5
7	H	24	Y	41	p	58	6
8	I	25	Z	42	q	59	7
9	J	26	a	43	r	60	8
10	K	27	b	44	s	61	9
11	L	28	c	45	t	62	+
12	M	29	d	46	u	63	/
13	N	30	e	47	v		
14	O	31	f	48	w	(pad)	=
15	P	32	g	49	x		
16	Q	33	h	50	y		

Table 2: The "URL and Filename safe" Base 64 Alphabet

62 - (minus)
63 _
(underline)

https://datatracker.ietf.org/doc/html/rfc4648
https://www.base64decode.org/

Binary to text encoding

- Base64

```
The following example of Base64 data is from [5], with corrections.

   Input data:  0x14fb9c03d97e
   Hex:     1   4    f   b    9   c     | 0   3    d   9    7   e
   8-bit:   00010100 11111011 10011100  | 00000011 11011001 01111110
   6-bit:   000101 001111 101110 011100 | 000000 111101 100101 111110
   Decimal: 5      15     46     28       0      61     37     62
   Output:  F      P      u      c        A      9      l      +

   Input data:  0x14fb9c03d9
   Hex:     1   4    f   b    9   c     | 0   3    d   9
   8-bit:   00010100 11111011 10011100  | 00000011 11011001
                                                   pad with 00
   6-bit:   000101 001111 101110 011100 | 000000 111101 100100
   Decimal: 5      15     46     28       0      61     36
                                                      pad with =
   Output:  F      P      u      c        A      9      k      =

   Input data:  0x14fb9c03
   Hex:     1   4    f   b    9   c     | 0   3
   8-bit:   00010100 11111011 10011100  | 00000011
                                          pad with 0000
   6-bit:   000101 001111 101110 011100 | 000000 110000
   Decimal: 5      15     46     28       0      48
                                               pad with =      =
   Output:  F      P      u      c        A      w      =      =
```

https://datatracker.ietf.org/doc/html/rfc4648
https://www.base64decode.org/

Binary to text encoding

- Base64
- Base32
- Base16

Table 3: The Base 32 Alphabet

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	A	9	J	18	S	27	3
1	B	10	K	19	T	28	4
2	C	11	L	20	U	29	5
3	D	12	M	21	V	30	6
4	E	13	N	22	W	31	7
5	F	14	O	23	X		
6	G	15	P	24	Y	(pad)	=
7	H	16	Q	25	Z		
8	I	17	R	26	2		

Table 5: The Base 16 Alphabet

Value	Encoding	Value	Encoding	Value	Encoding	Value	Encoding
0	0	4	4	8	8	12	C
1	1	5	5	9	9	13	D
2	2	6	6	10	A	14	E
3	3	7	7	11	B	15	F

https://datatracker.ietf.org/doc/html/rfc4648
https://www.base64decode.org/

Data Interoperability and Semantics

Part 1. Encoding base data types

Part 1.6. Date and time

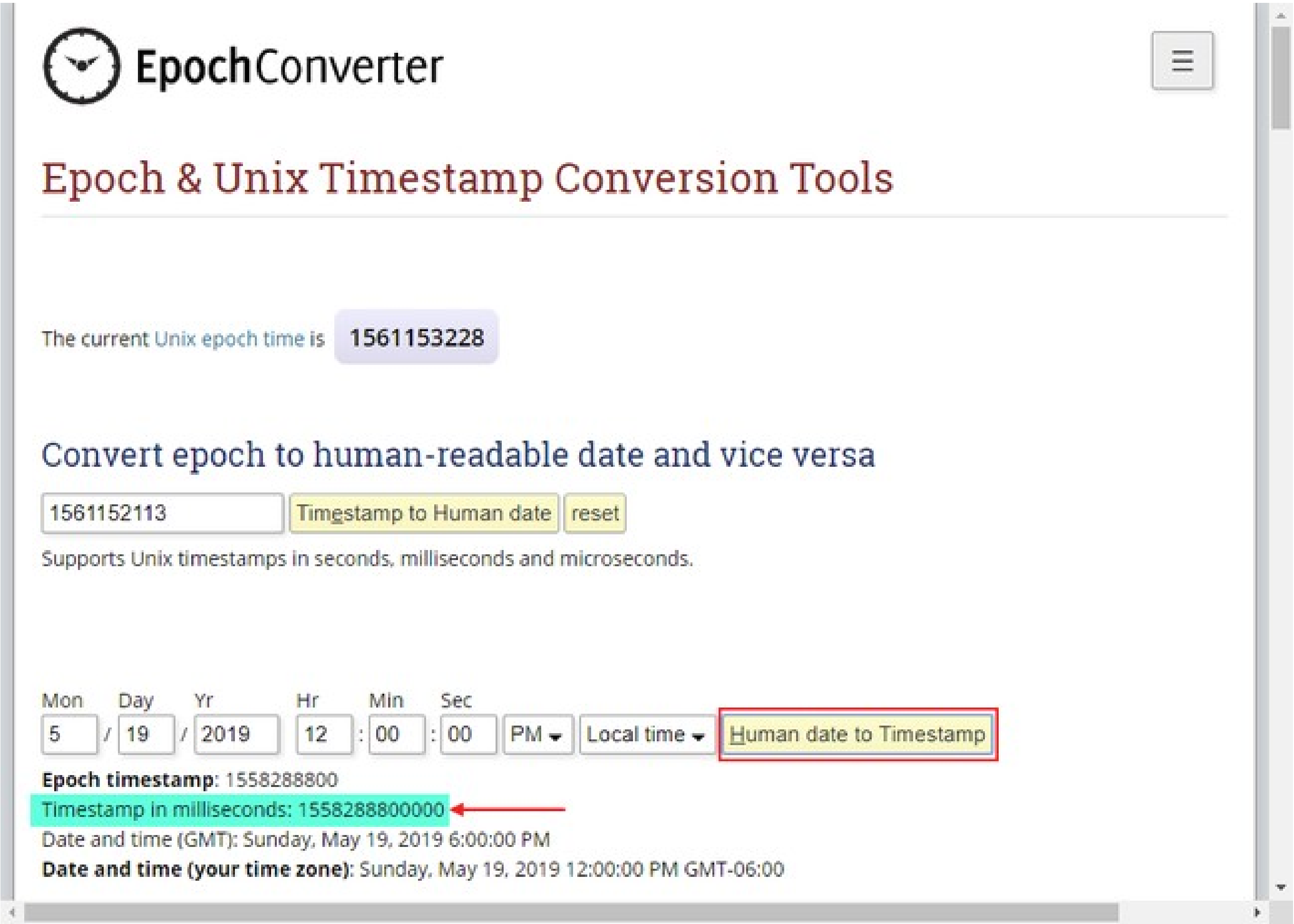
ICM – Toolbox Engineering and Interoperability of Software Systems – Course unit on Data Interoperability and Semantics
M1 Cyber Physical and Social Systems – Course unit on Data Interoperability and Semantics
Maxime Lefrançois https://maxime-lefrancois.info
Course unit URL: https://ci.mines-stetienne.fr/cps2/course/data

Representing Date and Time is not simple

- Leap year / bissextile years
  - *a calendar year that contains an additional day added to keep the calendar year synchronized with the astronomical year or seasonal year.*
- Leap seconds
  - *a one-second adjustment that is occasionally applied to Coordinated Universal Time (UTC), to accommodate the difference between precise time (International Atomic Time (TAI), as measured by atomic clocks) and imprecise observed solar time (UT1), which varies due to irregularities and long-term slowdown in the Earth's rotation*
- Timezones
  - UTC, GMT, CET, CEST, … https://www.timeanddate.com/time/zones/

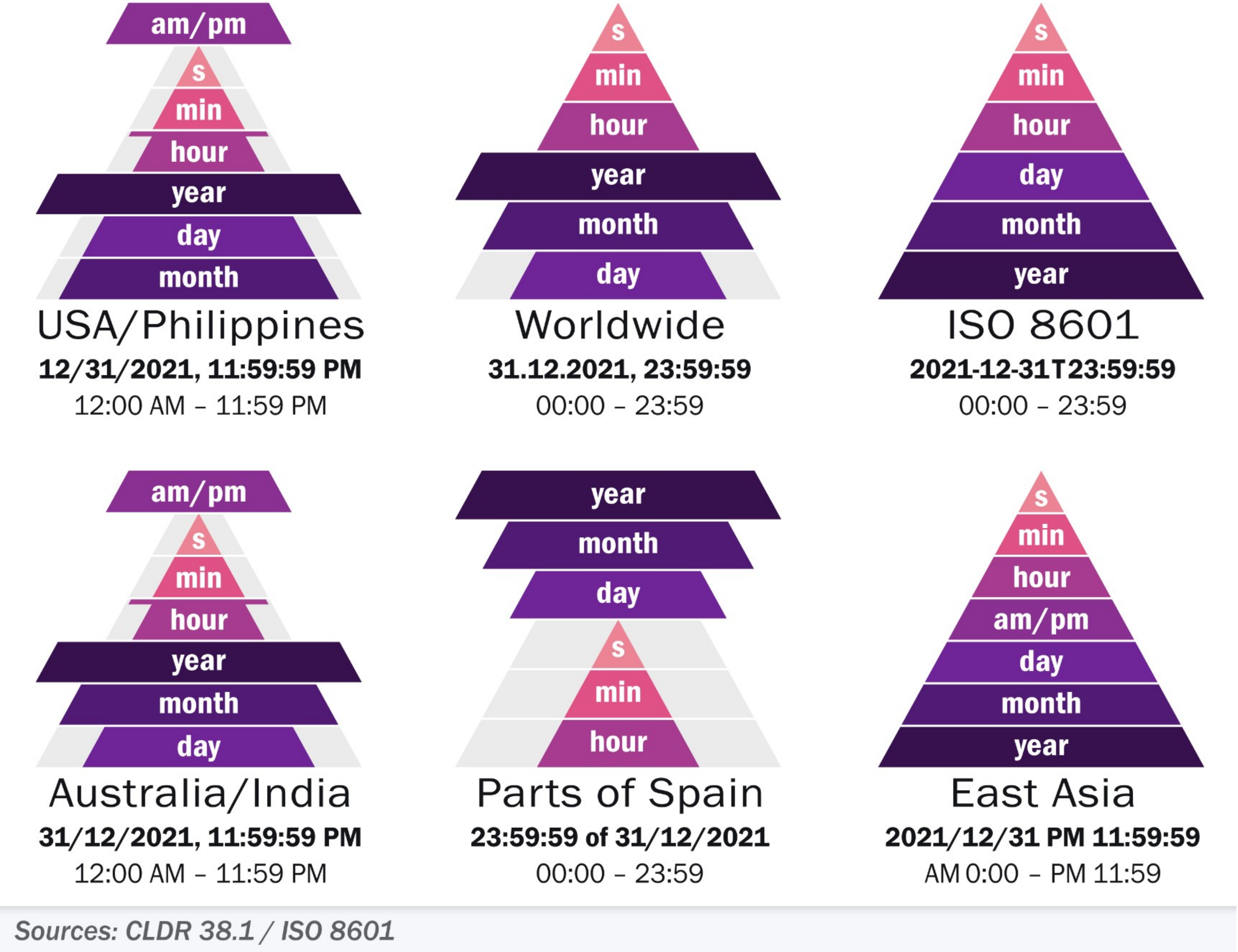
Unix time stamp

The unix time stamp is a way to track time as a running total of seconds. This count starts at the Unix Epoch on January 1st, 1970 at UTC. Therefore, the unix time stamp is merely the number of seconds between a particular date and the Unix Epoch.

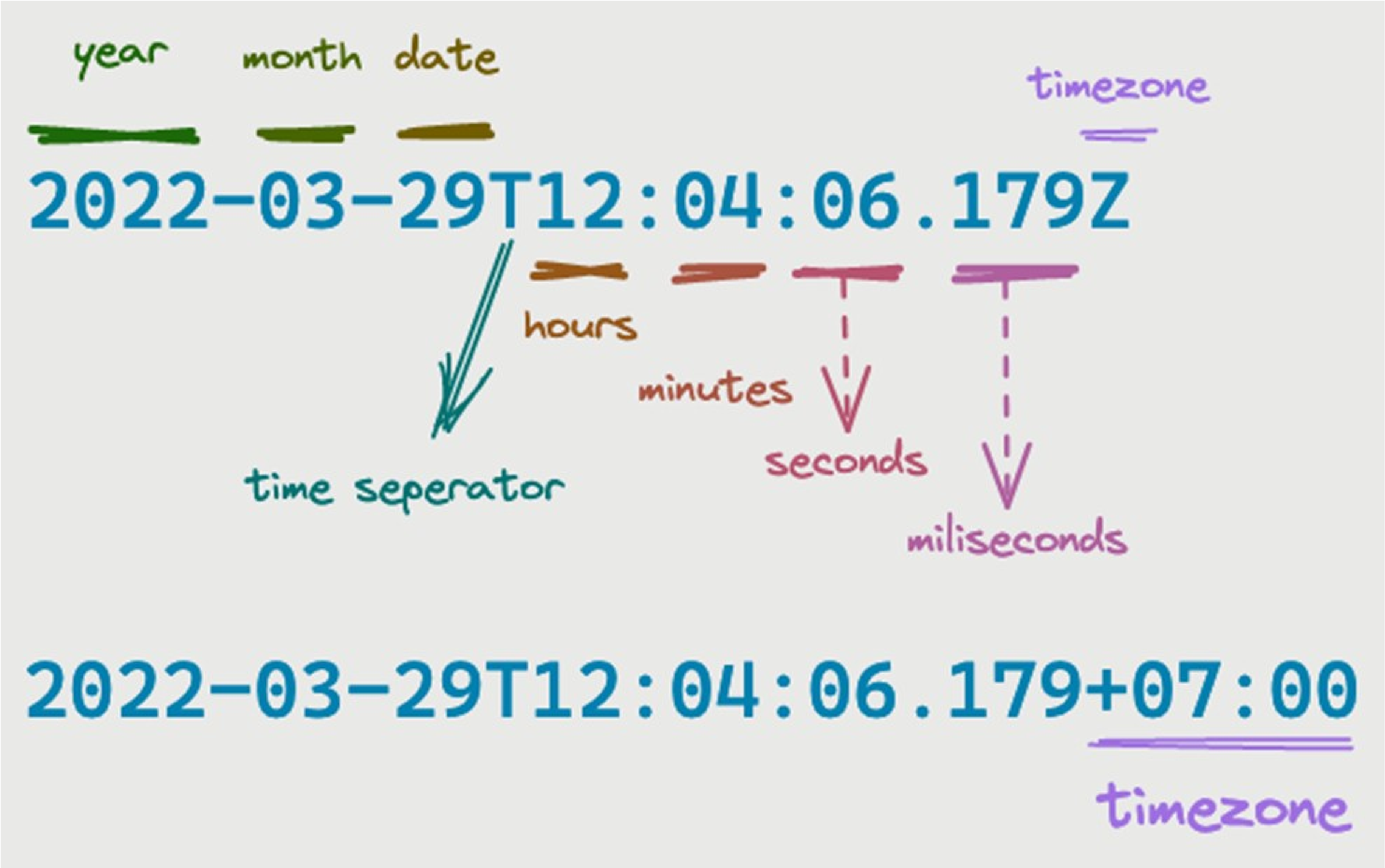


https://www.epochconverter.com/

Too many formats



ISO 8601



https://dev.to/raaynaldo/javascript-datetime-111e

Data Interoperability and Semantics

Part 1. Encoding base data types

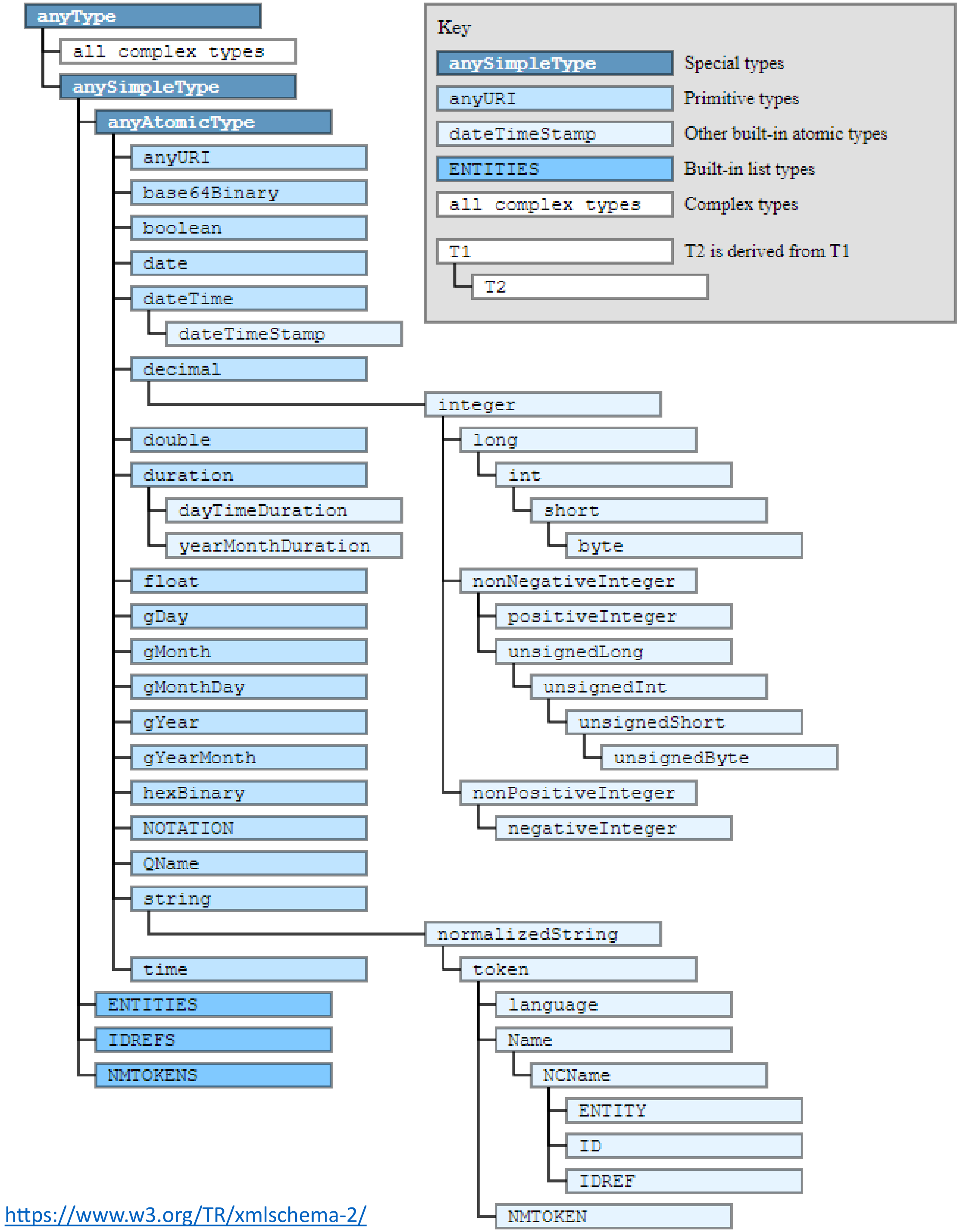
Part 1.7. XML Schema Datatypes

ICM – Toolbox Engineering and Interoperability of Software Systems – Course unit on Data Interoperability and Semantics
M1 Cyber Physical and Social Systems – Course unit on Data Interoperability and Semantics
Maxime Lefrançois https://maxime-lefrancois.info
Course unit URL: https://ci.mines-stetienne.fr/cps2/course/data

XML Schema Datatypes

A **datatype** is denoted by a IRI and has three properties:

- A **value space**, which is a set of values.
- A **lexical space**, which is a set of **literals** used to denote the values.
- A lexical-to-value mapping
- A small collection of *functions, relations, and procedures* associated with the datatype.



https://www.w3.org/TR/xmlschema-2/

Goal: type elements and attributes in XML

```xml
<ROOT>
  <Customers>
    <Customer CustomerName="Arshad Ali" CustomerID="C001">
      <Orders>
        <Order OrderDate="2012-07-04T00:00:00" OrderID="10248">
          <OrderDetail Quantity="5" ProductID="10" />
          <OrderDetail Quantity="12" ProductID="11" />
          <OrderDetail Quantity="10" ProductID="42" />
        </Order>
      </Orders>
      <Address> Address line 1, 2, 3</Address>
    </Customer>
    <Customer CustomerName="Paul Henriot" CustomerID="C002">
      <Orders>
        <Order OrderDate="2011-07-04T00:00:00" OrderID="10245">
          <OrderDetail Quantity="12" ProductID="11" />
          <OrderDetail Quantity="10" ProductID="42" />
        </Order>
      </Orders>
      <Address> Address line 5, 6, 7</Address>
    </Customer>
    <Customer CustomerName="Carlos Gonzlez" CustomerID="C003">
      <Orders>
        <Order OrderDate="2012-08-16T00:00:00" OrderID="10283">
          <OrderDetail Quantity="3" ProductID="72" />
        </Order>
      </Orders>
      <Address> Address line 1, 4, 5</Address>
    </Customer>
  </Customers>
</ROOT>
```

```xml
<xsd:attribute name="OrderID" type="xsd:integer"/>
```

```xml
<xsd:attribute name="OrderDate" type="xsd:dateTime"/>
```

```xml
<xsd:attribute name="CustomerID">
    <xsd:simpleType>
        <xsd:restriction base="xsd:string">
            <xsd:pattern value="/^C[0-9]{3}$/"></xsd:pattern>
        </xsd:restriction>
    </xsd:simpleType>
</xsd:attribute>
```

Anatomy of a XSD literal

```
"24"^^xsd:integer

"true"^^xsd:boolean

"2001-10-26T21:32:52+02:00"^^xsd:string
```

lexical form

datatype IRI

@prefix xsd: <http://www.w3.org/2001/XMLSchema#>.

xsd:int vs xsd:integer

A **xsd:int** represents a signed 32-bit integer

A **xsd:integer** is an integer unbounded value

https://studylib.net/doc/17671880/ieee-754-standard-representation-of-floating-point-number...

xsd:float vs xsd:double vs xsd:decimal

A **xsd:float** is patterned after the IEEE single-precision 32-bit floating point datatype

```
(\+|-)?([0-9]+(\.[0-9]*)?|\.[0-9]+)([Ee](\+|-)?[0-9]+)? |(\+|-)?INF|NaN
```

A **xsd:double** is patterned after the IEEE double-precision 64-bit floating point datatype

```
(\+|-)?([0-9]+(\.[0-9]*)?|\.[0-9]+)([Ee](\+|-)?[0-9]+)? |(\+|-)?INF|NaN
```

A **xsd:integer** represents a subset of the real numbers, which can be represented by decimal numerals

```
(\+|-)?([0-9]+(\.[0-9]*)?|\.[0-9]+)
```

https://studylib.net/doc/17671880/ieee-754-standard-representation-of-floating-point-number...

Data Interoperability and Semantics

Part 1. Encoding base data types

Part 1.8. Codes: countries, languages, …

ICM – Toolbox Engineering and Interoperability of Software Systems – Course unit on Data Interoperability and Semantics
M1 Cyber Physical and Social Systems – Course unit on Data Interoperability and Semantics
Maxime Lefrançois https://maxime-lefrancois.info
Course unit URL: https://ci.mines-stetienne.fr/cps2/course/data

Country codes

ISO 3166-1 – Codes for the representation of names of countries and their subdivisions – Part 1: Country codes

ISO 3166[1]			ISO 3166-1[2]			ISO 3166-2[3]	
Country name[5]	Official state name[6]	Sovereignty[6][7][8]	Alpha-2 code[5]	Alpha-3 code[5]	Numeric code[5]	Subdivision code links[3]	Internet ccTLD[9]
France[l]	The French Republic	UN member state	FR	FRA	250	ISO 3166-2:FR	.fr
United States of America (the)	The United States of America	UN member state	US	USA	840	ISO 3166-2:US	.us
China	The People's Republic of China	UN member state	CN	CHN	156	ISO 3166-2:CN	.cn
Austria	The Republic of Austria	UN member state	AT	AUT	040	ISO 3166-2:AT	.at

Example of country codes

Source: https://en.wikipedia.org/wiki/List_of_ISO_3166_country_codes

Language codes

ISO 639 is a standardized nomenclature used to classify languages. Each language is assigned a two-letter (639-1) and three-letter (639-2 and 639-3) lowercase abbreviation

ISO language name	639-1	639-2/T	639-2/B	639-3	
English	en	eng	eng	eng	
Chinese	zh	zho	**chi**	zho + 16	macrolanguage
Hindi	hi	hin	hin	hin	
Spanish, Castilian	es	spa	spa	spa	
French	fr	fra	**fre**	fra	
Arabic	ar	ara	ara	ara + 29	macrolanguage, Standard Arabic is arb
Bengali	bn	ben	ben	ben	

Example of language names for the most spoken languages

Source: https://en.wikipedia.org/wiki/List_of_ISO_639-1_codes

IETF BCP47: Tags for Identifying Languages

Internet Engineering Task Force (IETF) « Best Current Practice » (BCP)

Region subtags

en-GB

es-005

zh-Hant-HK

Read more in the BCP 47 spec:

2.2.4 Region Subtag

4.1 Choice of Language Tag

HTML Demo: lang

RESET

HTML CSS

```html
<p>This paragraph is English, but the language is not specifically defined.</p>

<p lang="en-GB">This paragraph is defined as British English.</p>

<p lang="fr">Ce paragraphe est défini en français.</p>

```

OUTPUT

This paragraph is English, but the language is not specifically defined.

(In British English) This paragraph is defined as British English.

(In French) Ce paragraphe est défini en français.

https://developer.mozilla.org/en-US/docs/Web/HTML/Global_attributes/lang

Currency codes

ISO 4217 defines alpha codes and numeric codes for the representation of currencies and provides information about the relationships between individual currencies and their minor units.

Active ISO 4217 currency codes[1] [hide]

Code	Num	D[a]	Currency	Locations listed for this currency[b]
AED	784	2	United Arab Emirates dirham	United Arab Emirates
EUR	978	2	Euro	Åland Islands (AX), European Union (EU), Andorra (AD), Austria (AT), Belgium (BE), Cyprus (CY), Estonia (EE), Finland (FI), France (FR), French Southern and Antarctic Lands (TF), Germany (DE), Greece (GR), Guadeloupe (GP), Ireland (IE), Italy (IT), Latvia (LV), Lithuania (LT), Luxembourg (LU), Malta (MT), French Guiana (GF), Martinique (MQ), Mayotte (YT), Monaco (MC), Montenegro (ME), Netherlands (NL), Portugal (PT), Réunion (RE), Saint Barthélemy (BL), Saint Martin (MF), Saint Pierre and Miquelon (PM), San Marino (SM), Slovakia (SK), Slovenia (SI), Spain (ES), Vatican City (VA)
KRW	410	0[c]	South Korean won	South Korea
GBP	826	2	Pound sterling	United Kingdom, Isle of Man (IM, see Manx pound), Jersey (JE, see Jersey pound), Guernsey (GG, see Guernsey pound), Tristan da Cunha (SH-TA)
CHF	756	2	Swiss franc	Switzerland, Liechtenstein (LI)

Example of currency codes

Source: https://en.wikipedia.org/wiki/ISO_4217

Data Interoperability and Semantics

Part 1. Encoding base data types

Part 1.9. Quantities and Units of measure

ICM – Toolbox Engineering and Interoperability of Software Systems – Course unit on Data Interoperability and Semantics
M1 Cyber Physical and Social Systems – Course unit on Data Interoperability and Semantics
Maxime Lefrançois https://maxime-lefrancois.info
Course unit URL: https://ci.mines-stetienne.fr/cps2/course/data

Units of measures: no consensus

- BIPM (International Bureau of Weights and Measures)
- ISO/IEC 1000 – ISO/IEC 80000
- VIM (international Vocabulary for Measurements)
- UnitsML
- UCUM (Unified Code for Units of Measure)
- UNECE Recommendation 20
- Sweet
- QUDT
- …

https://en.wikipedia.org/wiki/Category:Encodings

UCUM: Unified Code for Units of Measure

- A code system intended to include *all units* of measures being contemporarily used in international science, engineering, and business.
- **Used** by international organizations and standards
- **No ambiguity** possible
- **Clear semantics** of units
- **Con: Problematic** custom license

`([pi]/2).rad.kK4`

exponent

simple units

UCUM Code of Unit Concept	EN Unit	EN Symbol	EN Dimension[a]	NCI Concept Code	NCI Term	NCI Abbreviation	SNOMED CT Identifier[b]
[IU]			[arb]	C70497	Anti-Xa Activity International Unit	anti-Xa activity	258997004
Bq	becquerel	Bq	T^{-1}	C42562	Becquerel	Bq	282141004
Bq/g			$M^{-1}T^{-1}$	C70522	Becquerel per gram	Bq/g	
10^9.[CFU]			[arb]	C68897	Billion Colony Forming Units	Billion CFU	
10^9			1	C71189	Billion Organisms		
m3	cubic metre	m^3	L^3	C42570	Cubic Meter	m^3	396154006
Ci/ml			$L^{-3}T^{-1}$	C71172	Curie per Millilitre	Ci/ml	
d	day	d	T	C25301	Day	d	258703001
[drp]			L^3	C48491	Drop Dosing Unit	Gtt	404218003
[IU]/ml			[arb]	C67377	International Unit per Millilitre	IU/mL	259002007
k[USP'U]			[arb]	C71202	Kilo United States Pharmacopoeia Unit	KUSP'U	
kBq/l			$L^{-3}T^{-1}$	C71167	Kilobecquerel per Liter	kBq/L	
mmol/l			$L^{-3}N$	C64387	Millimole per Liter	mmol/L	258813002
[ppm]	part per million	ppm	1	C48523	Part Per Million	ppm	258731005
Pa	pascal	Pa	$L^{-1}MT^{-2}$	C42547	Pascal	P	259016002
%	per cent	%	1	C48570	Percent	%	118582008
%			1	C48571	Percent Volume per Volume	%V/V	419569009
g/ml	per cent (w/v)	%(w/v)	$L^{-3}M$	C48527	Percent Weight Volume	%M/V	396169007
%			1	C48528	Percent Weight Weight	%W/W	118582008
[PFU]			[arb]	C73575	Plaque Forming Unit Equivalent 1000 Mouse LD50	PFU Equivalent 1000 Mouse LD50	
[lb_av]	pound	lb	M	C48531	Pound	LB	258693003
/min	revolution per minute	r.p.m., rev/min, r/min	T^{-1}	C70469	Revolution per Minute	rpm	286549009
[tb'U]			[arb]	C65132	Tuberculin Unit		415758003
[arb'U]{ELISA}				C68875	Enzyme-Linked Immunosorbent Assay Unit	EL. U	

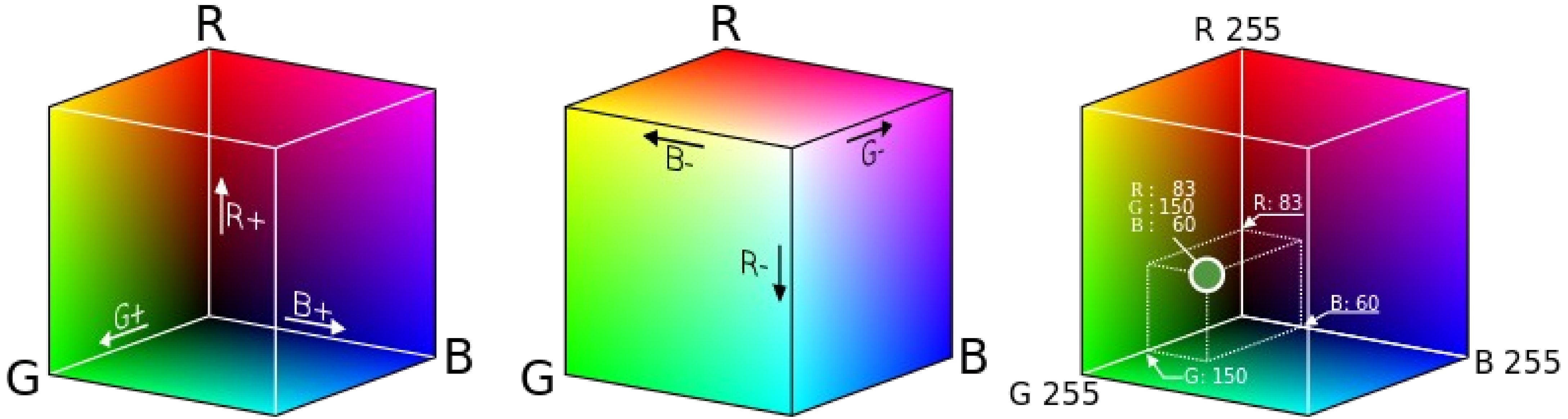
Data Interoperability and Semantics

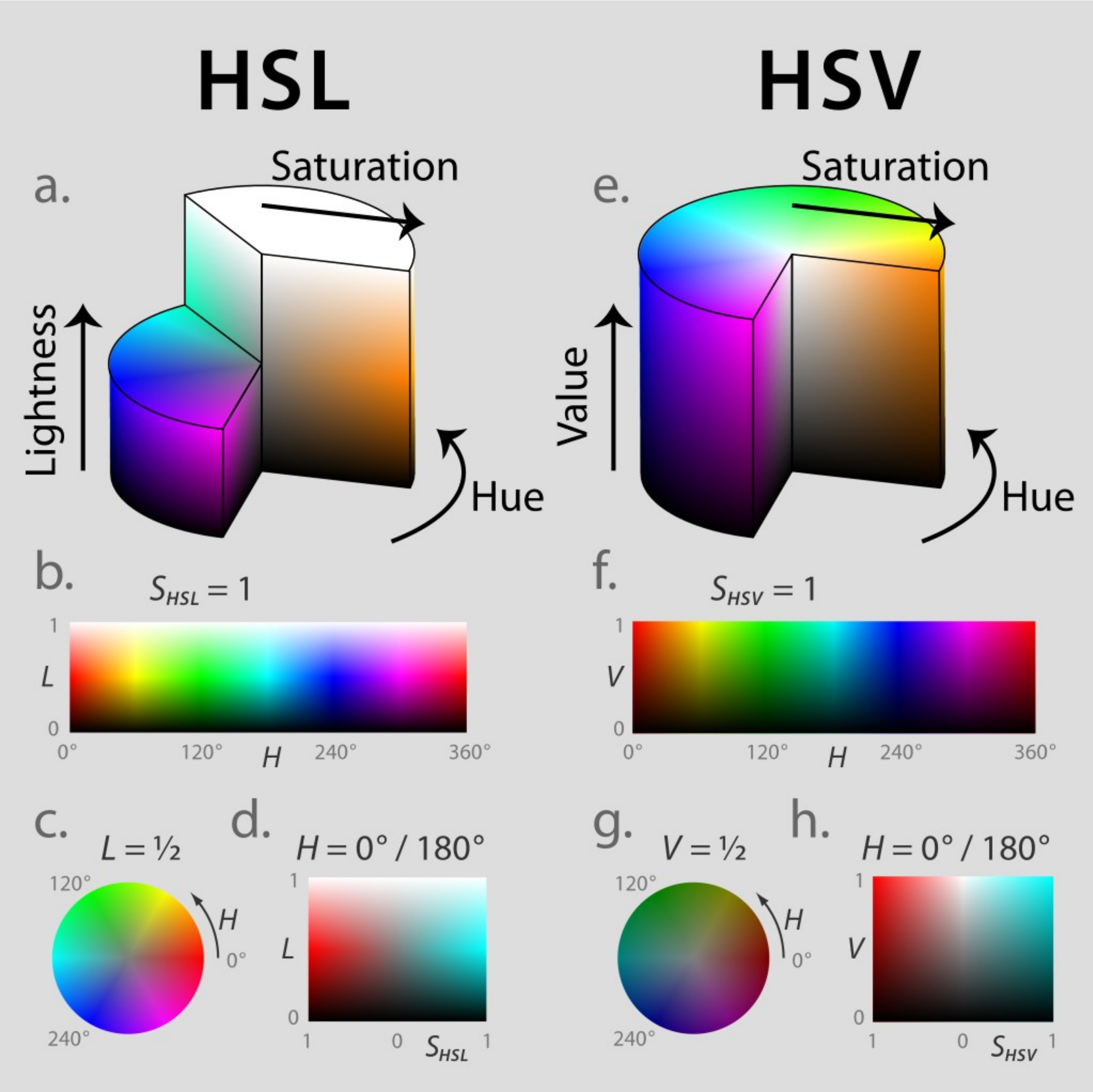
Part 1. Encoding base data types

Part 1.10. Colors

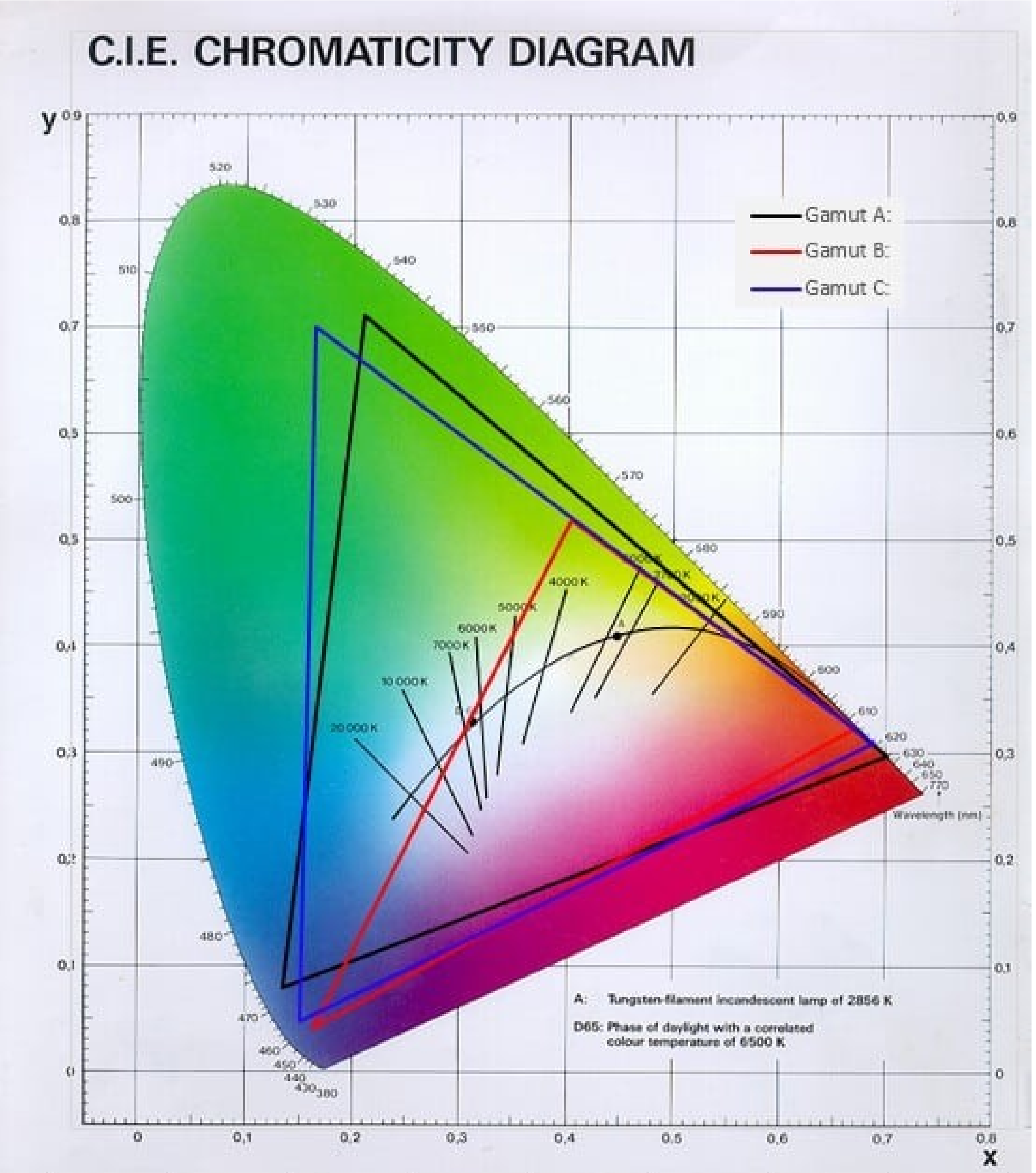
ICM – Toolbox Engineering and Interoperability of Software Systems – Course unit on Data Interoperability and Semantics
M1 Cyber Physical and Social Systems – Course unit on Data Interoperability and Semantics
Maxime Lefrançois https://maxime-lefrancois.info
Course unit URL: https://ci.mines-stetienne.fr/cps2/course/data

RGB color cube





HSL (for hue, saturation, lightness)
HSV (for hue, saturation, value; also known as HSB, for hue, saturation, brightness)
https://en.wikipedia.org/wiki/HSL_and_HSV

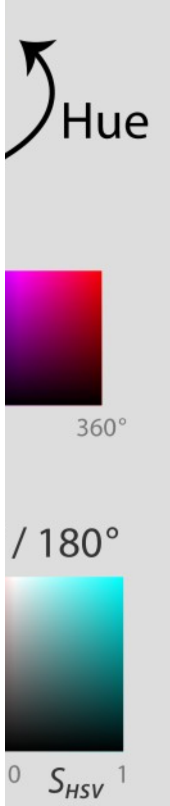


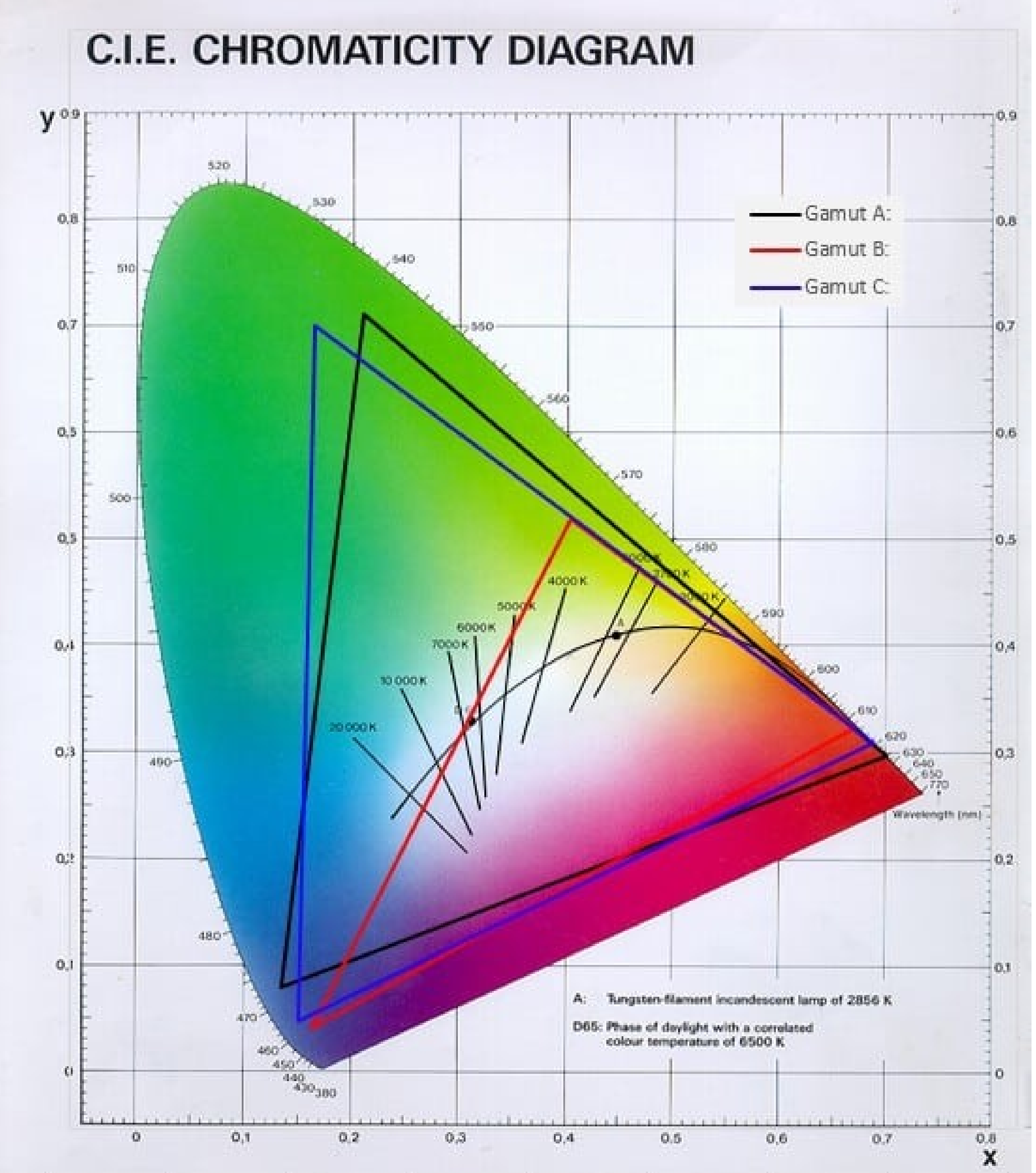
Philips hue color space

HSL HSV


```
{
  "1": {
    "state": {
      "on": true,
      "bri": 254,
      "hue": 14314,
      "sat": 172,
      "effect": "none",
      "xy": [
        0.4791,
        0.4139
      ],
      "ct": 405,
      "alert": "none",
      "colormode": "ct",
      "reachable": true
    },
    "type": "Extended color light",
    "name": "Hue color light 1",
    "modelid": "LCT001",
    "manufacturername": "Philips",
    "uniqueid": "00:17:88:01:00:ff:9a:28-0b",
    "swversion": "5.127.1.26581"
  }
}
```


PHILIPS hue personal wireless lighting





Philips hue color space

HSV ⟷ RGB conversion

When $0 \le H < 360$, $0 \le S \le 1$ and $0 \le V \le 1$:

$$C = V \times S$$

$$X = C \times (1 - |(H / 60°) \bmod 2 - 1|)$$

$$m = V - C$$

$$(R', G', B') = \begin{cases} (C, X, 0) & , 0° \le H < 60° \\ (X, C, 0) & , 60° \le H < 120° \\ (0, C, X) & , 120° \le H < 180° \\ (0, X, C) & , 180° \le H < 240° \\ (X, 0, C) & , 240° \le H < 300° \\ (C, 0, X) & , 300° \le H < 360° \end{cases}$$

$$(R,G,B) = ((R'+m)\times 255, (G'+m)\times 255, (B'+m)\times 255)$$

R’ = R / 255
G’ = G / 255
B’ = B / 255

Cmax = max(R’, G’, B’)
Cmin = min(R’, G’, B)
Δ = Cmax – Cmin

$$H = \begin{cases} 0°, \ \Delta = 0 \\ 60° \times \left(\frac{G'-B'}{\Delta} (mod\ 6)\right), \ Cmax = R' \\ 60° \times \left(\frac{B'-R'}{\Delta} + 2\right), \ Cmax = G' \\ 60° \times \left(\frac{R'-G'}{\Delta} + 4\right), \ Cmax = B' \end{cases}$$

$$S = \begin{cases} 0, \ Cmax = 0 \\ \frac{\Delta}{Cmax}, \ Cmax \neq 0 \end{cases}$$

V = Cmax

Data Interoperability and Semantics

</ Part 1. Encoding base data types >

ICM – Toolbox Engineering and Interoperability of Software Systems – Course unit on Data Interoperability and Semantics
M1 Cyber Physical and Social Systems – Course unit on Data Interoperability and Semantics
Maxime Lefrançois https://maxime-lefrancois.info
Course unit URL: https://ci.mines-stetienne.fr/cps2/course/data