ISEN
ALL IS DIGITAL!
OUEST
yncréa

Embedded Programming

Cortex M : STM32L152RE

jean-jacques.meneu@isen-ouest.yncrea.fr

ISEN ALL IS DIGITAL! OUEST | yncréa

Agenda

1. Introduction to microcontrollers ecosystem, Memory, binary and hex numbers, basic operations (OR, AND, XOR)
2. Numbers representation (2's complement), shift operations, basic architecture
3. ALU, basic instructions, registers
4. Assembly language
5. GPIOs, basics in electronics (push-pull/Open Drain), polling vs interrupt, mapping of peripherals
6. Peripherals: timers, PWM, purpose of PWM
7. Basics of C programming
8. Standard communication (I2C/SPI/UART), ADC

ISEN ALL IS DIGITAL! OUEST | yncréa

PART 1: Embedded Programming STM32

Introduction to microcontrollers ecosystem, Memory, binary and hex numbers, basic operations (OR, AND, XOR)

What is a microcontroller

A microcontroller is a computer. All computers – whether you are talking about a personal desktop computer or a large mainframe computer or a microcontroller- have several things in common

- ✓ CPU
- ✓ Memory
- ✓ Input and output communication

What is a microcontroller

Another way to see a microcontroller

- ✓ A small computer
- ✓ A Single IC
- ✓ Processor core, memory and input/output peripherals

MicroControllers are everywhere

- Industrial
  - PLC
  - Inverters
  - Printers, scanners
  - Industrial networking
  - Solar inverters

- Medical
  - Glucose meters
  - Portable medical care
  - VPAP, CPAP
  - Patient monitoring

- Buildings and security
  - Alarm systems
  - Access control
  - HVAC
  - Power meters

- Appliances
  - 3-phase motor drives
  - Application control
  - User interfaces
  - Induction cooking

- Consumer
  - Home audio
  - Gaming
  - PC peripherals
  - Digital cameras, GPS

life.augmented

Common Characteristics of µControllers

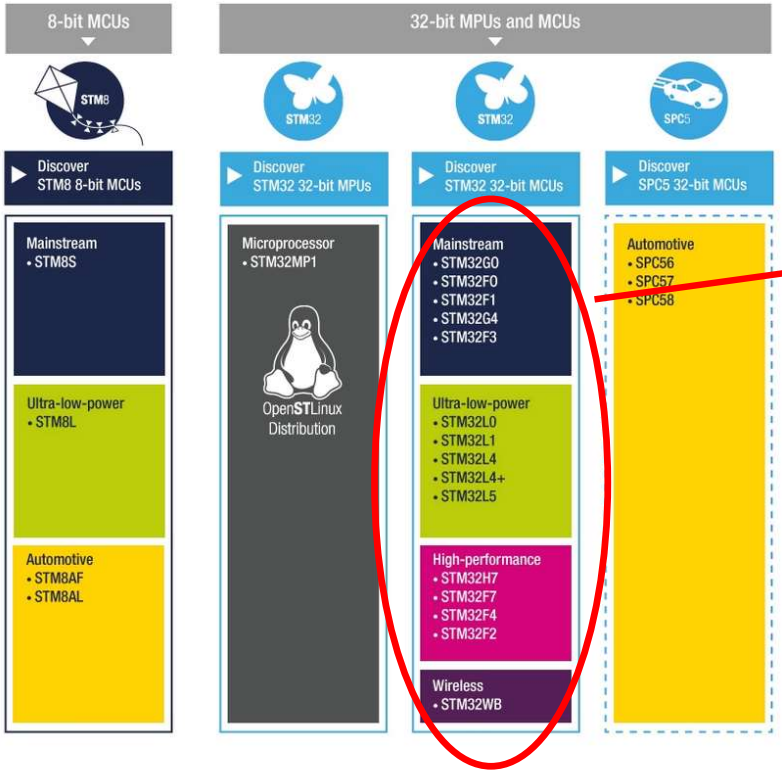
- ✓ µControllers are dedicated
- ✓ µControllers are low-power devices
- ✓ µControllers often take an input to perform an action
- ✓ µControllers are often small and low cost (from 10 cents to 10USD)
- ✓ µControllers are often ruggedized (temperature for instance)

Cost Constraints

Engineers not only solve a technical constraint. They ALWAYS have a trade-off solution/cost

- ✓ If program is not optimized and you request a µController 10 cents more expensive…
- ✓ In automotive application where in average, a car manufacturer orders around 5 million pieces of a given µController / year
- ✓ Do you think you employer will spend 500k$ because your code is not optimized?
- ✓ … and a car can have easily 1,000 µController/car

ST Microcontrollers Portfolio

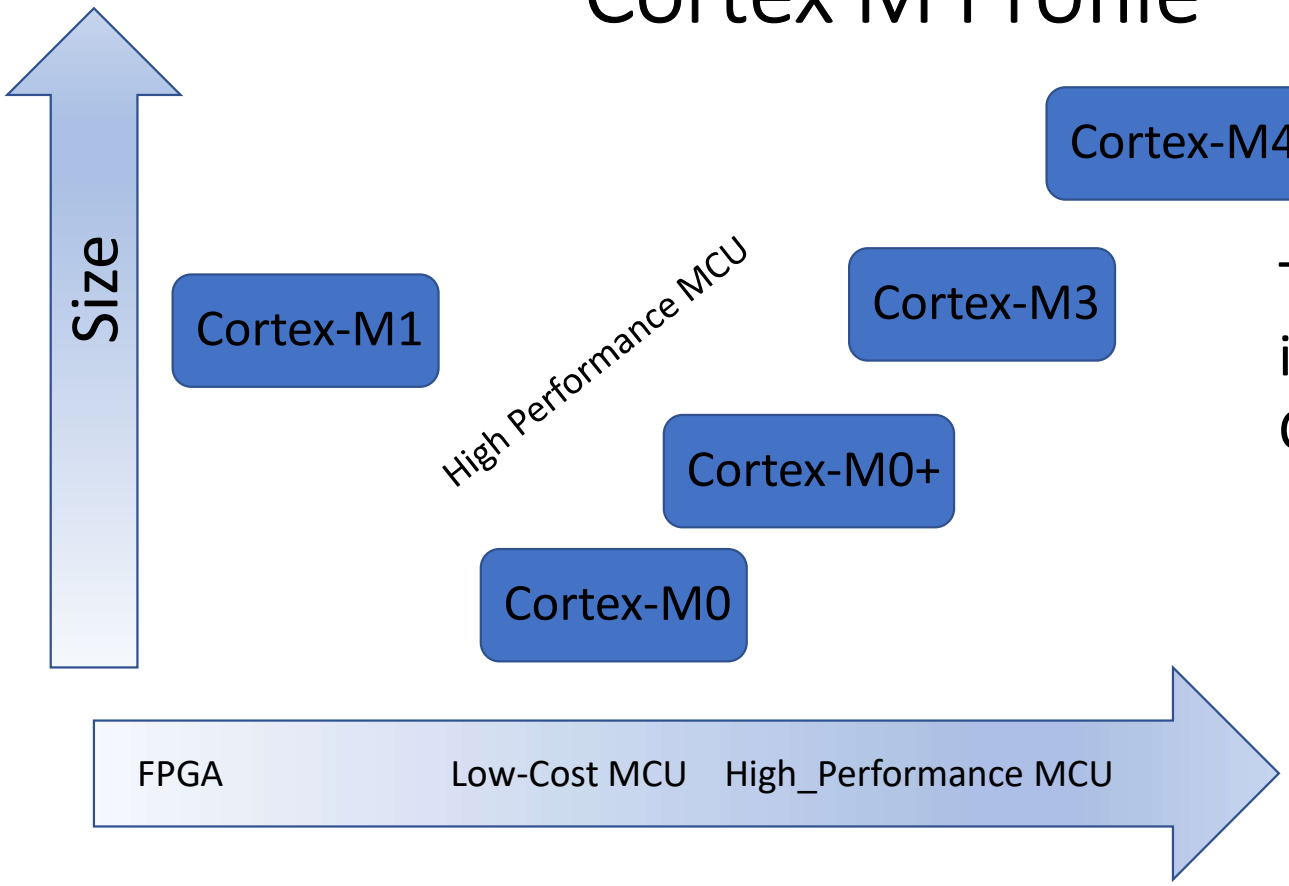


Microcontrollers based on Cortex M

Various Processor Cores

Company	Core Name	Data Bus Size	Comments
Atmel	AVR	8-bit	Used for Arduino
Microchip	PIC	8-bit	Bought Atmel
Renesas		8-bit	
STMicroelectronics	STM8	8-bit	
TI	MSP	16-bit	
ARM	Cortex	32-bit	Licence core, not a silicon vendor

Cortex M Profile



The STM32L152 is based on a Cortex M3 core

Comparison Clock Cycles

Table 1.1: Number of Cycles Taken for a 16 × 16 Multiply against Typical 8- and 16-bit Architectures

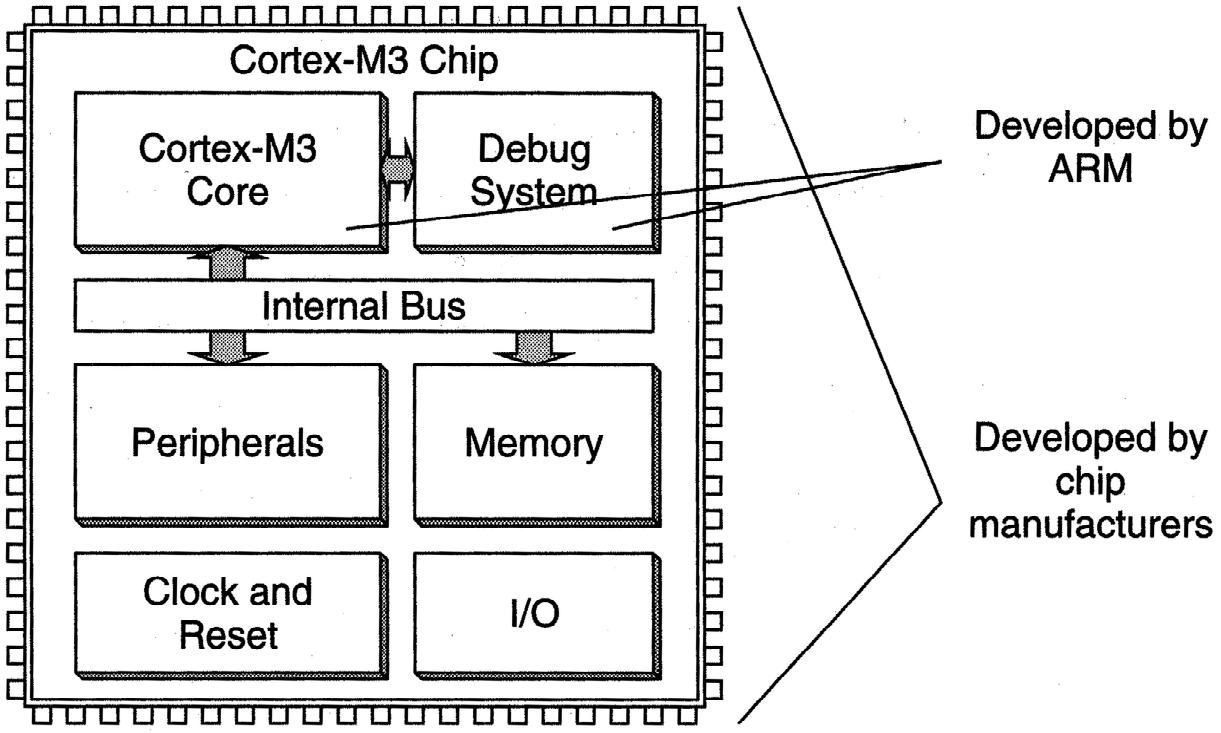
8-Bit Example (8051)	16-Bit Example	ARM Cortex-M

MOV A, XL ; 2 bytes
MOV B, YL ; 3 bytes
MUL AB; 1 byte
MOV R0, A; 1 byte
MOV R1, B; 3 bytes
MOV A, XL ; 2 bytes
MOV B, YH ; 3 bytes
MUL AB; 1 byte
ADD A, R1; 1 byte
MOV R1, A; 1 byte
MOV A, B ; 2 bytes
ADDC A, #0 ; 2 bytes
MOV R2, A; 1 byte
MOV A, XH ; 2 bytes
MOV B, YL ; 3 bytes
MUL AB; 1 byte
ADD A, R1; 1 byte
MOV R1, A; 1 byte
MOV A, B ; 2 bytes
ADDC A, R2 ; 1 bytes
MOV R2, A; 1 byte
MOV A, XH ; 2 bytes
MOV B, YH ; 3 bytes
MUL AB; 1 byte
ADD A, R2; 1 byte
MOV R2, A; 1 byte
MOV A, B ; 2 bytes
ADDC A, #0 ; 2 bytes
MOV R3, A; 1 byte
	MOV R1,&MulOp1
	MOV R2,&MulOp2
	MOV SumLo,R3
	MOV SumHi,R4
	
	(Memory mapped multiply unit)
		MULS r0,r1,r0

Time: 48 clock cycles* **Code size:** 48 bytes	**Time:** 8 clock cycles **Code size:** 8 bytes	**Time:** 1 clock cycle **Code size:** 2 bytes

*cycle count for a single cycle 8051 processor.

Cortex M is licenced by ARM



Microcontrollers Makers

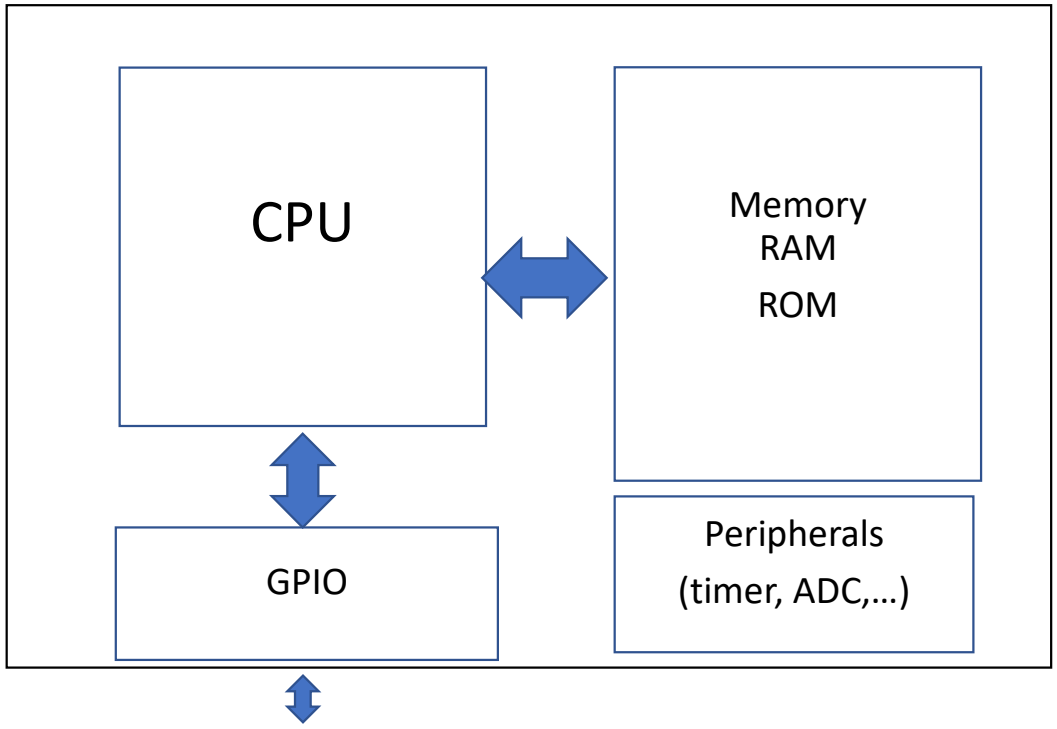
TOP 7

- STMicroelectronics, French-italian
- Cypress semiconductor (bought by Infineon), USA
- Infineon technologies ,Germany
- NXP, Germany
- Renesas, USA
- TI, USA
- Microchip

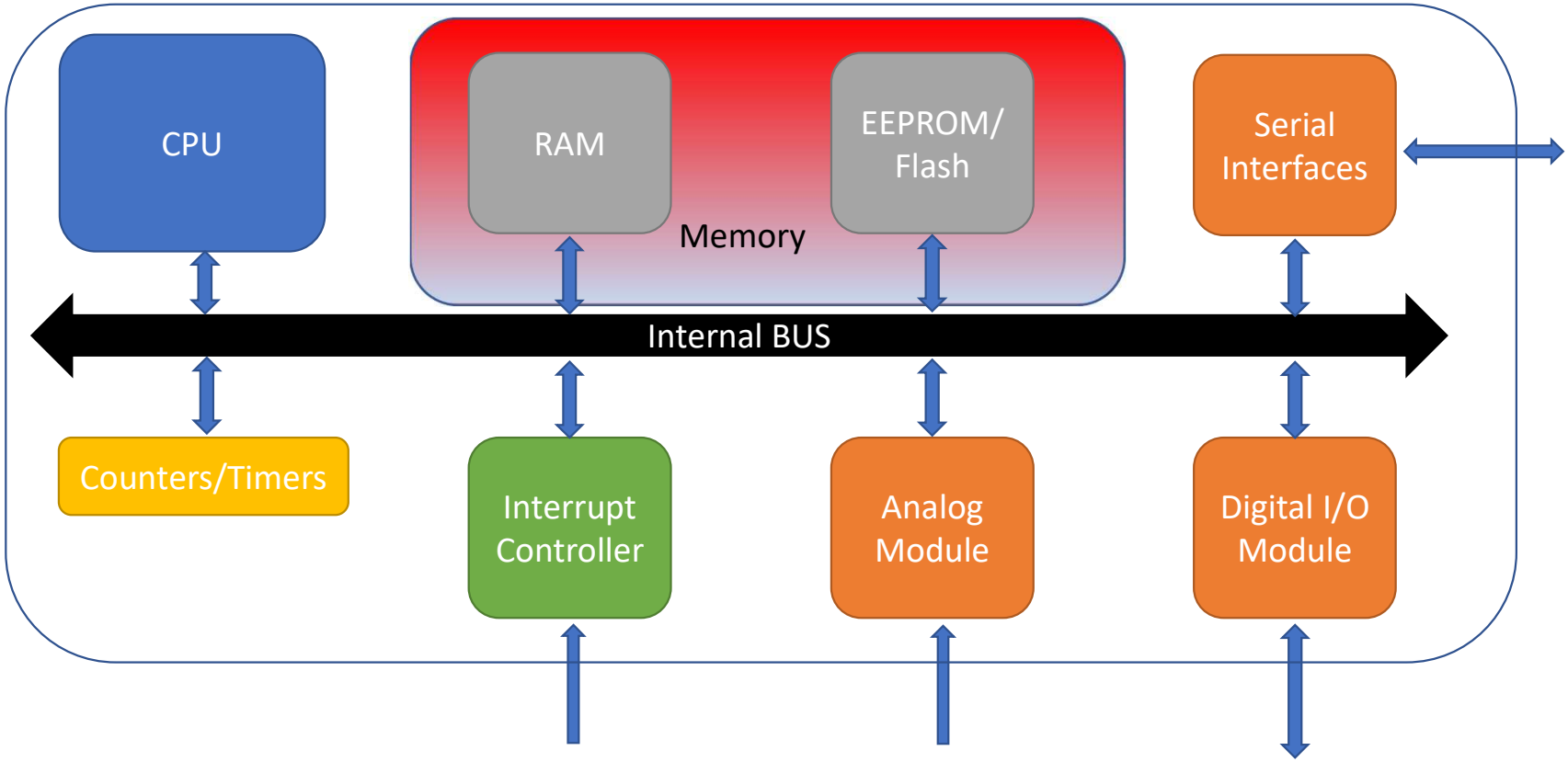
Microcontrollers Maket

Large Ecosystem around µControllers

µController in One Shot



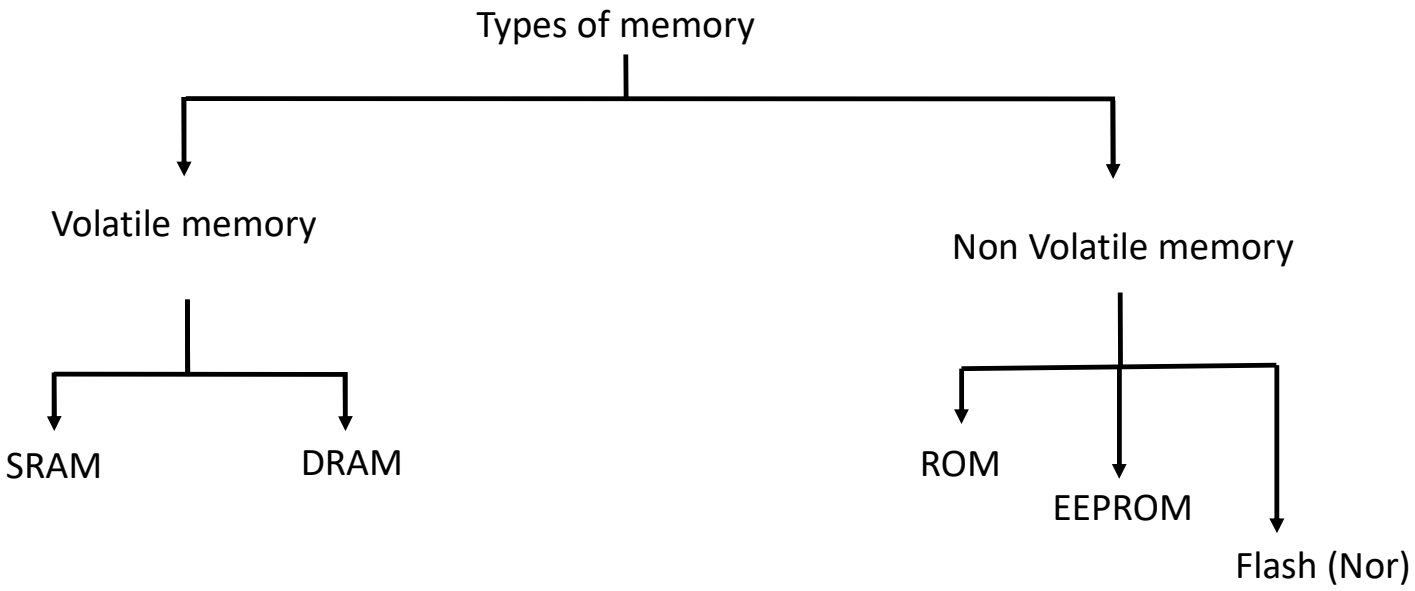
µController Global Architecture



Memory Parameters

- ✓ Volatility (able to keep data if Vcc is On or Off)
- ✓ Granularity in read and write cycle
- ✓ Endurance
- ✓ Access time to read and write
- ✓ Size / Cost

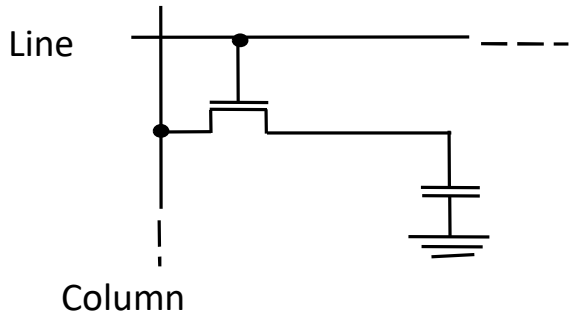
Memory types



DRAM Characteristics

DRAM stands for Dynamic Random Access Memory

- ✓ Volatile
- ✓ 8-bit granularity in read and write cycle
- ✓ Unlimited Endurance
- ✓ Fast access time to read and write
- ✓ Large Size / Low Cost
- ✓ High Current Consumption (must be refreshed)

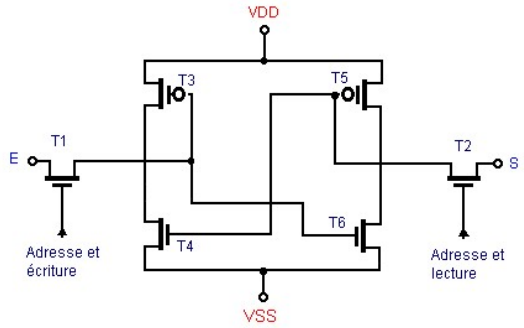


Not Used in µControllers

SRAM Characteristics

SRAM stands for Static Random Access Memory

- ✓ Volatile
- ✓ 8-bit granularity in read and write cycle
- ✓ Unlimited Endurance
- ✓ Fast access time to read and write
- ✓ Large Size / High Cost
- ✓ Low Current Consumption

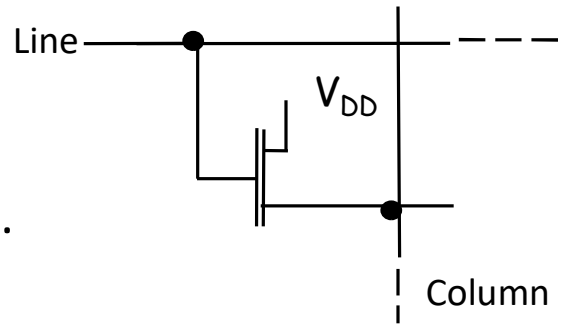


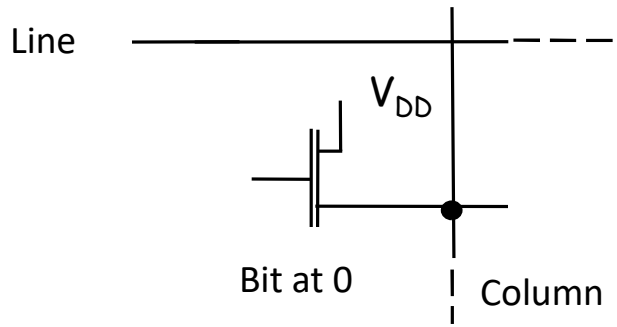
Always Used in µControllers

ROM Characteristics

ROM stands for Read Only Memory

- ✓ Non-Volatile
- ✓ 8-bit granularity in read cycle.
- ✓ Unlimited Endurance
- ✓ Fast access time to read
- ✓ Large Size / Low Cost
- ✓ Impossible to Write



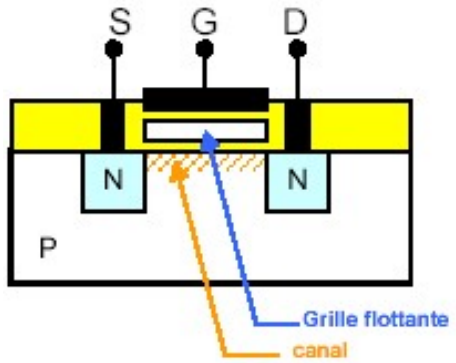


Specific Use in µControllers for final Mask

EEPROM Characteristics

EEPROM stands for electrically Erasable Programmable Read Only Memory

- ✓ Non-Volatile
- ✓ 8-bit granularity in read / write cycle.
- ✓ limited Endurance in write cycle
- ✓ Slow access time to read / write cycle
- ✓ Medium Size / Low Cost

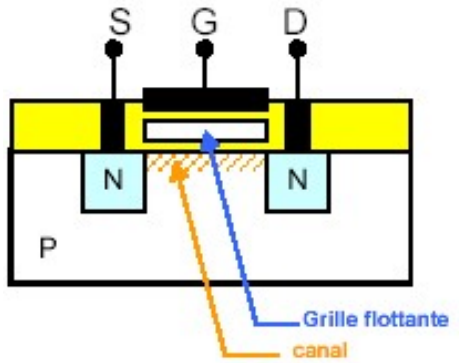


Often used in µControllers

FLASH (NOR) Characteristics

EEPROM stands for electrically Erasable Programmable Read Only Memory

- ✓ Non-Volatile
- ✓ 8-bit granularity in read cycle. Must write in pages (block of bytes)
- ✓ limited Endurance in write cycle
- ✓ Very Slow access time in write cycle
- ✓ Medium Size / Lower Cost than EEPROM



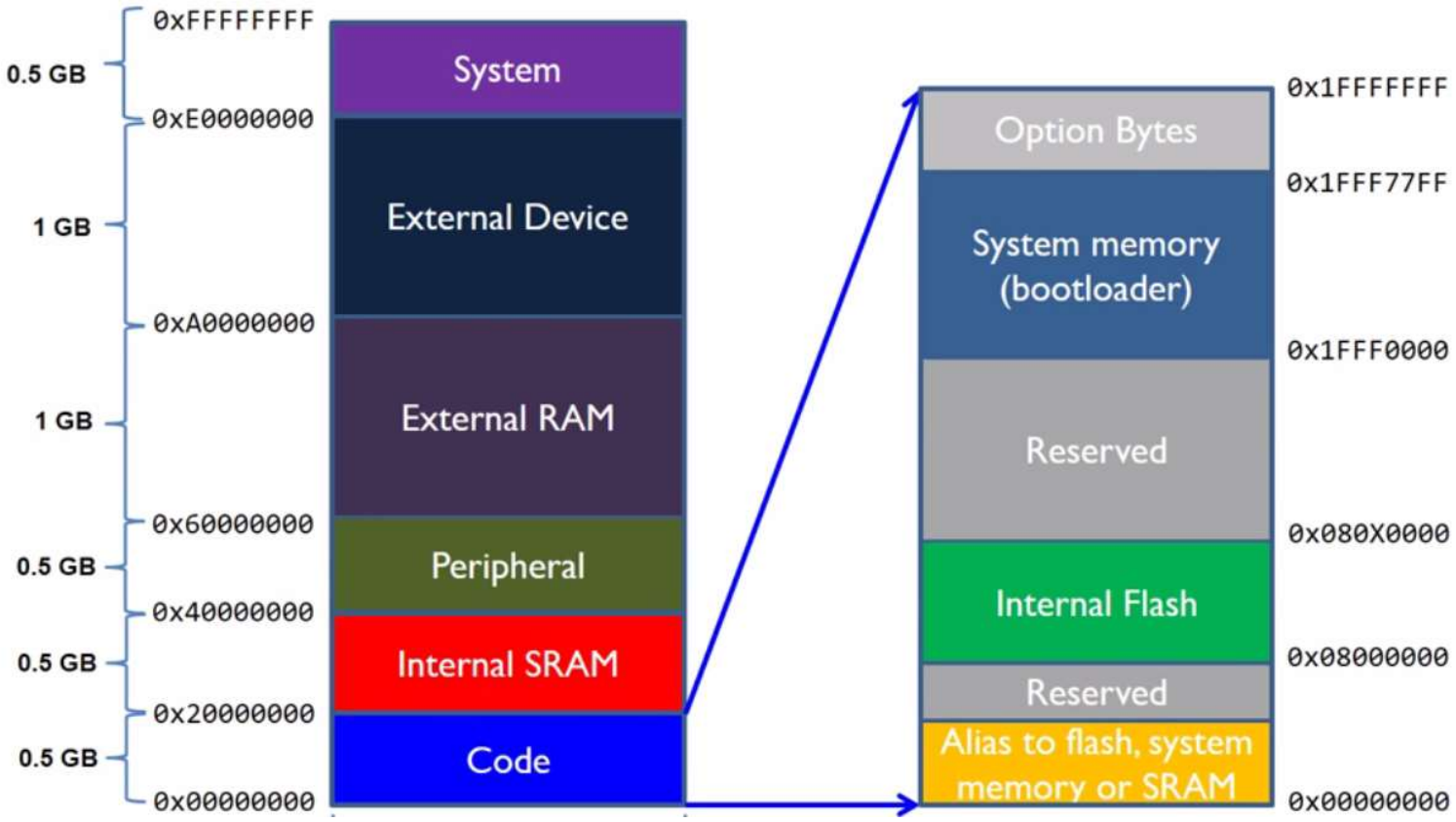
Always used in µControllers

Other Memory Types

- ✓ Nand Flash
- ✓ SDRAM
- ✓ PSRAM
- ✓ MRAM
- ✓ FRAM

These memories are not used in Microcontroller application (a little bit for FRAM)

STM32 Memory Organization



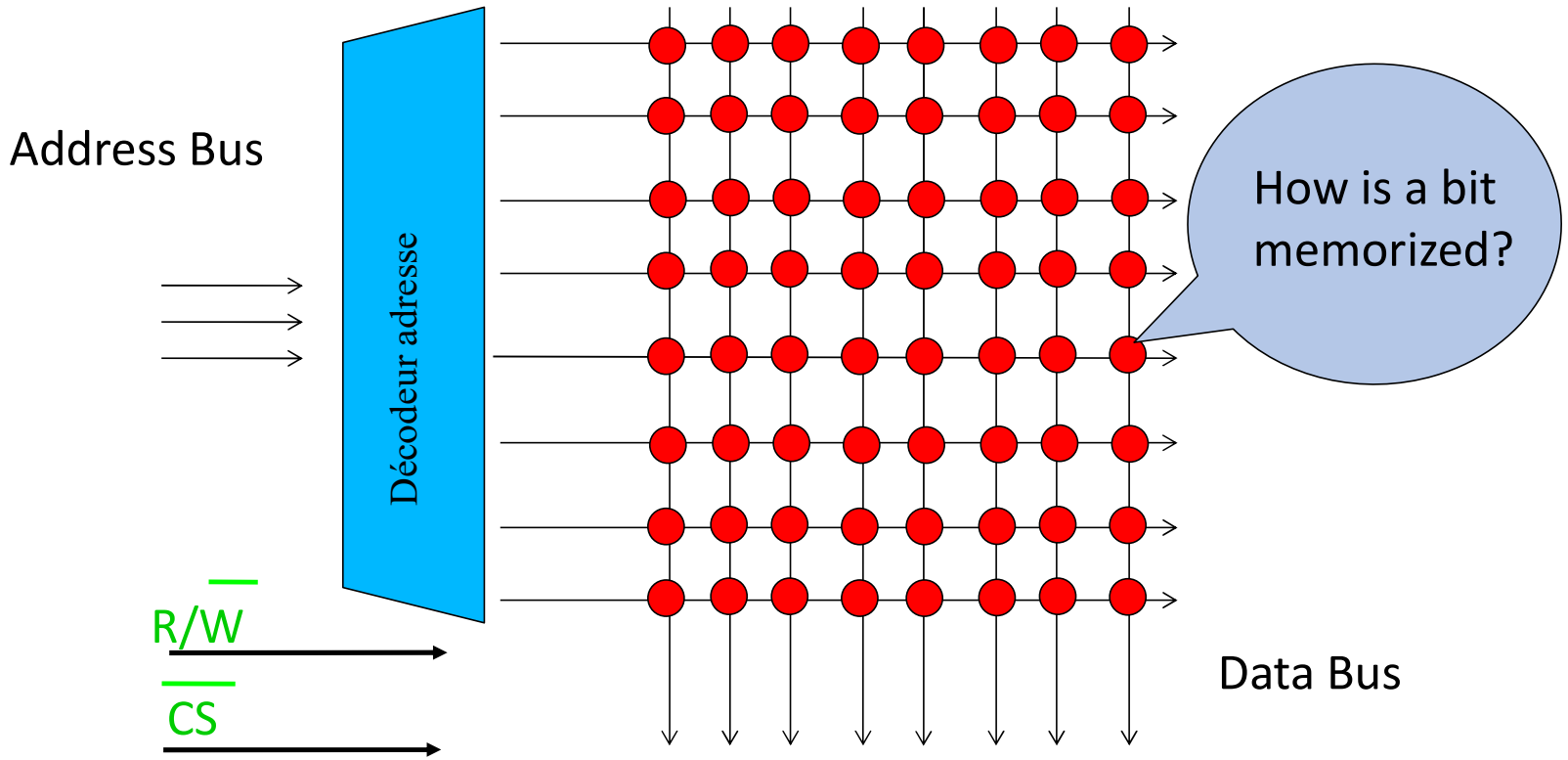
Summary Memory Usage in µControllers

Memory	Usage
FLASH (NOR)	Store Program
EEPROM	Parameters and data
SRAM	Stack

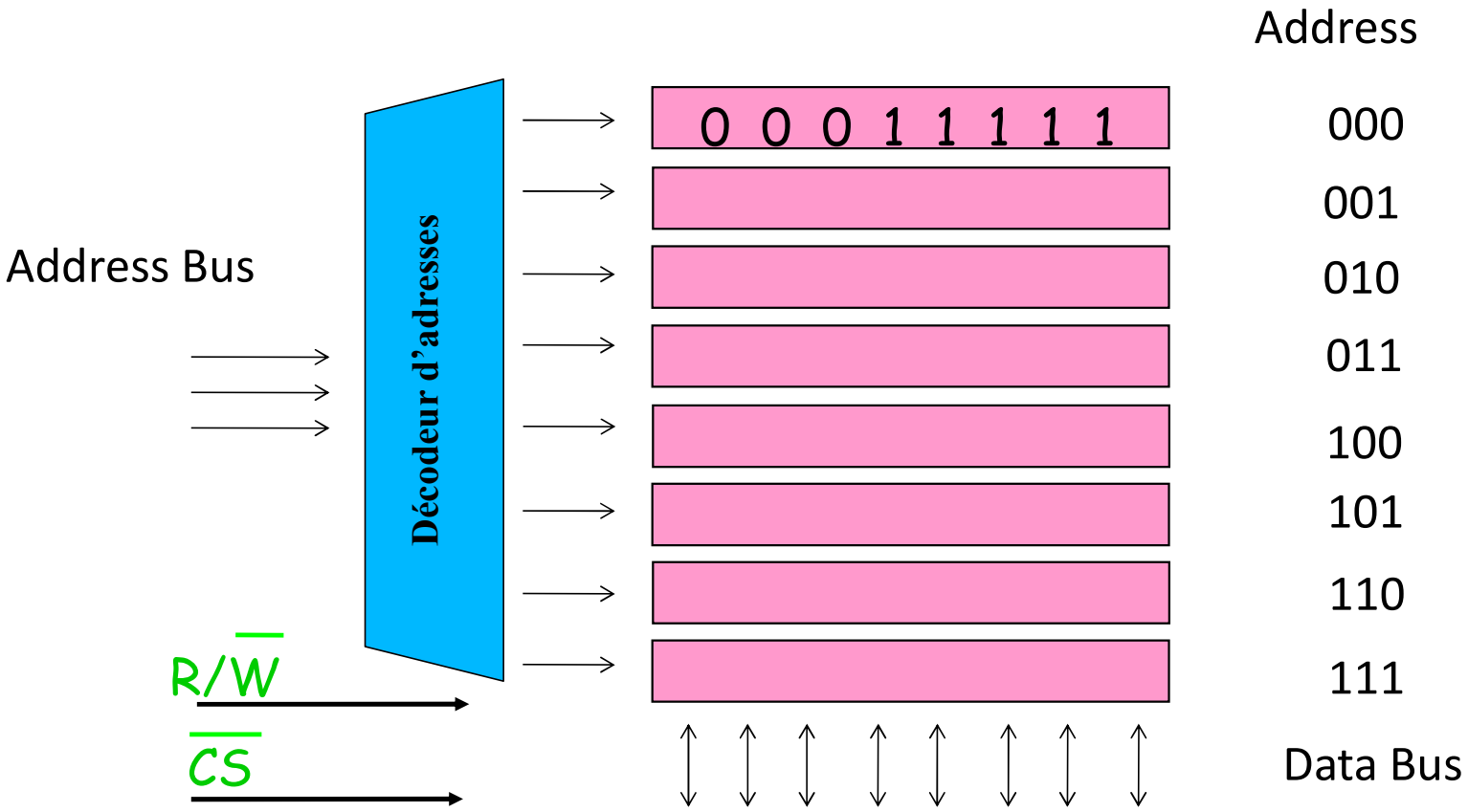
Reason Memory Usage in µControllers

Memory	Usage	Reason
FLASH (NOR)	Store Program	✓ Non-volatile ✓ Cheaper than EEPROM
EEPROM	Parameters and data	✓ Non-volatile ✓ 8-bit granularity
SRAM	Stack	✓ Volatile ✓ 8-bit granularity ✓ Fast access ✓ Unlimited endurance

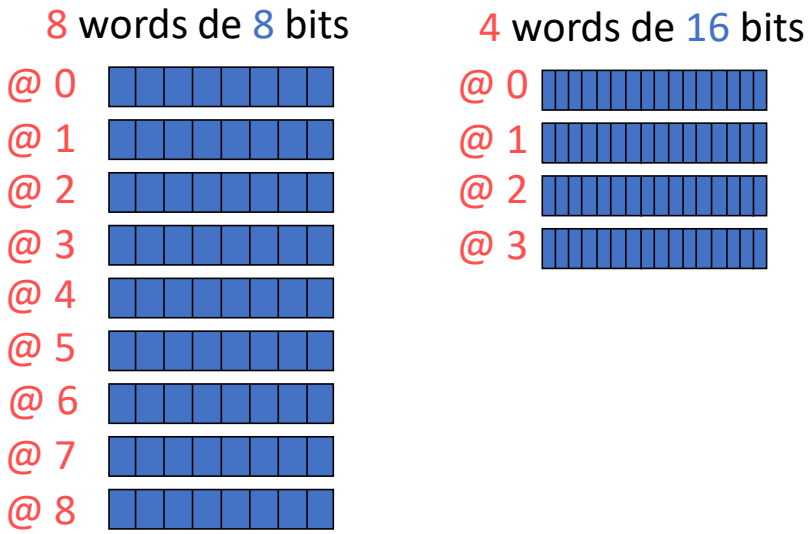
Memory as a Matrix



Memory Organization (8 bytes)

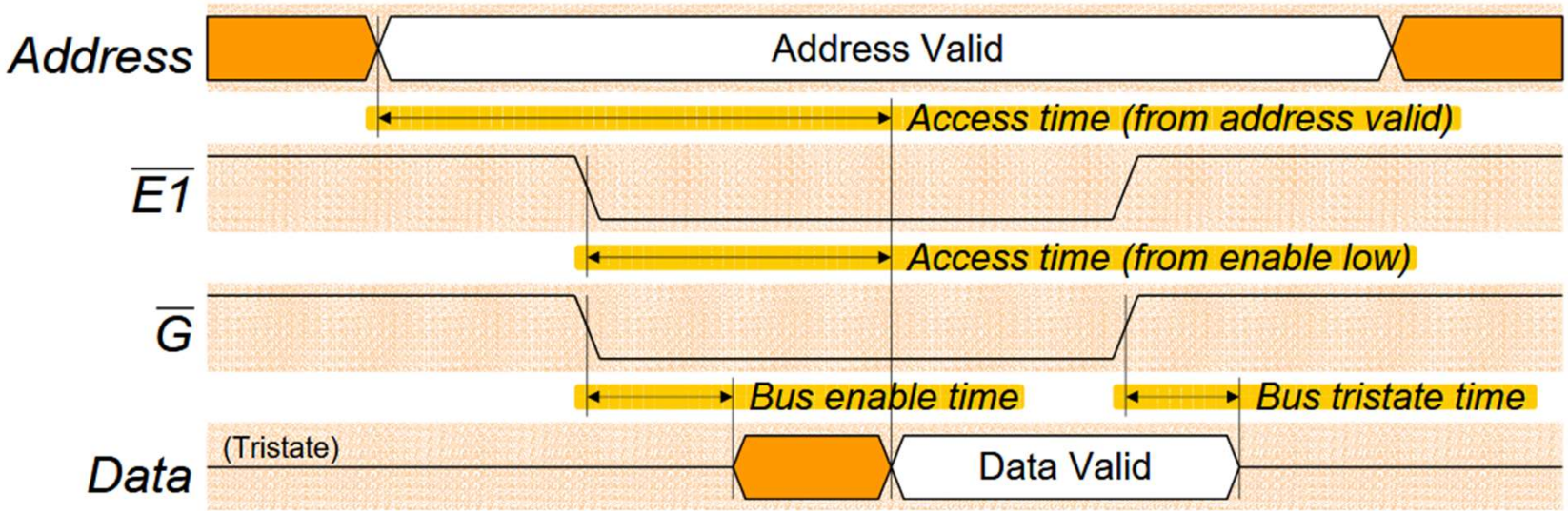


64-bit memory



Memory can be organized with various widths.
In the STM32, memory is organized by 8 bits

Reading Asynchronous SRAM



Time is required to read and write in memory

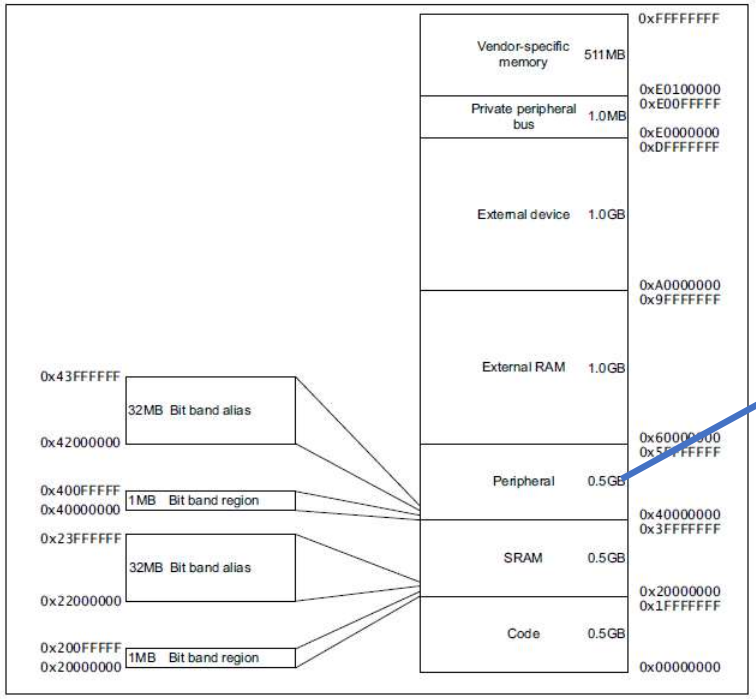
STM32 Memory Organization

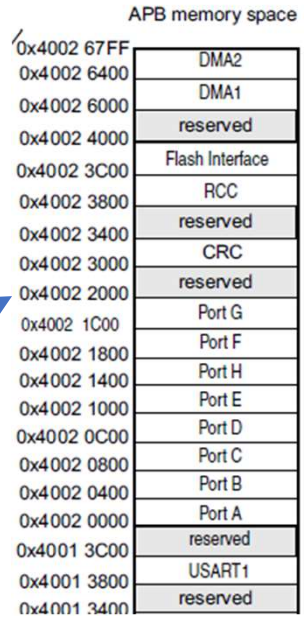
The Cortex-M processors have 32-bit memory addressing and therefore have 4GB memory space. The memory space is unified, which means instructions and data share the same address space.
The 4GB memory space is divided into a number of regions.

4GB Memory Map

Vendor Specific	
Private peripheral bus - external	
Private peripheral bus- internal	
External Device	1.0GB
External RAM	1.0GB
Peripheral	0.5GB
SRAM	0.5GB
Code	0.5GB

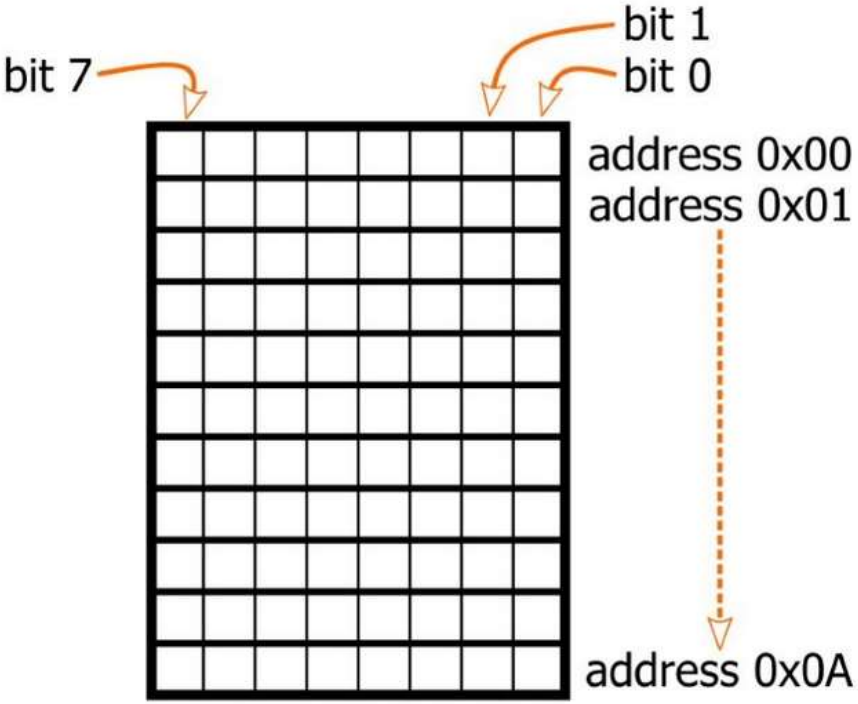
STM32 Memory Map





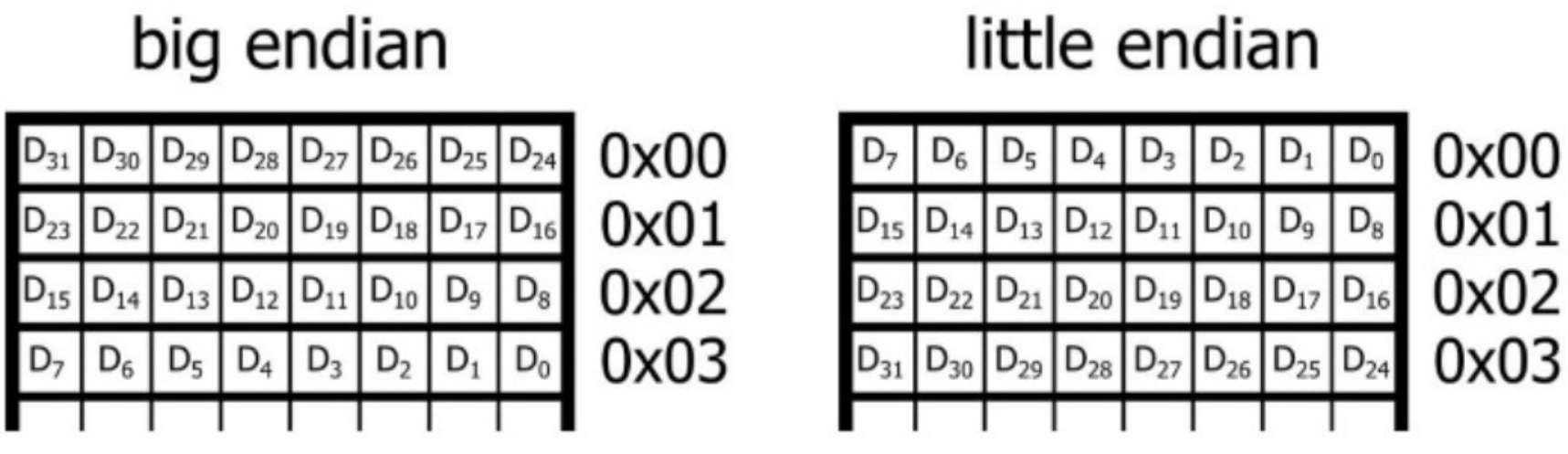
The configuration of a microcontroller is done through registers that are read as a memory with its address.

Endianness



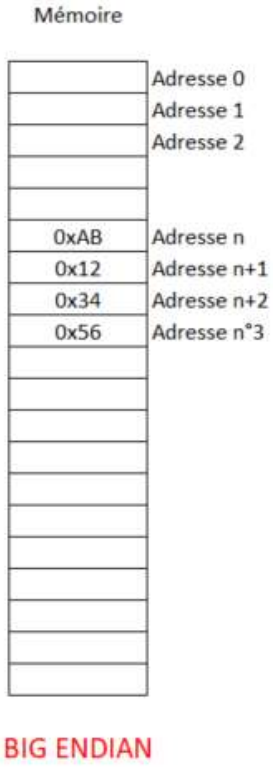
- ✓ Let's assume we want to store a 32-bit variable (4 bytes)
- ✓ It makes sense to store them in contiguous memory location
- ✓ But where to store the MSB?

Big Vs Little Endianness



'D' refers to the 32-bit data word and the subscript numbers indicate the individual bits from MSB (D_{31}) to LSB (D_0)

Endianness

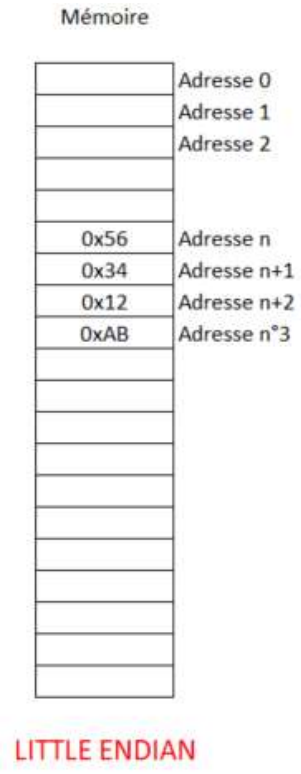


The number 0xAB123456 (2870096982) occupies 4 bytes in memory

With Big Endian, The MSB is at the lowest address.

With Little Endian, the LSB is at the lowest address.

Cortex M can be little or big Endian. Read the specifications to confirm the endianness

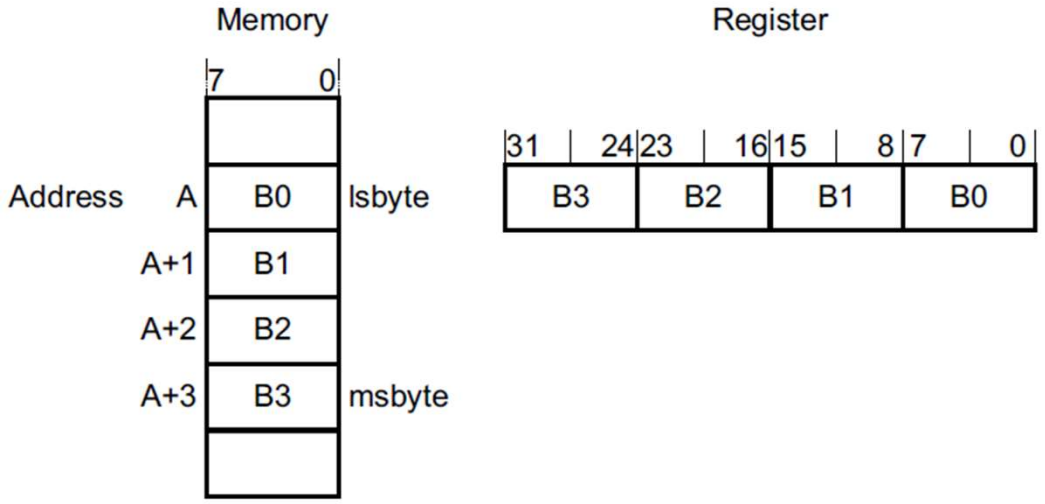


STM32 Endianness

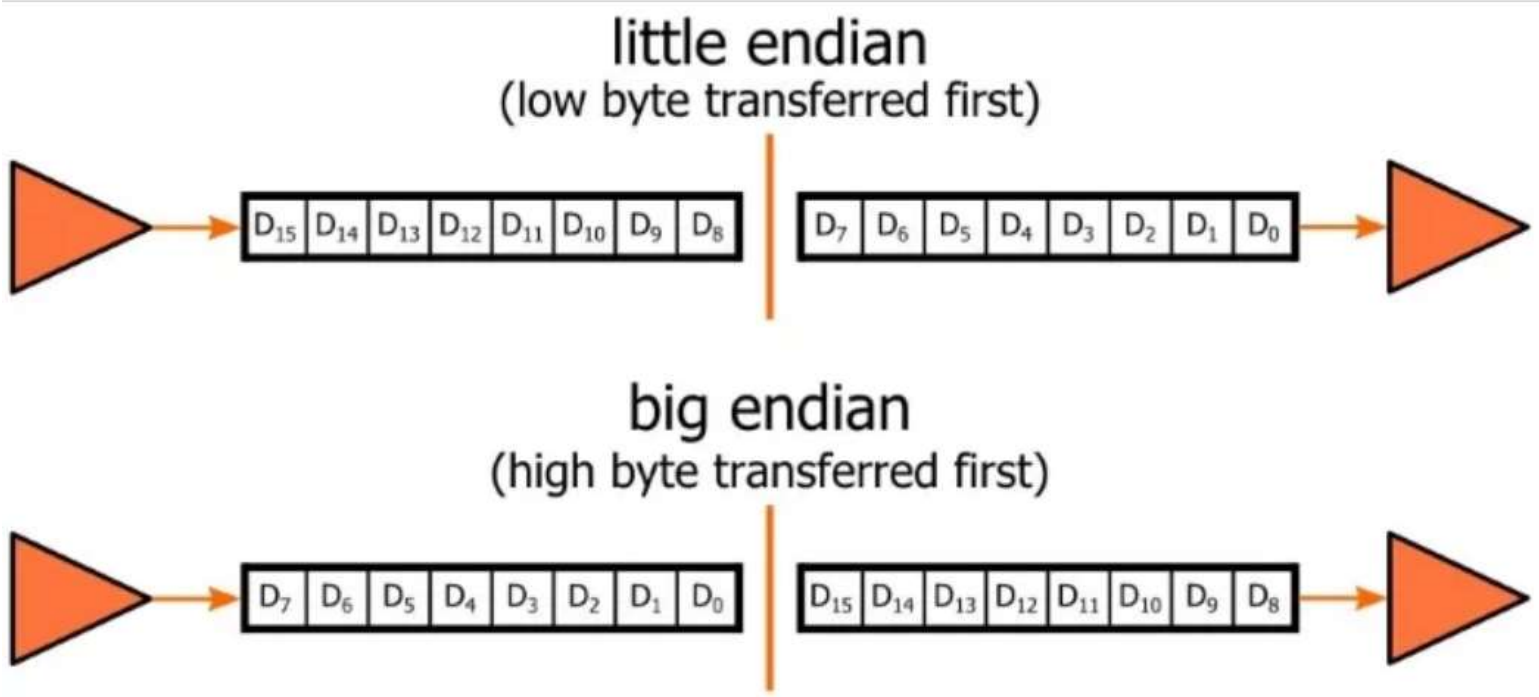
Cortex M can accept big-endian and little-endian format. Silicon vendors choose their format. Information is given in the programming manual.

The STM32L152 is designed with a little-endian format

Figure 6. Little-endian format example

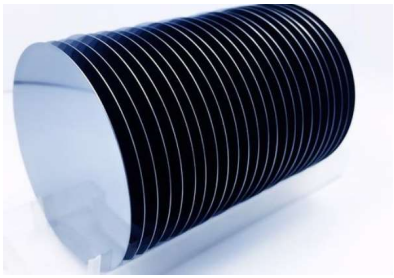


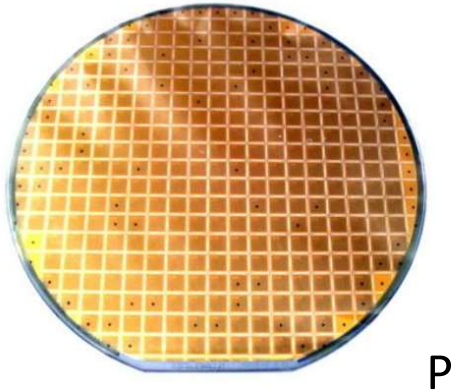
Endianness During Communication



Programming Language of µController

Microcontrollers are manufactured on a silicon wafer with a CMOS technology.

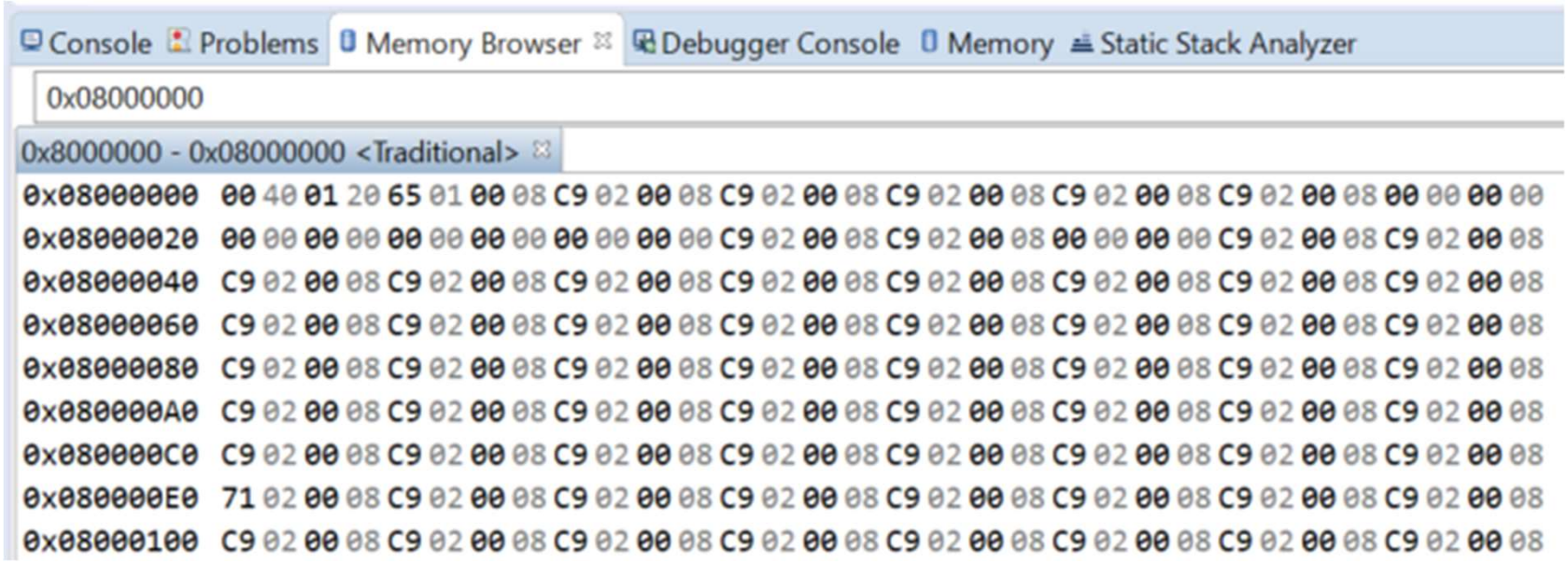




0 1 1 0 1 0 1 0 0 1

Programming language is 0 and 1

Example of Memory



This is the beginning of a program in the STM8L152

Représentation des Nombres Base 10

Système décimal:

Les nombres en base 10 sont constitués de 10 chiffres possibles (0-9).
Chaque chiffre d'un nombre est associé à une puissance de 10 selon sa position dans le nombre.

Par exemple :

$123 = 1 \times 10^2 + 2 \times 10^1 + 3 \times 10^0$

Représentation des Nombres Base 2

Dans un ordinateur, les programmes et les données sont codées en binaire (base 2).
Il n'y a donc que 2 chiffres utilisés: 0 et 1.

Système binaire:

Les nombres en base 2 sont constitués de 2 chiffres (0 et 1).
Chaque chiffre d'un nombre est associé à une puissance de 2 selon sa position dans le nombre.
Par exemple :

$10011 = 1 \times 2^4 + 0 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 1 \times 2^0$

$= 16 + 2 + 1 = 19$

Les bits sont numérotés b0 à bn, de la droite vers la gauche. Le bit de droite est le bit 0 ou Least Significant Bit.
Le bit le plus à gauche est le Most Significant Bit.

Changement de Base (1/2)

Si on veut connaitre la valeur des chiffres décimaux constituant un nombre,
il suffit d'effectuer plusieurs divisons par 10.

Exemple: pour 9876
9876 / 10 = 987 reste 6 , 6 est le chiffre des unités
987 /10 = 98 reste 7, 7 est le chiffre des dizaines
98 /10 =9 reste 8, 8 est le chiffre des centaines et 9 celui des milliers

ISEN ALL IS DIGITAL! OUEST | yncréa

Changement de Base (2/2)

De même pour avoir la représentation binaire d'un nombre, il suffit d'effectuer plusieurs divisions par 2

25 / 2 = 12 reste 1, 1 est la valeur du bit 0
12 / 2 = 6 reste 0, 0 est la valeur du bit 1
6 / 2 = 3 reste 0, 0 est la valeur du bit 2
3 / 2 =1 reste 1, 1 est la valeur du bit 3
1 / 2 = 0 reste 1, 1 est la valeur du bit 4

25 est représenté en binaire par 11001

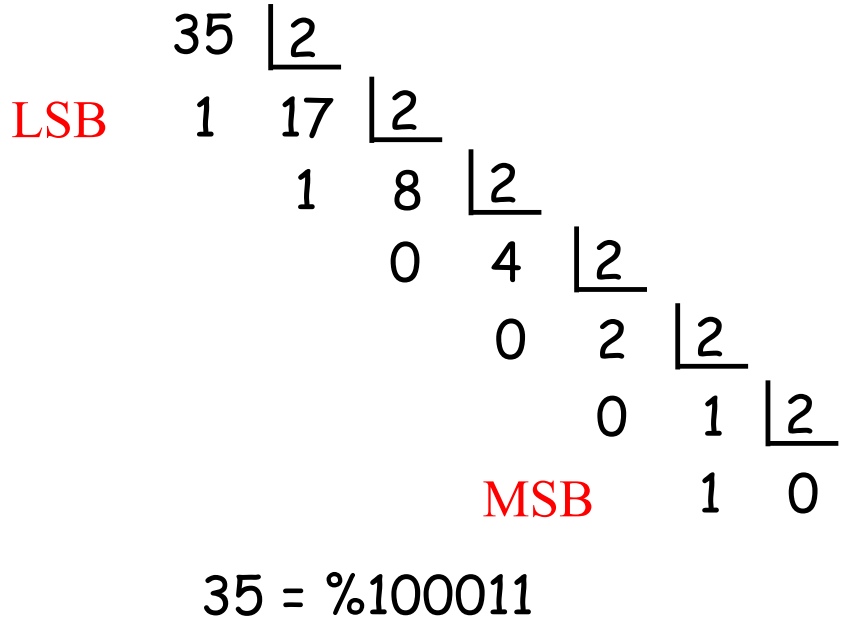
NB: en langage assembleur, un nombre binaire est précédé de %: ex %11001
en langage C, un nombre binaire est précédé de 0b: ex 0b11001

Système de Représentation des Nombres

Changement de base (décimale -> binaire) :

Division successive par 2

Exemple :



35 = %100011

Addition Binaire

A	B	S	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

$$\begin{array}{rr} & 0\,1\,0\,1 \\ + & 0\,0\,1\,1 \\ \hline = & 1\,0\,0\,0 \end{array}$$

S = somme

C = retenue (carry)

Basic Operations

In a microcontroller, the principle is to perform binary operations. The basics ones are:

- ✓ AND
- ✓ OR
- ✓ XOR

ET (AND)

A	B	AND
0	0	0
0	1	0
1	0	0
1	1	1

```
        0 1 0 1 0 0 1 1
And     0 0 1 1 0 1 0 1
        ---------------
=       0 0 0 1 0 0 0 1
```

Application pratique:

- le ET sert à mettre un bit à 0 sans changer les autres bits

Exemple: %xxxxxxxx and %11011111=%xx0xxxxx

ISEN ALL IS DIGITAL! OUEST | yncréa

OU (OR)

A	B	OR
0	0	0
0	1	1
1	0	1
1	1	1

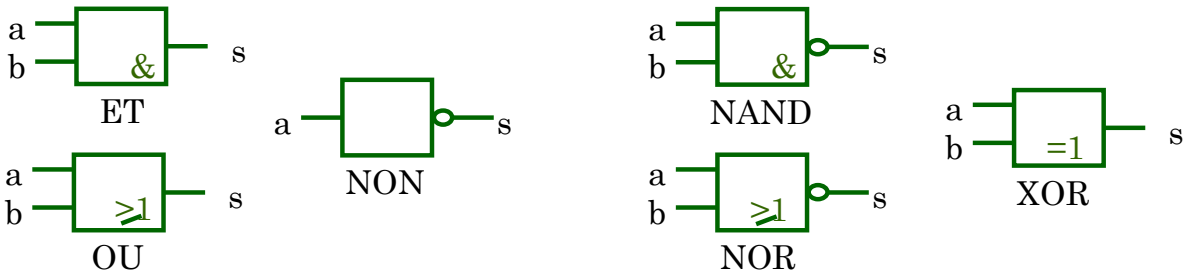
```
        0 1 0 1 0 0 1 1
OR      0 0 1 1 0 1 0 1
=       -----------------
        0 1 1 1 0 1 1 1
```

OU Exclusif (XOR)

A	B	OR
0	0	0
0	1	1
1	0	1
1	1	0

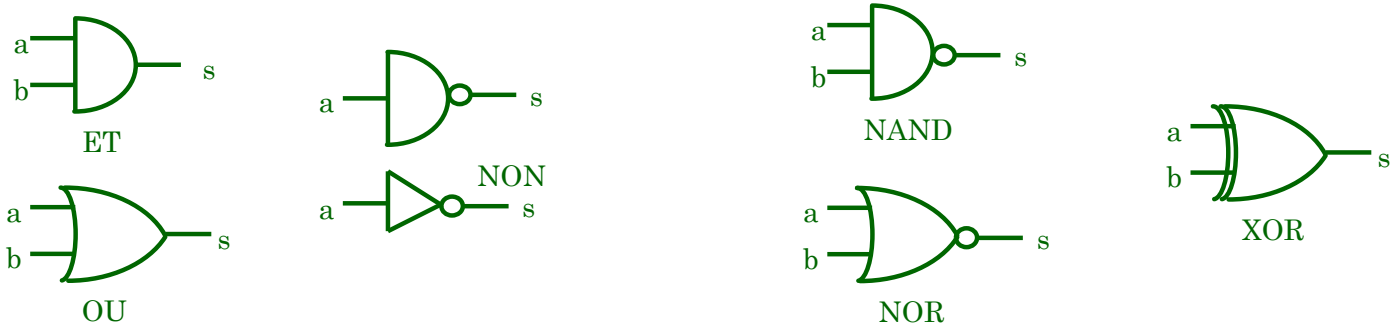
```
        0 1 0 1 0 0 1 1
XOR     0 0 1 1 0 1 0 1
=       0 1 1 0 0 1 1 0
```

Graphiques Portes Logiques (1/2)



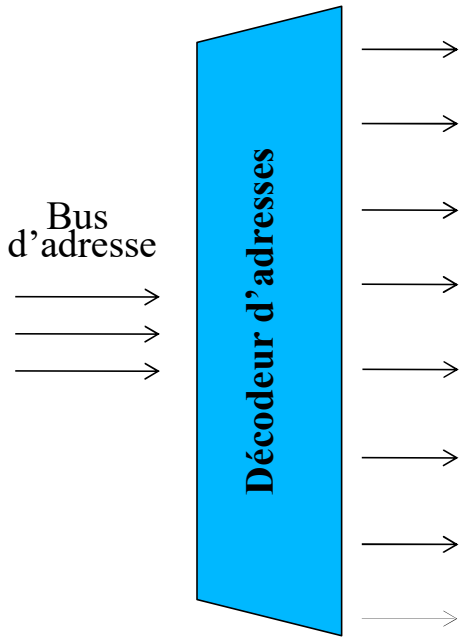
La représentation française est utilisée en symboles logiques
Norme ANSI/IEEE Std: 91-1984 : Standard Graphic Symbols for Logic Functions

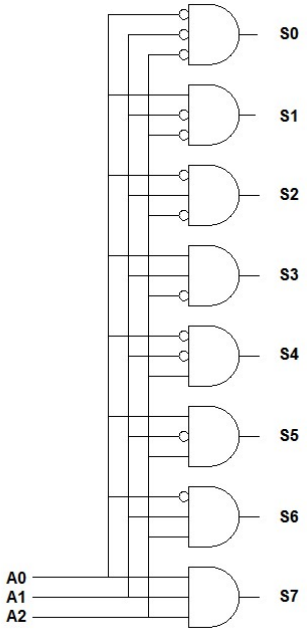
Graphiques Portes Logiques



La norme américaine est la norme utilisée dans les « datasheets » pour dessiner les diagrammes (schémas) logiques.

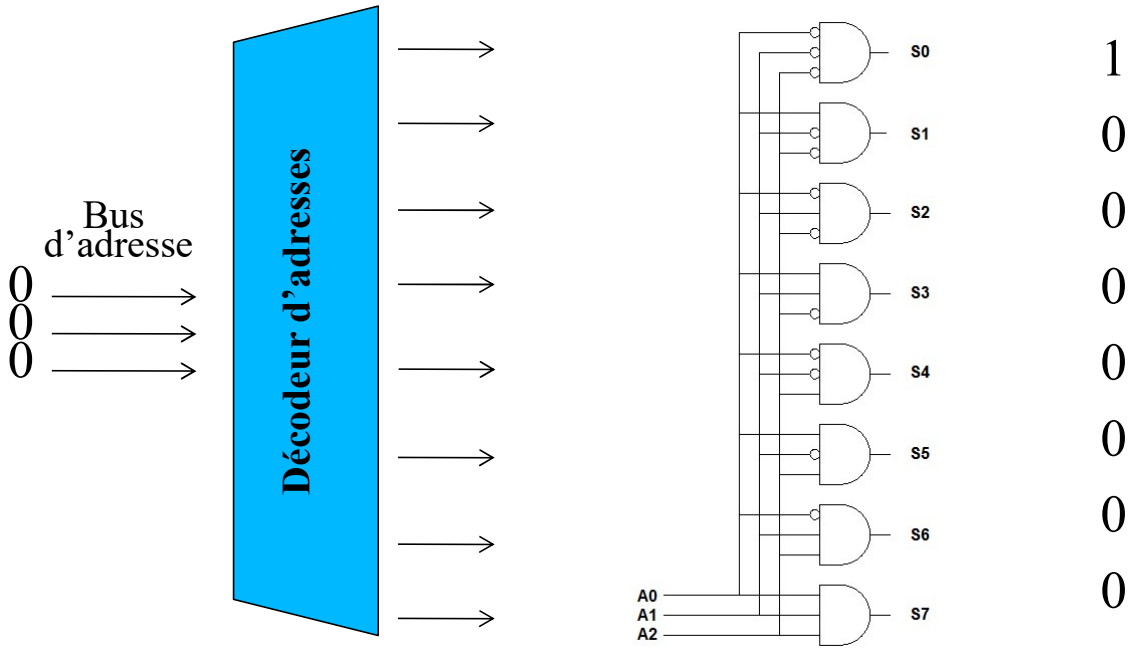
Memory Decoding



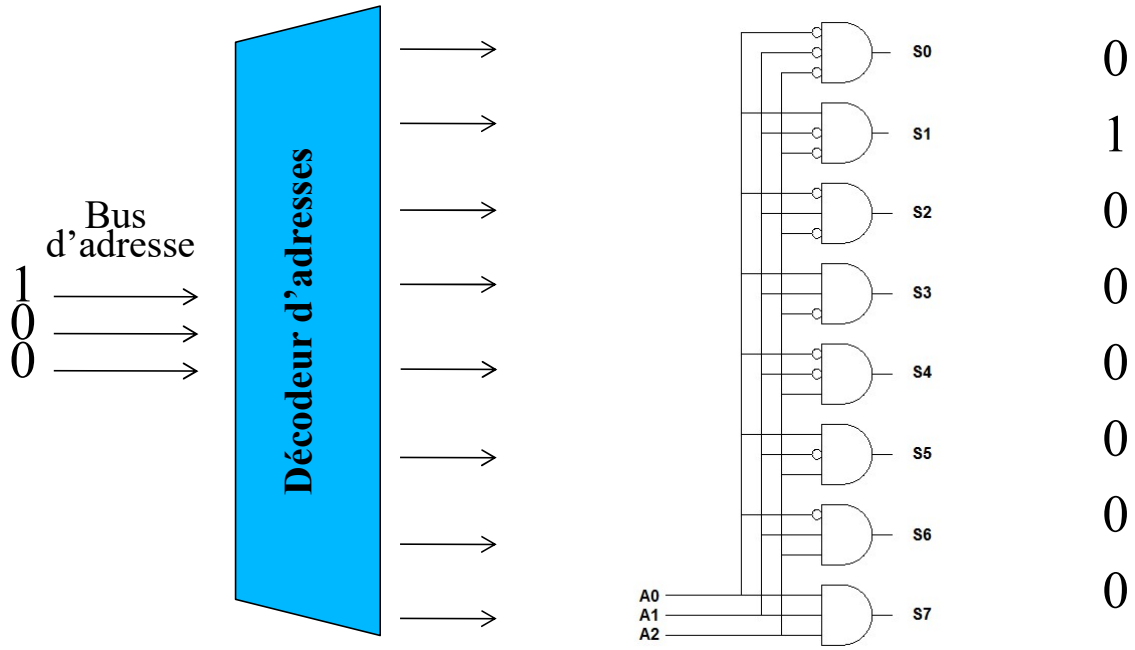


The larger the address bus is, the more complex the electronics is

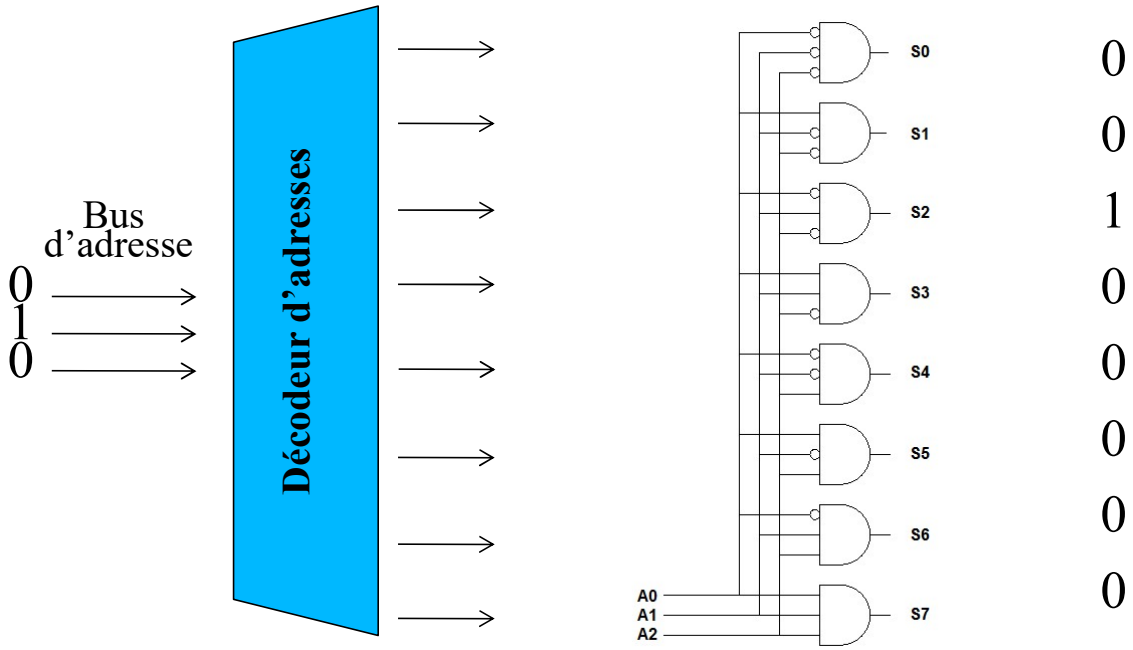
Memory Decoding / 000



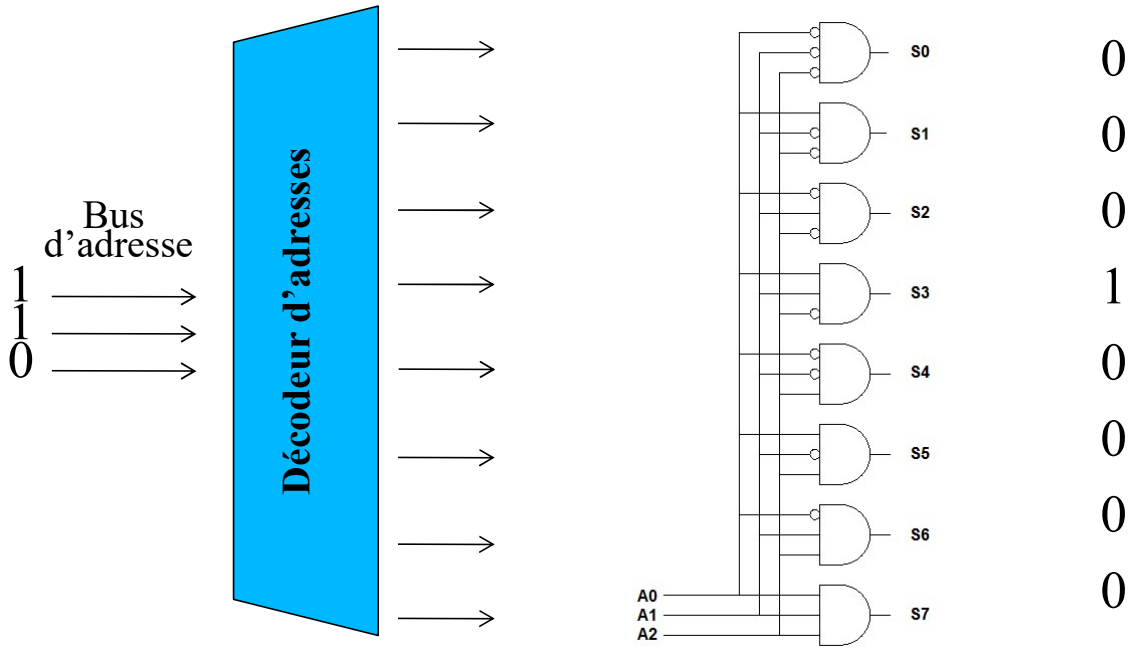
Memory Decoding / 001



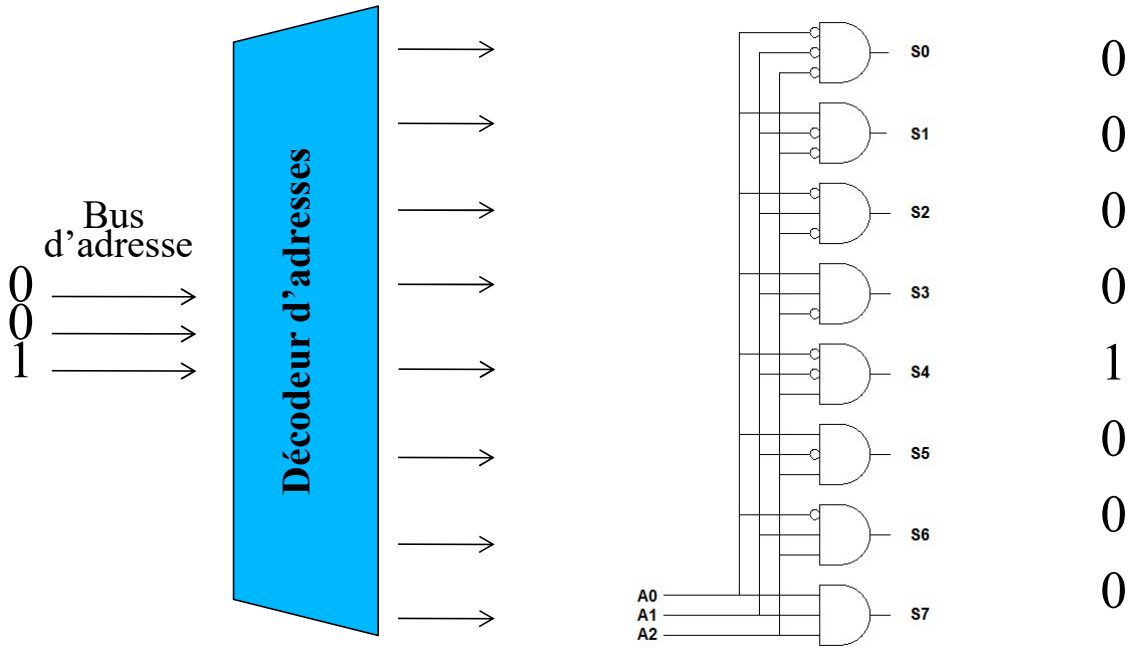
Memory Decoding / 010



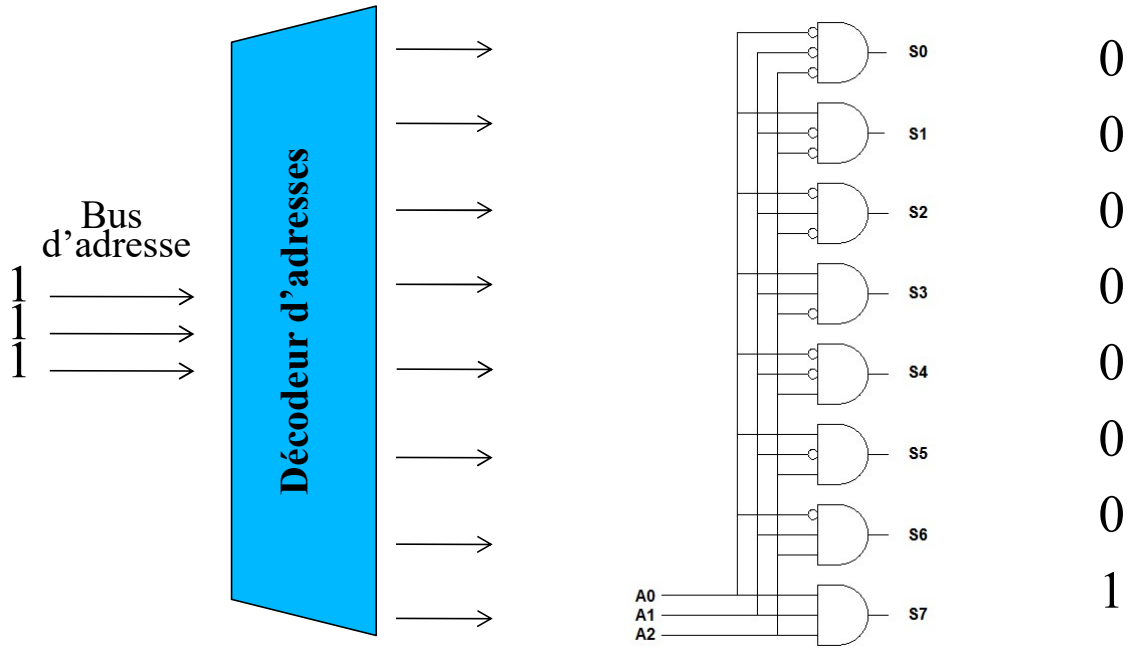
Memory Decoding / 011



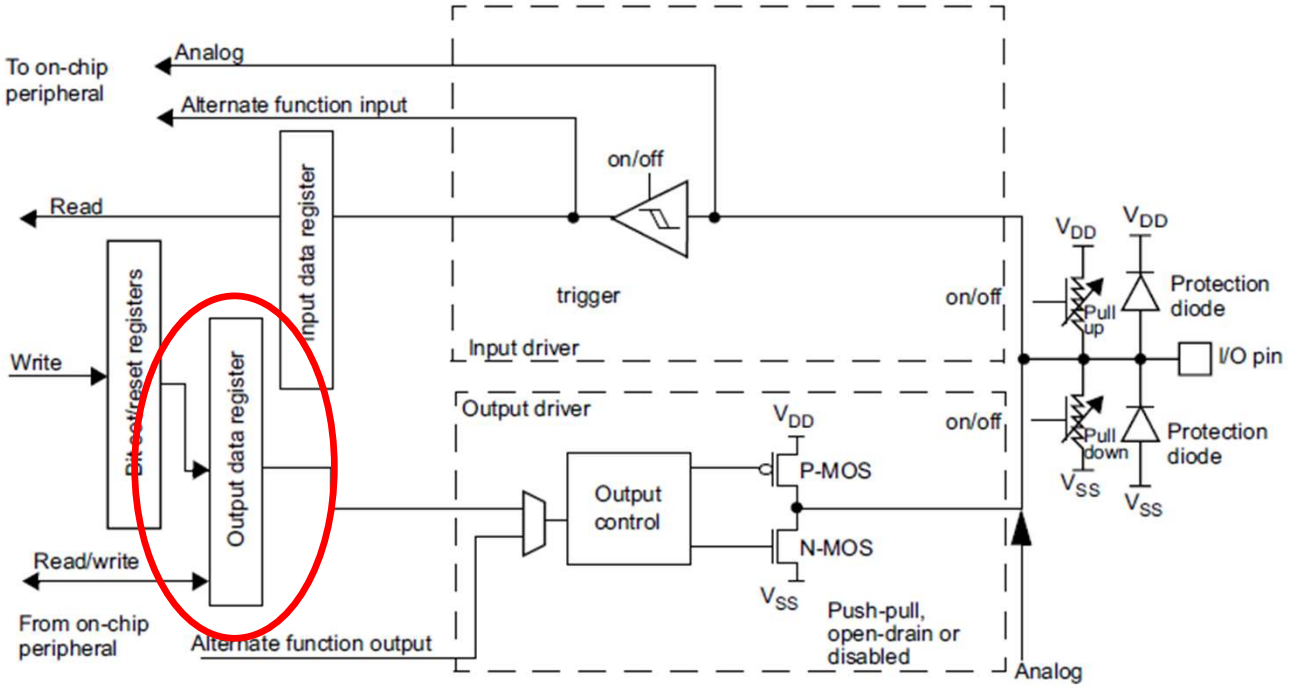
Memory Decoding / 100



Memory Decoding / 111



GPIO Schematic STM32



Every peripheral is configured thanks to registers. GPIO peripheral has an Output Data Register to drive the level of a pin

GPIO ODR STM32

7.4.6 GPIO port output data register (GPIOx_ODR) (x = A..H)

Address offset: 0x14

Reset value: 0x0000 0000

31	30	29	28	27	26	25	24	23	22	21	20	19	18	17	16
Reserved															
15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
ODR15	ODR14	ODR13	ODR12	ODR11	ODR10	ODR9	ODR8	ODR7	ODR6	ODR5	ODR4	ODR3	ODR2	ODR1	ODR0
rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw	rw

Bits 31:16 Reserved, must be kept at reset value.

Bits 15:0 **ODRy**: Port output data (y = 0..15)

These bits can be read and written by software.

Note: For atomic bit set/reset, the ODR bits can be individually set and reset by writing to the GPIOx_BSRR register (x = A..H).

The Output Data Register of a General Purpose Input Output is to drive the level of a pin. Each bit represents a pin → Requirement to change the value of one bit → Requirement of bit Masking

Bit Masking

Bit masking has the following purpose

- ✓ Masking bits to 1
- ✓ Masking bits to 0
- ✓ Querying the status of a bit
- ✓ Toggling bit values

Masking bits to 1

To turn some bits on, the bitwise OR operation is used. Following the principle that

Y OR 1 = 1 and Y OR 0 = Y

Therefore, to make sure a bit is ON, OR can be used with a '1'. To leave a bit unchanged, OR is used with a '0'.

X	Y	X OR Y
0	0	0
0	1	1
1	0	1
1	1	1

masking on the higher nibble (bits 4, 5, 6, 7)

	1001 0101
OR	1111 0000
=	1111 0101

masking on the lower nibble (bits 0, 1, 2, 3)

	1010 0101
OR	0000 1111
=	1010 1111

Masking bits to 0

To turn certain bits off, the bitwise AND operation can be used. Following the principle that

Y AND 0 = 0 and Y AND 1 = Y

Therefore, to make sure a bit is OFF, AND can be used with a '0'. To leave a bit unchanged, AND is used with a '1'.

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

masking on the higher nibble (bits 4, 5, 6, 7)

	1001 0101
AND	0000 1111
=	0000 0101

masking on the lower nibble (bits 0, 1, 2, 3)

	1010 0101
AND	1111 0000
=	1010 0000

Querying Status Bit

To check the state of individual bits regardless of the other bits, all the other bits use the bitwise AND to be turned off. Then the value is compared with '1'. If it is equal to '0', then the bit was off ('0')

X	Y	X AND Y
0	0	0
0	1	0
1	0	0
1	1	1

Check Status bit 3

	1001 1101
AND	0000 1000
=	0000 1000

	1001 0101
AND	0000 1000
=	0000 0000

Toggling a Bit

In order to toggle, you need need to perform an exclusive OR called XOR

If X = 0, Y is conserved

If X = 1, Y is toggled

X	Y	X XOR Y
0	0	0
0	1	1
1	0	1
1	1	0

Toggling one byte

	1001 1101
XOR	1111 1111
=	0110 0010

Toggling on the lower nibble (bits 0, 1, 2, 3)

	1001 1101
XOR	0000 1111
=	1001 0010

ISEN ALL IS DIGITAL! OUEST | yncréa

Numération en Hexadécimal Base 16

It’s easier to manipulate large number in hexadecimal format

Numbers in base 16 have 16 available symbols

0, 1, 2, 3, 4, 5, 6, 7, 8 , 9, A, B, C, D, E, F

Hex numbers from 0 to 32

0 1 2 3 4 5 6 7 8 9 A B C D E F

10 11 12 13 14 15 16 17 18 19 1A 1B 1C 1D 1E 1F

20 ….

Hex Numbers Notation

- ✓ In assembly language, an hex number is written with an '$' sign: e.i. $1AB
- ✓ In C language, an hex number is written with an '0x' sign: i.e. 0x1AB

- ✓ Binary numbers is written with a '0b' sign: i.e. 0b1000

- ✓ 0b1000 1010 = 0x4A

Conversion Hexadecimal-decimal

$1A2B = 1 x 16^3 + 10 x 16^2 + 2 x 16^1 + 11 x 16^0

= 1 x 4096 + 10 x 256 + 2 x 16 + 11

= 6699

ISEN ALL IS DIGITAL! OUEST | yncréa

Conversion Decimal-Hexadecimal

e.i. : 5567

```
5567 |16
  15 347 |16
     11  21 |16
          5  1 |16
             1  0
```

5567 = $15BF

ISEN ALL IS DIGITAL! OUEST | yncréa

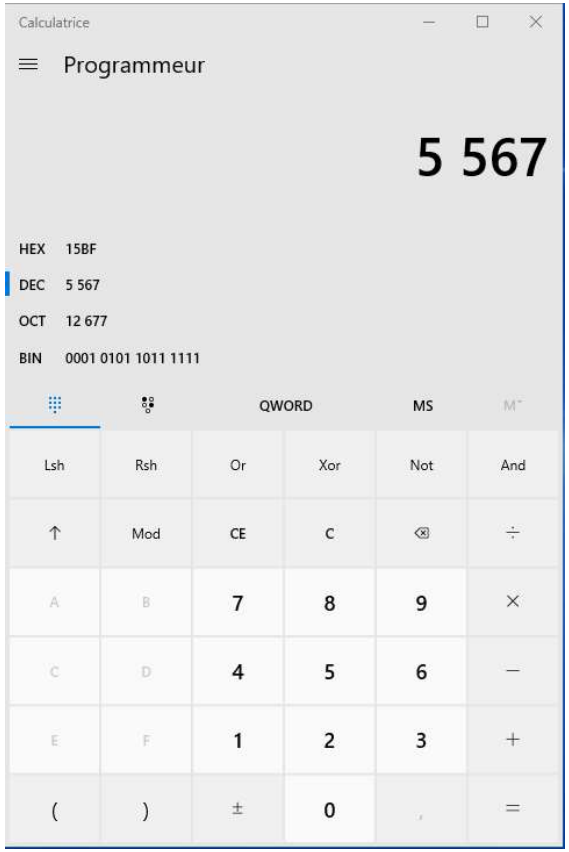
Conversion Decimal-Hexadecimal

décimal	binaire	hexadécimal
0	0000	0
1	0001	1
2	0010	2
3	0011	3
4	0100	4
5	0101	5
6	0110	6
7	0111	7

décimal	binaire	hexadécimal
8	1000	8
9	1001	9
10	1010	A
11	1011	B
12	1100	C
13	1101	D
14	1110	E
15	1111	F

Conversions in Calculator

Windows calculator Program



Conversion Dec-Hex in IDE