ELEC 5 : Microcontrôleur GISTRE

Corentin Vigourt
26/02/2025

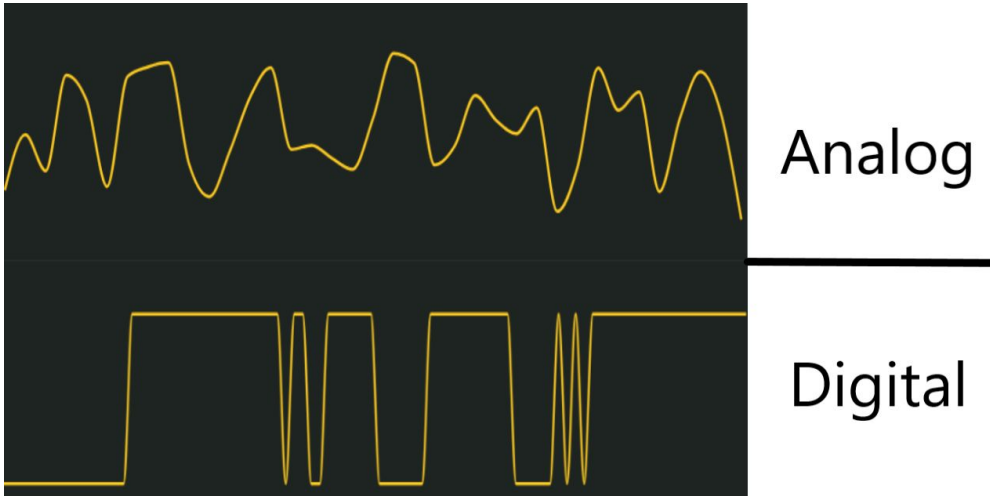
Sommaire

- Communication en électronique
- ICs avancés
- Quésaco ?
- Familles
- ATTINY85
- Manipulation du jour

Communication en électronique

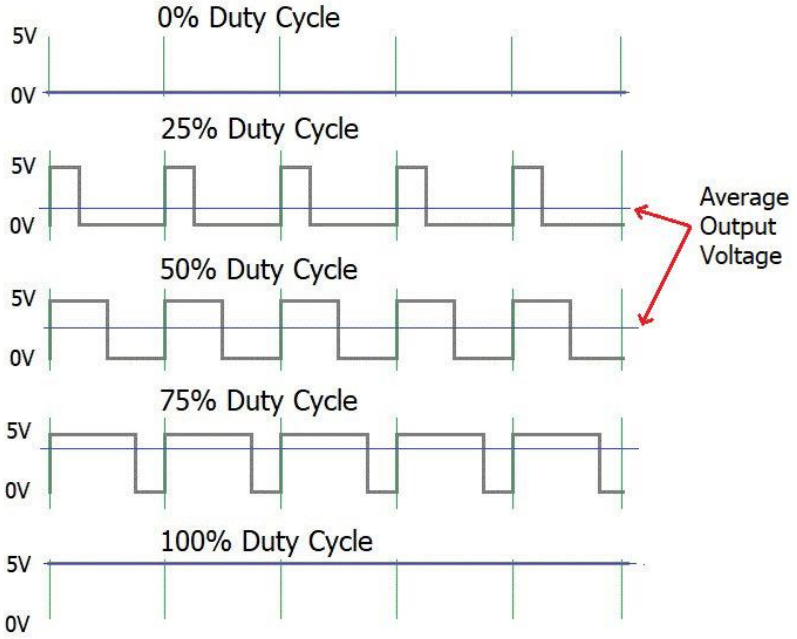
I/O : Analog / Digital

- Permet l'envoie d'informations simples
- L'analogique est surtout utilisée pour des capteurs simples (température, pression, humidité, ...)
- Le digital, pour des signaux de contrôle (on/off, activation de relais, ...)



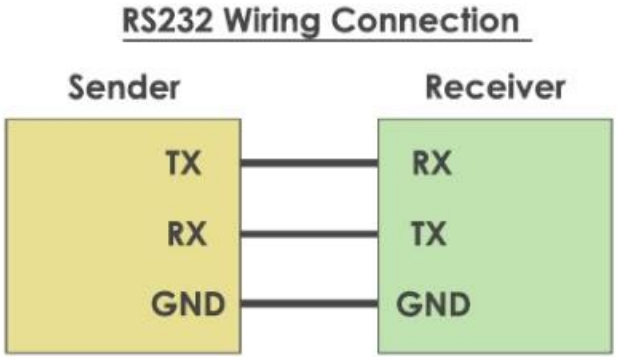
I/O : PWM (Pulse width modulation)

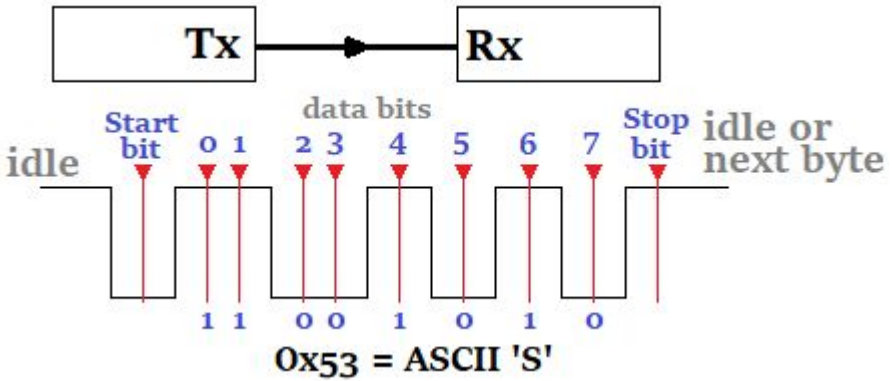
- Permet de simuler une valeur analogique sur un temps donné
- Utilisé pour :
  - Le contrôle de moteur en vitesse
  - Le contrôle de servomoteur
  - L'envoie de données



I/O : Serial

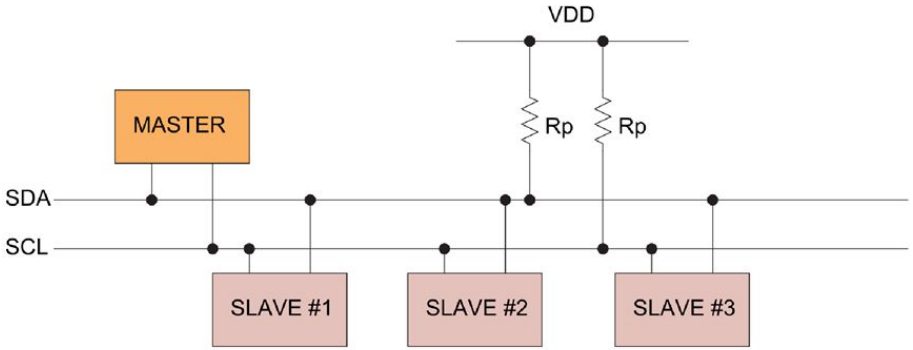
- Permet d'échanger des informations entre deux appareils
- Fiable
- Simple à mettre en place
- Standard
- Utilisé partout (USB)

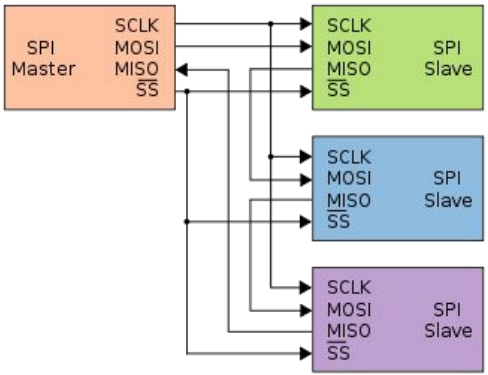




I/O : I2C - SPI

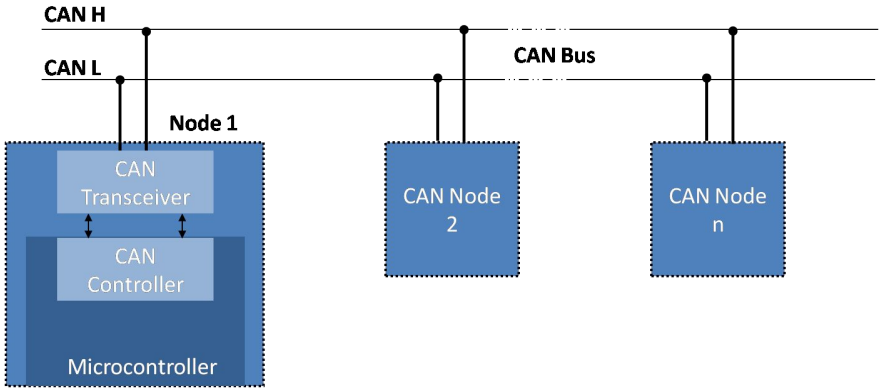
- Communication dite “Master-Slave”
- Permet de facilement ajouter des appareils sur le même bus
- Communication 100% à l’initiative du Master
- Beaucoup utilisé en IOT
- C’est de la merde





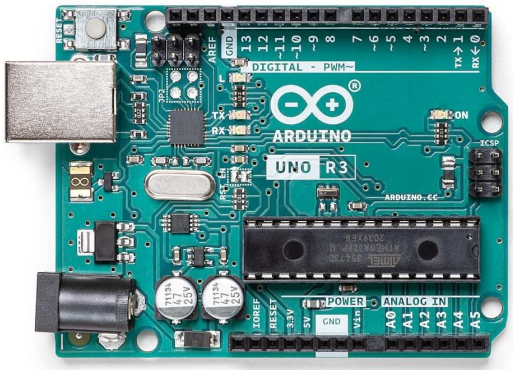
I/O : CAN

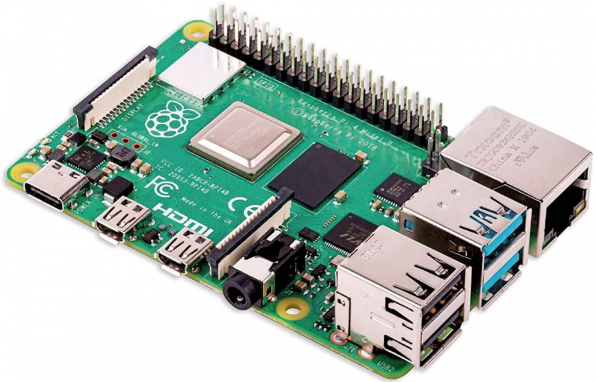
- Très fiable
- Très rapide
- Très modulaire
- Surtout utilisé dans l'automobile
- Plus difficile à mettre en place



ICs avancés

Trois gros challengers





Il existe des combinaisons (ex : Xilinx)

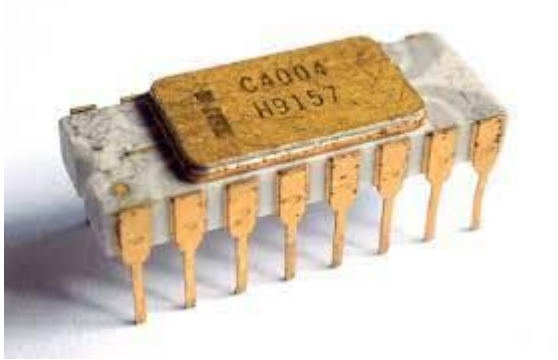
Pourquoi utilise-t-on ces ICs ?

- Faire plus de choses avec un seul composant
- Pouvoir être modulaire
- Avoir de la puissance de calcul
- Faire du code

Quésaco ?

Microcontrôleur

- Premier modèle en 1971 (Intel 4004)
- IC qui embarque l’essentiel d’un ordinateur
- Taille réduite
- Consommation faible
- Intégration facile et rapide



Microprocesseur vs Microcontrôleur

- Dépourvu de périphériques
- Circuit volumineux
- Cher
- Grande capacité de calcul
- Peut effectuer plusieurs tâches
- Consomme beaucoup
- Pas de temps réel (dépendance)

- Comprend tout le nécessaire pour fonctionner
- Compact
- Peu cher
- Faible capacité de calcul
- Peut effectuer qu’une seul tâche à la fois
- Consomme peu
- Peut faire du temps réel

Pourquoi utilisons-nous des microcontrôleurs ?

- Développement rapide
- S'embarque sur toutes les plateformes
- Beaucoup de périphériques disponibles

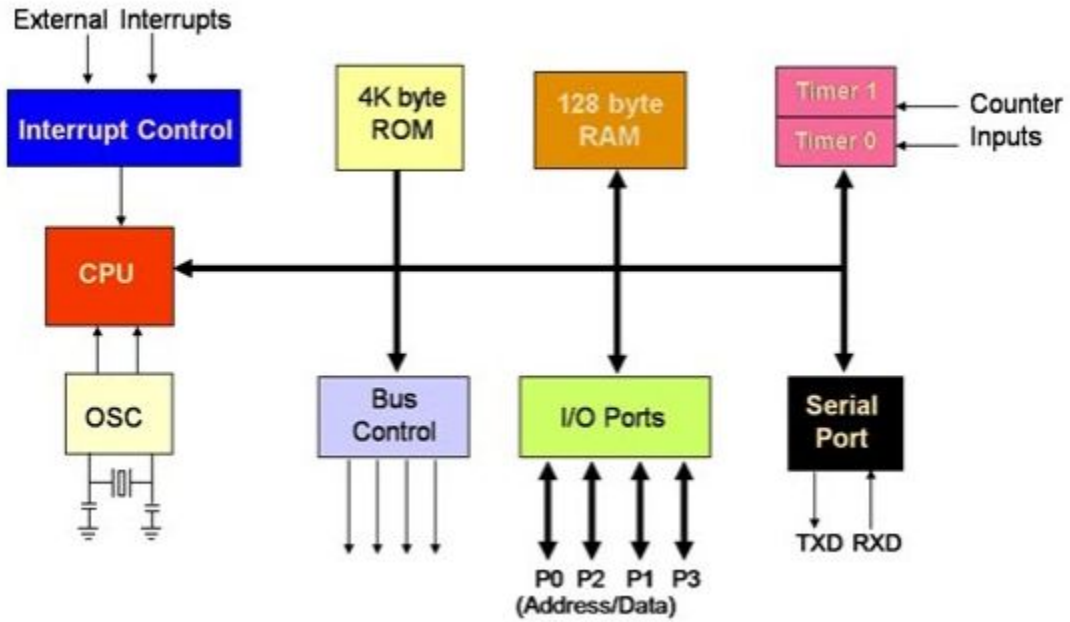
On en trouve notamment dans :

- Tous les appareils du quotidien (grille-pains, machines à laver, jouets, …)
- Toutes les plateformes critiques (avions, voitures, appareils médicaux, …)

Périphériques

- Mémoire vive (RAM)
- Mémoire morte (ROM, Flash, EMMC)
- Timers
- I/O
- ADC
- DAC
- …

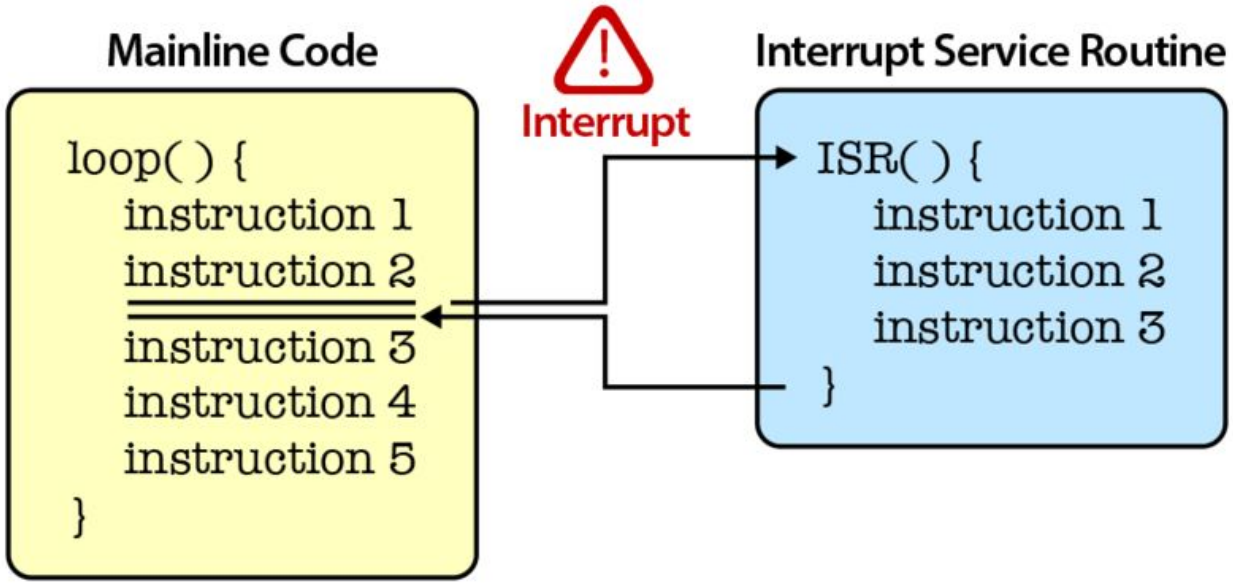
Architecture type



Tout dans les registres

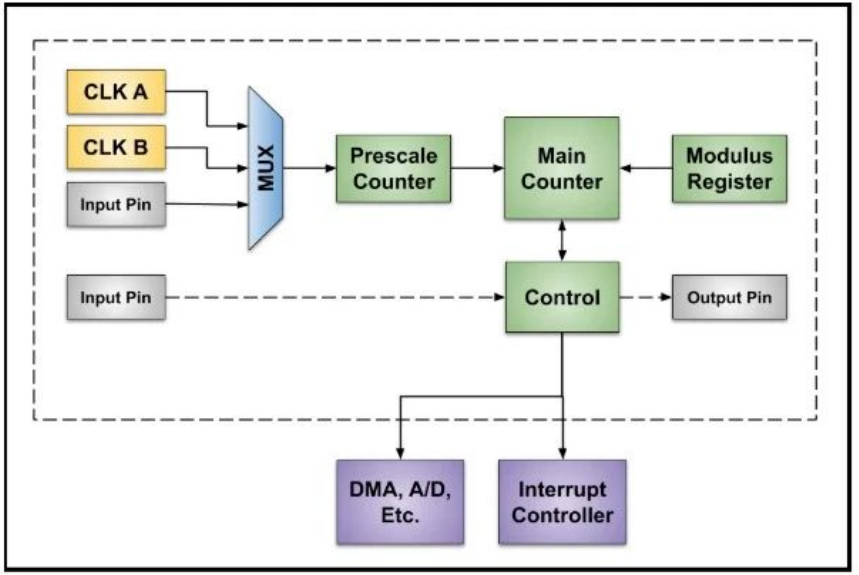
- Les fonctionnalités sont accessibles par les registres
- Ce sont juste des bascules D
- Manipulables par le programme
- La taille dépend de l’architecture

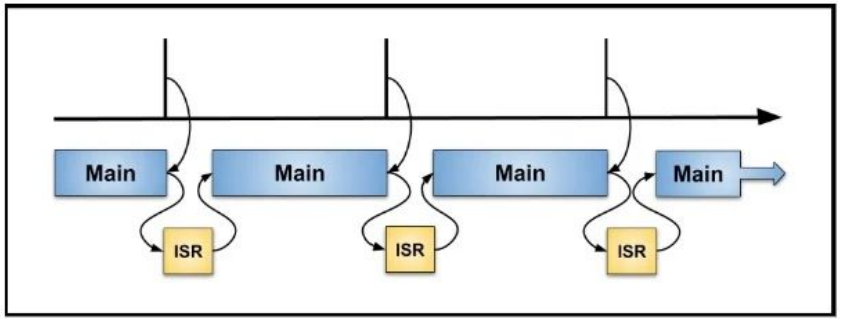
Interruptions



Timers

- Périphérique de manipulation de temps
- Basiquement compte
- Permet de faire pas mal de choses :
  - Compteur
  - PWM
  - Watchdog
  - ...
- Configurable
- Indépendant du CPU

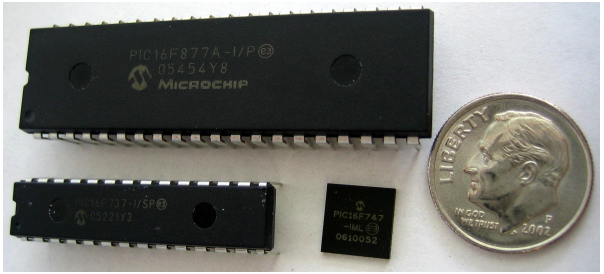


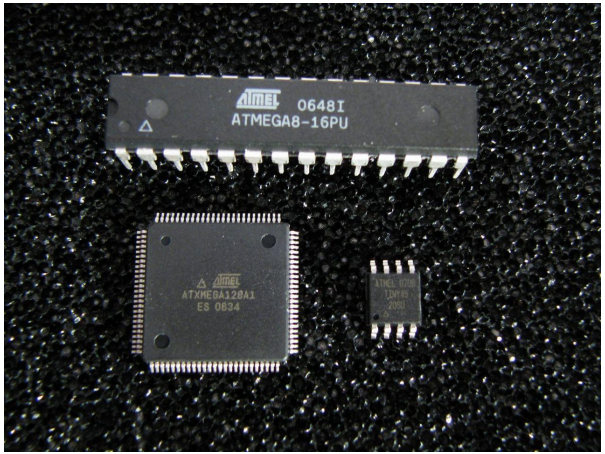


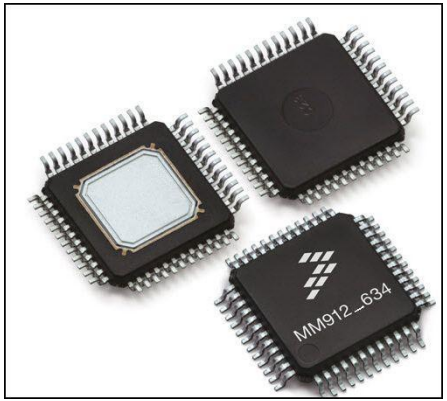
Familles

Grandes familles

- PIC
- AVR
- ARM
- Freescale
- Intel
- ...
-

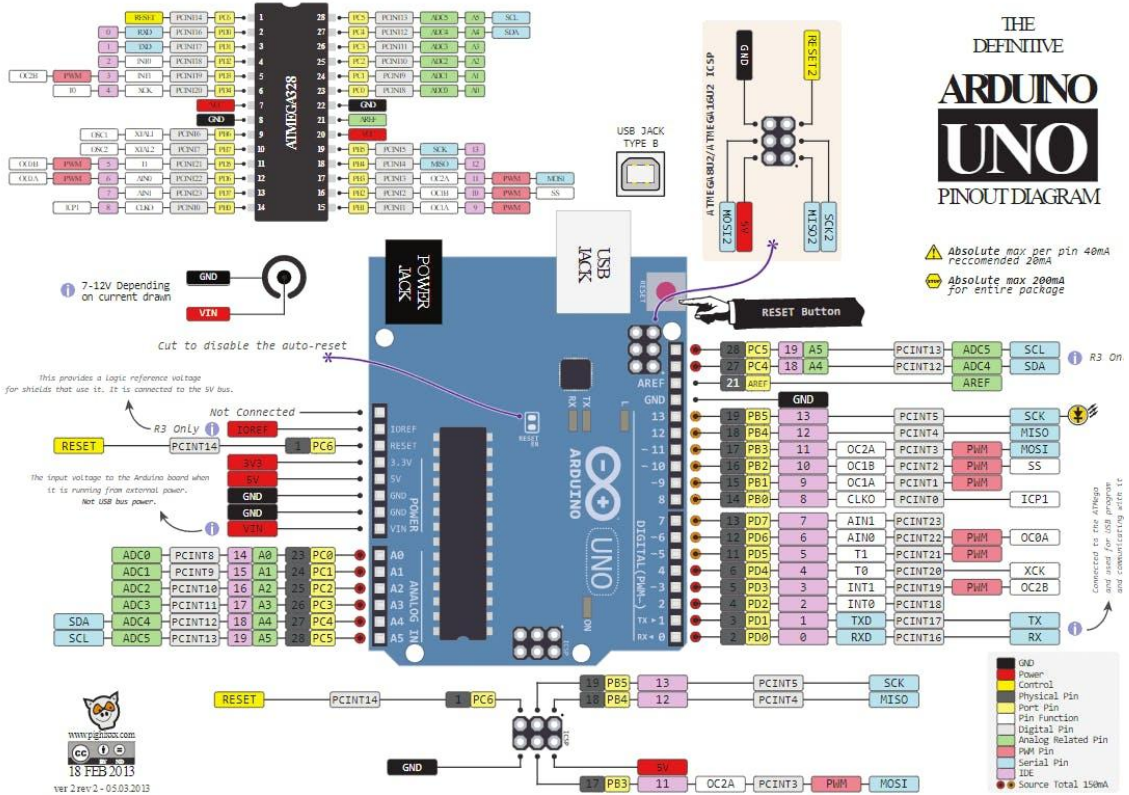






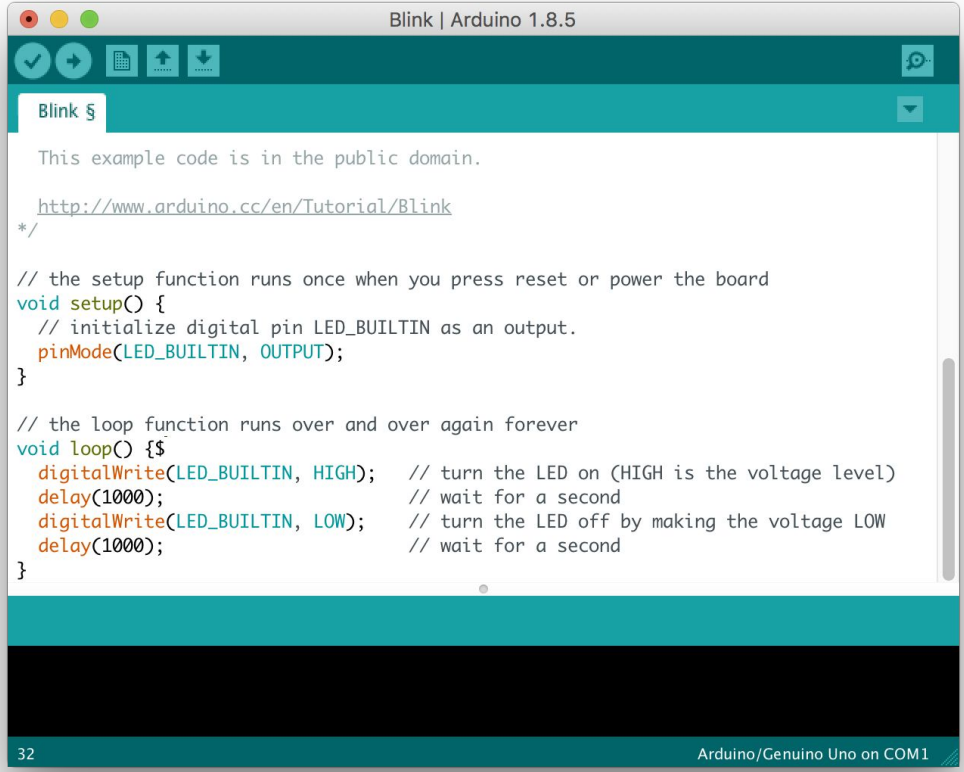
Arduino

ARDUINO™

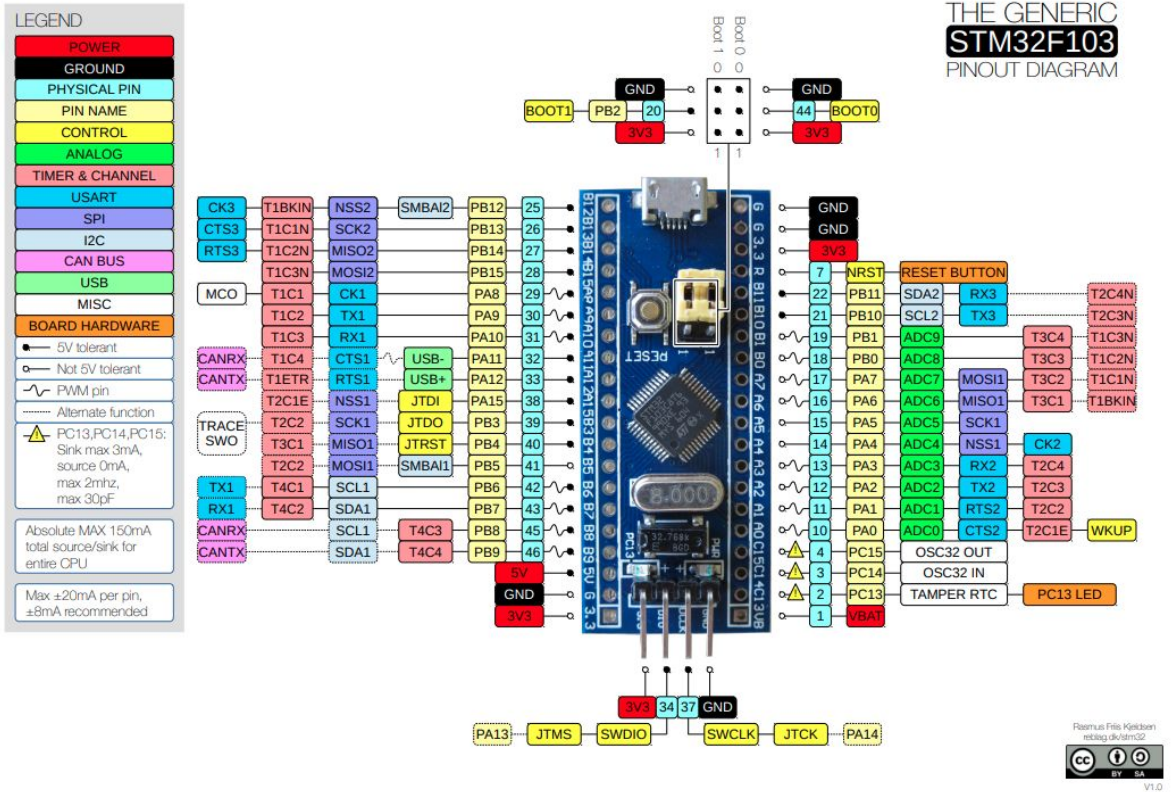


Exemple : Hello world !

- Pour développer et programmer : Arduino IDE
- Hello world des électroniciens : faire clignoter une led
- Deux fonctions :
  - setup : setup de votre projet
  - loop : fonction appelée en boucle
- Fonctions appelées :
  - pinMode : permet de configurer le mode d'une pin
  - digitalWrite : permet d'écrire une valeur digital sur une pin
  - delay : permet de faire dormir le code pendant x ms

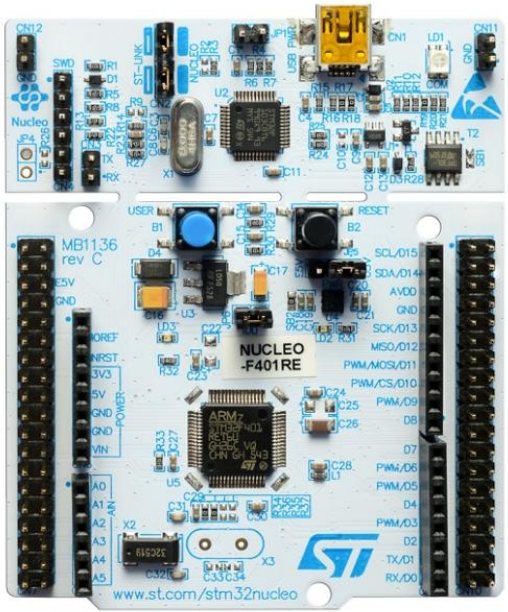


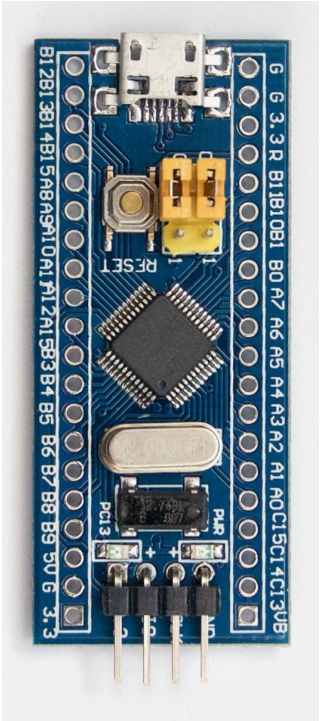
life.augmented



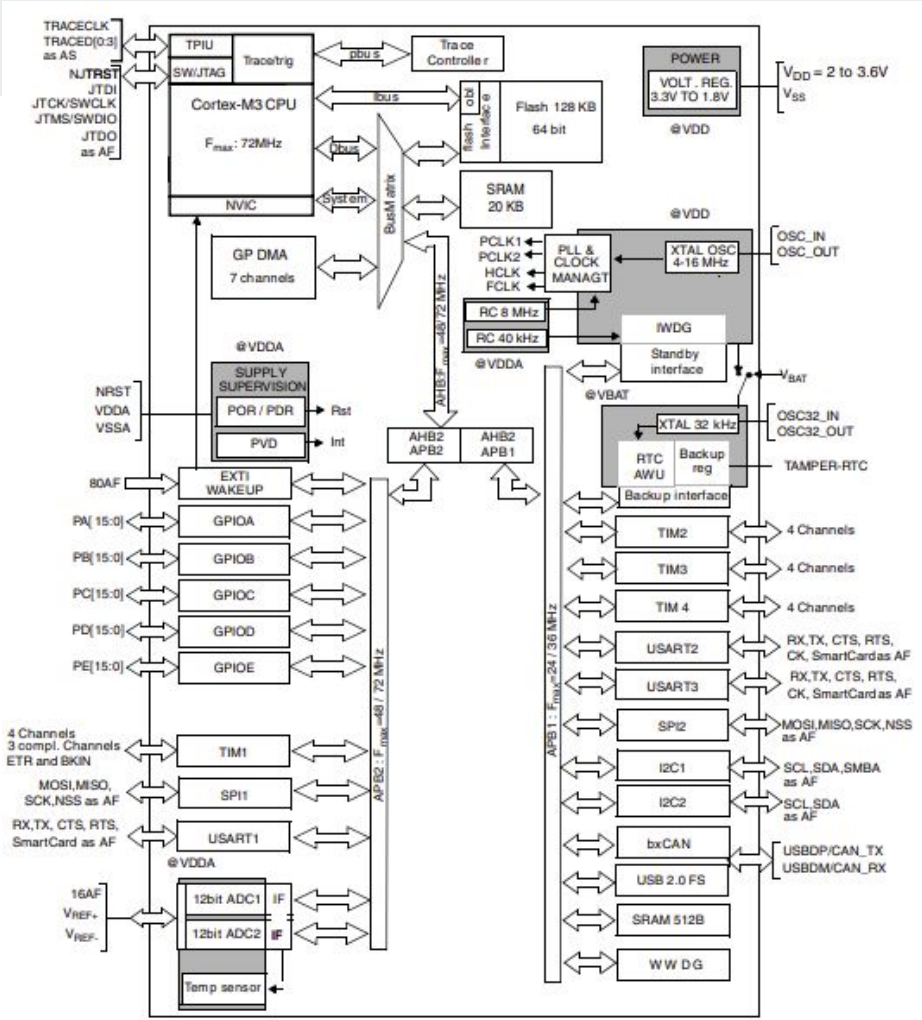

MB997A
www.st.com/stm32f4-discovery

S C

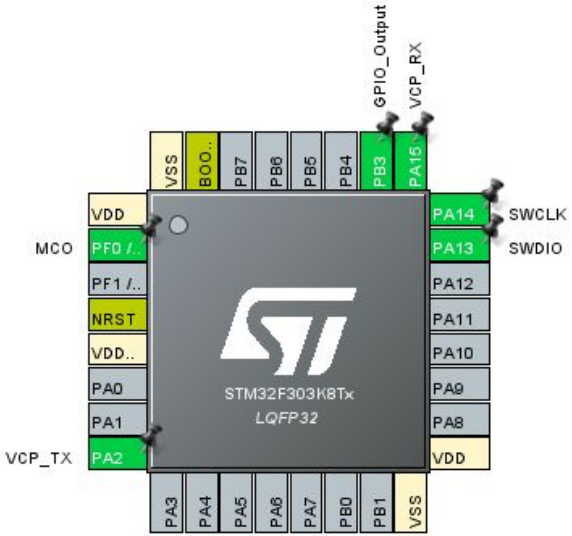

MB1136
rev C
NUCLEO
-F401RE
www.st.com/stm32nucleo


RESET

Architecture type



Premier exemple : Hello world !



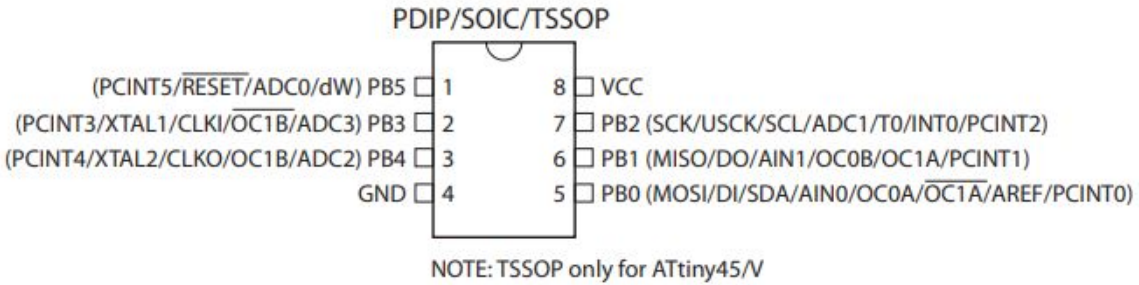

```c
/* Infinite loop */
/* USER CODE BEGIN WHILE */
while (1)
{
  /* USER CODE END WHILE */

  /* USER CODE BEGIN 3 */
    HAL_GPIO_TogglePin(GPIOB, GPIO_PIN_3);
    HAL_Delay(1000);
}
/* USER CODE END 3 */
```

ATTINY85

Quésaco ?

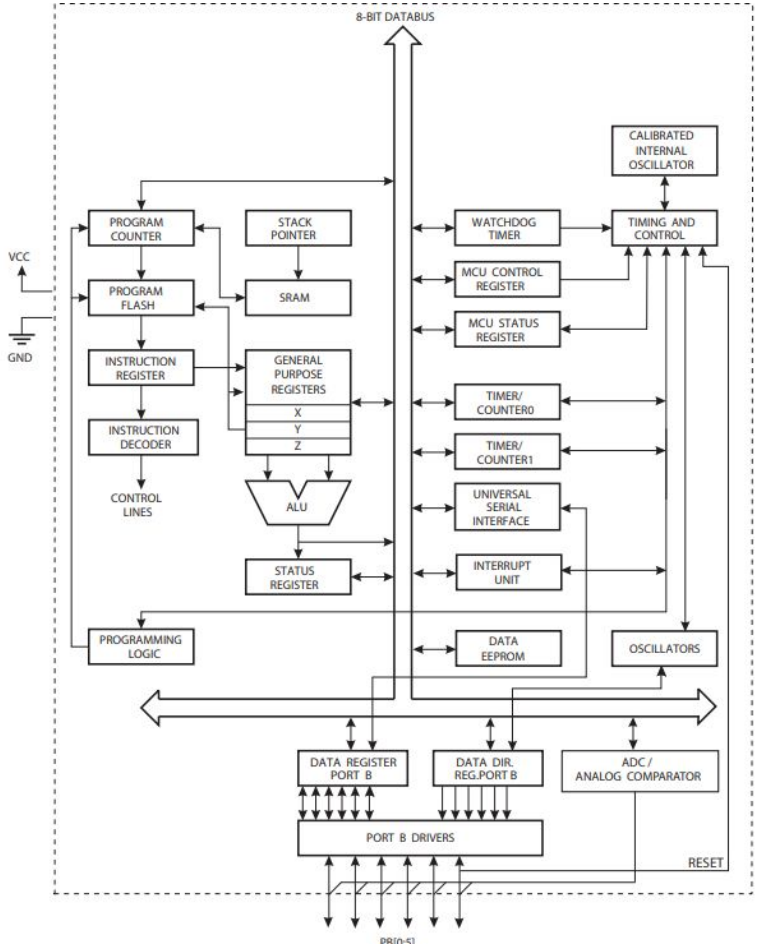
- Petit microcontrôleur de chez Microchip
- AVR 8 bit basé RISC
- Basique
  - 120 instructions
  - 32 x 8 registres



Caractéristiques

- 8 KB de Flash
- 20 MHz
- 512 bytes d’EEPROM
- 2 timers 8 bits
- 4 channel ADC
- 8 pins

Architecture



Registre

Page 200 de la datasheet

Address	Name	Bit 7	Bit 6	Bit 5	Bit 4	Bit 3	Bit 2	Bit 1	Bit 0	Page
0x3F	SREG	I	T	H	S	V	N	Z	C	page 8
0x3E	SPH	–	–	–	–	–	–	SP9	SP8	page 11
0x3D	SPL	SP7	SP6	SP5	SP4	SP3	SP2	SP1	SP0	page 11
0x3C	Reserved	–								
0x3B	GIMSK	–	INT0	PCIE	–	–	–	–	–	page 51
0x3A	GIFR	–	INTF0	PCIF	–	–	–	–	–	page 52
0x39	TIMSK	–	OCIE1A	OCIE1B	OCIE0A	OCIE0B	TOIE1	TOIE0	–	pages 81, 102
0x38	TIFR	–	OCF1A	OCF1B	OCF0A	OCF0B	TOV1	TOV0	–	page 81
0x37	SPMCSR	–	–	RSIG	CTPB	RFLB	PGWRT	PGERS	SPMEN	page 145
0x36	Reserved	–								
0x35	MCUCR	BODS	PUD	SE	SM1	SM0	BODSE	ISC01	ISC00	pages 37, 51, 64
0x34	MCUSR	–	–	–	–	WDRF	BORF	EXTRF	PORF	page 44,
0x33	TCCR0B	FOC0A	FOC0B	–	–	WGM02	CS02	CS01	CS00	page 79
0x32	TCNT0	Timer/Counter0								page 80
0x31	OSCCAL	Oscillator Calibration Register								page 31
0x30	TCCR1	CTC1	PWM1A	COM1A1	COM1A0	CS13	CS12	CS11	CS10	pages 89, 106
0x2F	TCNT1	Timer/Counter1								pages 91, 102
0x2E	OCR1A	Timer/Counter1 Output Compare Register A								pages 91, 102
0x2D	OCR1C	Timer/Counter1 Output Compare Register C								pages 91, 102
0x2C	GTCCR	TSM	PWM1B	COM1B1	COM1B0	FOC1B	FOC1A	PSR1	PSR0	pages 77, 90, 101
0x2B	OCR1B	Timer/Counter1 Output Compare Register B								page 92
0x2A	TCCR0A	COM0A1	COM0A0	COM0B1	COM0B0	–		WGM01	WGM00	page 77
0x29	OCR0A	Timer/Counter0 – Output Compare Register A								page 80
0x28	OCR0B	Timer/Counter0 – Output Compare Register B								page 81
0x27	PLLCSR	LSM	–	–	–	–	PCKE	PLLE	PLOCK	pages 94, 103
0x26	CLKPR	CLKPCE	–	–	–	CLKPS3	CLKPS2	CLKPS1	CLKPS0	page 32
0x25	DT1A	DT1AH3	DT1AH2	DT1AH1	DT1AH0	DT1AL3	DT1AL2	DT1AL1	DT1AL0	page 107
0x24	DT1B	DT1BH3	DT1BH2	DT1BH1	DT1BH0	DT1BL3	DT1BL2	DT1BL1	DT1BL0	page 107
0x23	DTPS1	-	-	-	-	-	-	DTPS11	DTPS10	page 106
0x22	DWDR	DWDR[7:0]								page 140
0x21	WDTCR	WDIF	WDIE	WDP3	WDCE	WDE	WDP2	WDP1	WDP0	page 45
0x20	PRR	–				PRTIM1	PRTIM0	PRUSI	PRADC	page 36
0x1F	EEARH								EEAR8	page 20
0x1E	EEARL	EEAR7	EEAR6	EEAR5	EEAR4	EEAR3	EEAR2	EEAR1	EEAR0	page 21
0x1D	EEDR	EEPROM Data Register								page 21
0x1C	EECR	–	–	EEPM1	EEPM0	EERIE	EEMPE	EEPE	EERE	page 21
0x1B	Reserved	–								
0x1A	Reserved	–								
0x19	Reserved	–								
0x18	PORTB	–	–	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	page 64
0x17	DDRB	–	–	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	page 64
0x16	PINB	–	–	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	page 64
0x15	PCMSK	–	–	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	page 52
0x14	DIDR0	–	–	ADC0D	ADC2D	ADC3D	ADC1D	AIN1D	AIN0D	pages 121, 138
0x13	GPIOR2	General Purpose I/O Register 2								page 10
0x12	GPIOR1	General Purpose I/O Register 1								page 10
0x11	GPIOR0	General Purpose I/O Register 0								page 10
0x10	USIBR	USI Buffer Register								page 115
0x0F	USIDR	USI Data Register								page 115
0x0E	USISR	USISIF	USIOIF	USIPF	USIDC	USICNT3	USICNT2	USICNT1	USICNT0	page 115
0x0D	USICR	USISIE	USIOIE	USIWM1	USIWM0	USICS1	USICS0	USICLK	USITC	page 116
0x0C	Reserved	–								
0x0B	Reserved	–								
0x0A	Reserved	–								
0x09	Reserved	–								
0x08	ACSR	ACD	ACBG	ACO	ACI	ACIE	–	ACIS1	ACIS0	page 120
0x07	ADMUX	REFS1	REFS0	ADLAR	REFS2	MUX3	MUX2	MUX1	MUX0	page 134
0x06	ADCSRA	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	page 136
0x05	ADCH	ADC Data Register High Byte								page 137
0x04	ADCL	ADC Data Register Low Byte								page 137
0x03	ADCSRB	BIN	ACME	IPR	–	–	ADTS2	ADTS1	ADTS0	pages 120, 137
0x02	Reserved	–								
0x01	Reserved	–								
0x00	Reserved	–								

Manipulations du jour

ATTINY85

- Programmation via ISP (In-System programming)
- Blink simple
- Blink avec les registres
- Génération de PWM

Lisez bien la datasheet

Des question ?