Carte ARDUINO UNO

Microcontrôleur ATMega328

B. Cottenceau B311 ISTIA
bertrand.cottenceau@univ-angers.fr

1 Introduction

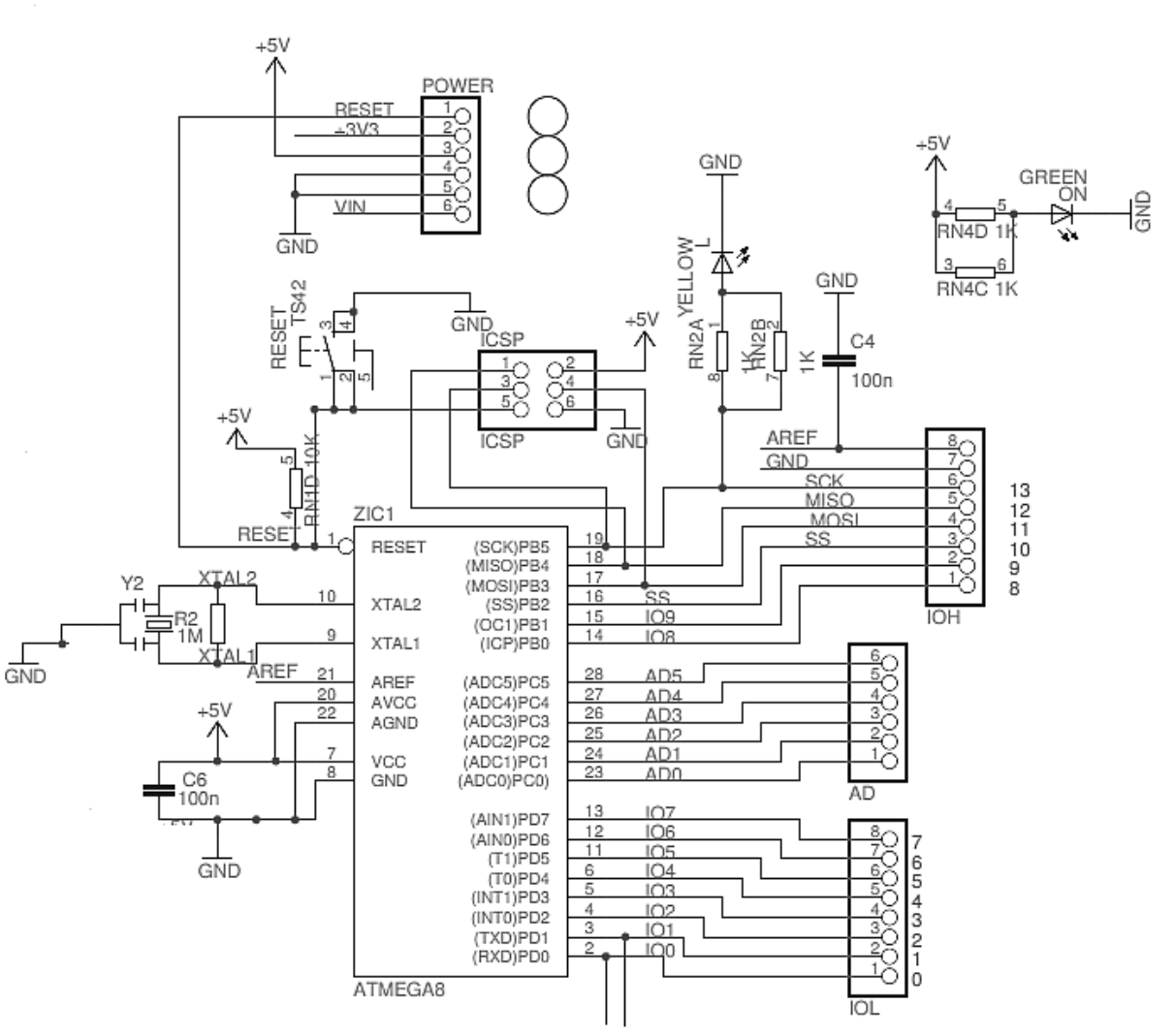
Le modèle UNO de la société ARDUINO est une carte électronique dont le coeur est un microcontrôleur ATMEL de référence ATMega328. Le microcontrôleur ATMega328 est un microcontrôleur 8bits de la famille AVR dont la programmation peut être réalisée en langage C.

L'intérêt principal des cartes ARDUINO (d'autres modèles existent) est leur facilité de mise en oeuvre. ARDUINO fournit un environnement de développement s'appuyant sur des outils open-source. Le chargement du programme dans la mémoire du microcontrôleur se fait de façon très simple par port USB. En outre, des bibliothèques de fonctions "clé en main" sont également fournies pour l'exploitation d'entrées-sorties courantes : gestion des E/S TOR, gestion des convertisseurs ADC, génération de signaux PWM, exploitation de bus TWI/I^{2}C, exploitation de servomoteurs ...

L'objectif de ce document est de mettre en évidence certaines informations techniques concernant l'exploitation des périphériques intégrés, en particulier lorsqu'on n'utilise pas les fonctions "clé en main" d'ARDUINO, dans l'objectif de comprendre comment ça marche !

2 Schéma simplifié de la carte Arduino UNO

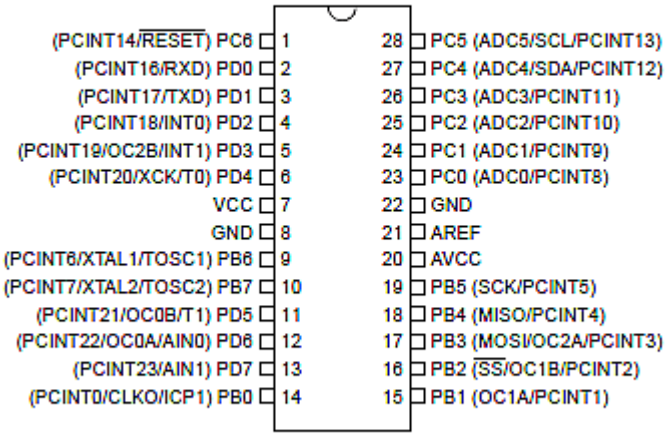
Les signaux d'entrée-sortie du microcontrôleur sont reliés à des connecteurs selon le schéma ci-dessous.



3 Microcontrôleur ATMEL ATMega328

Le microcontrôleur utilisé sur la carte Arduino UNO est un microcontrôleur **ATMega328**. C'est un microcontrôleur ATMEL de la famille AVR
8bits.

Les principales caractéristiques sont :



FLASH = mémoire programme de 32Ko
SRAM = données (volatiles) 2Ko
EEPROM = données (non volatiles) 1Ko

Digital I/O (entrées-sorties Tout Ou Rien) =
3 ports PortB, PortC, PortD
(soit 23 broches en tout I/O)

Timers/Counters : Timer0 et Timer2 (comptage 8 bits), Timer1 (comptage 16bits)
Chaque timer peut être utilisé pour générer deux signaux PWM. (6 broches OCxA/OCxB)

Plusieurs broches multi-fonctions : certaines broches peuvent avoir plusieurs fonctions différentes choisies par programmation

PWM = 6 broches **OC0A(PD6)**, **OC0B(PD5)**, **0C1A(PB1)**, **OC1B(PB3)**, **OC2A(PB3)**, **OC2B(PD3)**

Analog to Digital Converter (résolution 10bits) = 6 entrées multiplexées **ADC0(PC0)** à **ADC5(PC5)**

Gestion bus I2C (TWI Two Wire Interface) = le bus est exploité via les broches **SDA(PC5)/SCL(PC4).**

Port série (USART) = émission/réception série via les broches **TXD(PD1)/RXD(PD0)**

Comparateur Analogique = broches AIN0(PD6) et AIN1 (PD7) peut déclencher interruption

Watchdog Timer programmable.

Gestion d'interruptions (24 sources possibles (cf interrupt vectors)**) :** en résumé

- Interruptions liées aux entrées **INT0 (PD2) et INT1 (PD3)**
- Interruptions sur changement d'état des broches **PCINT0** à **PCINT23**
- Interruptions liées aux Timers 0, 1 et 2 (plusieurs causes configurables)
- Interruption liée au comparateur analogique
- Interruption de fin de conversion **ADC**
- Interruptions du port série **USART**
- Interruption du bus **TWI** (**I2C**)

4 Structure interne de l'ATMega328 (extraits de documentations ATMEL)

L'utilisation des périphériques intégrés (Entrées Sorties TOR, Timers, ...) repose sur l'exploitation (lecture/écriture) de registres internes. Ces registres, essentiellement 8 bits, sont décrits par un nom, y compris dans la programmation en C. Cette section fournit quelques détails importants sur les registres internes du microcontrôleur ATMega328.

Notation : par la suite, pour un registre nommé **R**, la notation **R.n** désigne le bit de rang n du registre R. Le bit R.0 est le bit de poids faible de R.

4.1 Status Register (SREG)

Le registre **SREG** contient des indicateurs liés aux opérations et le bit d'autorisation générale des interruptions. Les bits de ce registre sont : **Z** (Zero), **C** (Carry), **S** (Sign) ...
Le bit d'activation général du système d'interruptions est le bit **I** (**SREG.7**)

Note : en langage C, ce bit I est modifiable via les appels `sei()` (set IT) `cli()` (Clear IT)

SREG – AVR Status Register

The AVR Status Register – SREG – is defined as:

Bit	7	6	5	4	3	2	1	0	
0x3F (0x5F)	I	T	H	S	V	N	Z	C	SREG
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 7 – I: Global Interrupt Enable**

The Global Interrupt Enable bit must be set for the interrupts to be enabled. The individual interrupt enable control is then performed in separate control registers. If the Global Interrupt Enable Register is cleared, none of the interrupts are enabled independent of the individual interrupt enable settings. The I-bit is cleared by hardware after an interrupt has occurred, and is set by the RETI instruction to enable subsequent interrupts. The I-bit can also be set and cleared by the application with the SEI and CLI instructions, as described in the instruction set reference.

4.2 Digital I/O (Entrées Sorties Tout Ou Rien TOR)

Souvent, les microcontrôleurs disposent de broches d'entrée/sortie TOR, comme sur un automate programmable industriel (cf cours Automatismes Industriels). Pour placer l'état d'une sortie à 0 ou 1, ou lire l'état d'une entrée, il faut exploiter des registres internes, décrits ci-dessous.

Les entrées-sorties sont réparties dans 3 groupes de broches appelés *ports*. Le port B regroupe les broches notées PBx, le port C les broches PCx et le port D les broches PDx. Chaque port est configuré/expoité grâce à 3 registres.

PORTx = pour l'écriture de valeurs en sortie
DDRx = détermine la direction de chaque broche du port (1-sortie 0-entrée)
PINx = permet la lecture de la valeur en entrée

Chaque broche de port E/S a une résistance de pull-up interne qui peut être désactivée. Le bit **PUD** du registre MCUCR permet la désactivation des resistances de pull-up.

Direction des ports : si le bit **DDRB.2** vaut 1 alors la broche **PB2** est une sortie TOR.

Ecriture des sorties TOR : si une broche est configurée en sortie (DDRx.n=1) alors l'écriture du bit PORTx.n permet de définir l'état de la sortie (0 ou 1).

Ex : **DDRB.5=1** (donc **PB5 en sortie**) écrire 0 ou 1 dans le bit **PORTB.5** permet de définir l'état de la sortie **PB5**.

Lectures des entrées TOR : si une broche est configurée en entrée (DDRx.n=0) alors la lecture du bit PINx.n permet de connaître l'état de l'entrée.

Ex : **DDRB.4=0** (donc **PB4** en **entrée**), lire **PINB.4** permet de connaître l'état de l'entrée **PB4**.

MCUCR – MCU Control Register

Bit	7	6	5	4	3	2	1	0	
0x35 (0x55)	–	BODS	BODSE	PUD	–	–	IVSEL	IVCE	MCUCR
Read/Write	R	R	R	R/W	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 4 – PUD: Pull-up Disable**

When this bit is written to one, the pull-ups in the I/O ports are disabled even if the DDxn and PORTxn Registers are configured to enable the pull-ups ({DDxn, PORTxn} = 0b01). See "Configuring the Pin" on page 76 for more details about this feature.

PORTB – The Port B Data Register

Bit	7	6	5	4	3	2	1	0	
0x05 (0x25)	PORTB7	PORTB6	PORTB5	PORTB4	PORTB3	PORTB2	PORTB1	PORTB0	PORTB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

DDRB – The Port B Data Direction Register

Bit	7	6	5	4	3	2	1	0	
0x04 (0x24)	DDB7	DDB6	DDB5	DDB4	DDB3	DDB2	DDB1	DDB0	DDRB
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

PINB – The Port B Input Pins Address

Bit	7	6	5	4	3	2	1	0	
0x03 (0x23)	PINB7	PINB6	PINB5	PINB4	PINB3	PINB2	PINB1	PINB0	PINB
Read/Write	R	R	R	R	R	R	R	R	
Initial Value	N/A	N/A	N/A	N/A	N/A	N/A	N/A	N/A	

Gestion des résitances pull-up internes

En technologie MOS, une entrée "en l'air" a un état indeterminé. Lorsqu'on veut exploiter des boutons poussoir, on les branche de façon à ramener l'entrée à 0 quand on ferme le contact. Lorsque le contact est ouvert, l'état de l'entrée est ramené à 1 par des resistances de tirage à 1 (pull-up).

PORTx.n=1 ET **DDRx**.n=0 ALORS **pull-interne de Pxn activée**

PORTx.n=0 OU **DDRx**.n=1 ALORS pull-interne de Pxn désactivée

Si **PUD**=1 ALORS toutes les resistances de pull-up -interne de tous les ports désactivées

4.3 Sources d'interruption exploitables sur ATMega328 (carte Arduino UNO)

Le vecteur d'interruptions décrit toutes les sources pouvant, sous-réserve de bonne configuration, conduire à un déclenchement d'interruption.

11.4 Interrupt Vectors in ATmega328P

Table 11-6. Reset and Interrupt Vectors in ATmega328P

VectorNo.	Program Address(2)	Source	Interrupt Definition
1	0x0000(1)	RESET	External Pin, Power-on Reset, Brown-out Reset and Watchdog System Reset
2	0x0002	INT0	External Interrupt Request 0
3	0x0004	INT1	External Interrupt Request 1
4	0x0006	PCINT0	Pin Change Interrupt Request 0
5	0x0008	PCINT1	Pin Change Interrupt Request 1
6	0x000A	PCINT2	Pin Change Interrupt Request 2
7	0x000C	WDT	Watchdog Time-out Interrupt
8	0x000E	TIMER2 COMPA	Timer/Counter2 Compare Match A
9	0x0010	TIMER2 COMPB	Timer/Counter2 Compare Match B
10	0x0012	TIMER2 OVF	Timer/Counter2 Overflow
11	0x0014	TIMER1 CAPT	Timer/Counter1 Capture Event
12	0x0016	TIMER1 COMPA	Timer/Counter1 Compare Match A
13	0x0018	TIMER1 COMPB	Timer/Coutner1 Compare Match B
14	0x001A	TIMER1 OVF	Timer/Counter1 Overflow
15	0x001C	TIMER0 COMPA	Timer/Counter0 Compare Match A
16	0x001E	TIMER0 COMPB	Timer/Counter0 Compare Match B
17	0x0020	TIMER0 OVF	Timer/Counter0 Overflow
18	0x0022	SPI, STC	SPI Serial Transfer Complete
19	0x0024	USART, RX	USART Rx Complete
20	0x0026	USART, UDRE	USART, Data Register Empty
21	0x0028	USART, TX	USART, Tx Complete
22	0x002A	ADC	ADC Conversion Complete

Table 11-1. Reset and Interrupt Vectors in ATmega48PA (Continued)

Vector No.	Program Address	Source	Interrupt Definition
24	0x017	ANALOG COMP	Analog Comparator
25	0x018	TWI	2-wire Serial Interface
26	0x019	SPM READY	Store Program Memory Ready

4.3.1 Interruptions Externes

Il s'agit d'interruptions liées à des niveaux ou des changements d'états de broches du microcontrôleur. Les broches impliquées doivent être configurées en entrée (**cf. 4.2 DIGITAL I/O**).

Broches INT0 (PD2)/INT1(PD3) : configurables pour déclencher les interruptions (n° 2 et 3) sur niveau 0, front négatif ou positif.

La cause d'interruption est choisie par le registre EICRA

EICRA – External Interrupt Control Register A

The External Interrupt Control Register A contains control bits for interrupt sense control.

Bit	7	6	5	4	3	2	1	0	
(0x69)	–	–	–	–	ISC11	ISC10	ISC01	ISC00	EICRA
Read/Write	R	R	R	R	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Table 12-2. Interrupt 0 Sense Control

ISC01	ISC00	Description
0	0	The low level of INT0 generates an interrupt request.
0	1	Any logical change on INT0 generates an interrupt request.
1	0	The falling edge of INT0 generates an interrupt request.
1	1	The rising edge of INT0 generates an interrupt request.

Note : Pour l'interruption INT1, les bits ISC11 et ISC10 ont le même sens que ci-dessus.

Activation des interruptions INT0/INT1 = bit SREG.7 (I)=1 et mise à 1 de EIMSK.0/EIMSK.1

EIMSK – External Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
0x1D (0x3D)	–	–	–	–	–	–	INT1	INT0	EIMSK
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Flags internes = lorsqu'une cause d'IT est détectée, un flag interne de EIFR est positionné

EIFR – External Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x1C (0x3C)	–	–	–	–	–	–	INTF1	INTF0	EIFR
Read/Write	R	R	R	R	R	R	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Exemple : on souhaite qu'un front négatif sur **INT1** (PD3) conduise à une interruption **INT1**. Il faut donc

SREG.7=1 (bit général d'activation des IT, sans lui aucune IT)
EIMSK.1 (INT1)= 1 (pour prise en compte des IT INT1)
EICRA.2(ISC10)=0 ET EICRA.3(ISC11)=1 (IT sur front négatif sur INT1)

Broches PCINT0 à PCINT23 : configurables pour déclencher des interruptions (n° 4,5 et 6) sur des changements d'état ("Pin Change") des broches (configurées en entrée DDRx.n=1). Les broches sont séparées en 3 sous-groupes, il y a une source d'interruption par sous-groupe, et pour chaque broche on peut activer ou non le système "Pin Change Interrupt"

L'interruption PCI2 (IT n°6) a lieu sur changement d'état de l'une des broches PCINT23..16, l'interruption PCI1 (IT n°5) sur les changements d'état de PCINT14..8, et l'interruption PCI0 (IT n°4) a lieu sur les changements d'état des broches PCINT7..0. Les registres **PCMSK2**, **PCMSK1** and **PCMSK0** contrôlent quelles broches peuvent conduire (ou non) à une interruption de type "pin change".

Activation des interruptions PCINT0 à PCINT23 = bit SREG.7 (I)=1 et mise à 1 de PCIEx

PCICR – Pin Change Interrupt Control Register

Bit	7	6	5	4	3	2	1	0	
(0x68)	–	–	–	–	–	PCIE2	PCIE1	PCIE0	PCICR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Configuration détaillée : le registre PCMSKx détermine si les broches du sous-groupe x sont prises en compte pour l'interruption "pin change"

PCMSK0 – Pin Change Mask Register 0

Bit	7	6	5	4	3	2	1	0	
(0x6B)	PCINT7	PCINT6	PCINT5	PCINT4	PCINT3	PCINT2	PCINT1	PCINT0	PCMSK0
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

PCMSK1 – Pin Change Mask Register 1

Bit	7	6	5	4	3	2	1	0	
(0x6C)	–	PCINT14	PCINT13	PCINT12	PCINT11	PCINT10	PCINT9	PCINT8	PCMSK1
Read/Write	R	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

PCMSK2 – Pin Change Mask Register 2

Bit	7	6	5	4	3	2	1	0	
(0x6D)	PCINT23	PCINT22	PCINT21	PCINT20	PCINT19	PCINT18	PCINT17	PCINT16	PCMSK2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Flags internes pour les IT "Pin Change"

PCIFR – Pin Change Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x1B (0x3B)	–	–	–	–	–	PCIF2	PCIF1	PCIF0	PCIFR
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Exemple : on souhaite qu'un changement d'état de la broche **PCINT2** (PB2) conduise à une interruption **PCIE0**. Il faut donc

SREG.7=1 (bit général d'activation des IT, sans lui aucune IT)
PCICR.0 = PCIE0=1 (pour que le sous-groupe 0 puisse conduire à IT)
PCMSK0.2 = PCINT2 = 1 (pour que la broche PCINT2 soit prise en compte)

4.3.2 Interruptions Timers

Plusieurs sources d'interruptions sont liées à un même timer.

Les indicateurs internes liés aux interruptions timers 0 sont dans le registre TIFR0

TIFR0 – Timer/Counter 0 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x15 (0x35)	–	–	–	–	–	OCF0B	OCF0A	TOV0	TIFR0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

Activation des interruptions Timer 0 (3 sources n°14, 15 et 16)

TIMSK0 – Timer/Counter Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x6E)	–	–	–	–	–	OCIE0B	OCIE0A	TOIE0	TIMSK0
Read/Write	R	R	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

- **Bit 2 – OCIE0B: Timer/Counter Output Compare Match B Interrupt Enable**

When the OCIE0B bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter Compare Match B interrupt is enabled. The corresponding interrupt is executed if a Compare Match in Timer/Counter occurs, i.e., when the OCF0B bit is set in the Timer/Counter Interrupt Flag Register – TIFR0.

- **Bit 1 – OCIE0A: Timer/Counter0 Output Compare Match A Interrupt Enable**

When the OCIE0A bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter0 Compare Match A interrupt is enabled. The corresponding interrupt is executed if a Compare Match in Timer/Counter0 occurs, i.e., when the OCF0A bit is set in the Timer/Counter 0 Interrupt Flag Register – TIFR0.

- **Bit 0 – TOIE0: Timer/Counter0 Overflow Interrupt Enable**

When the TOIE0 bit is written to one, and the I-bit in the Status Register is set, the Timer/Counter0 Overflow interrupt is enabled. The corresponding interrupt is executed if an overflow in Timer/Counter0 occurs, i.e., when the TOV0 bit is set in the Timer/Counter 0 Interrupt Flag Register – TIFR0.

Remarque : pour les Timers 1 et 2, les configurations sont similaires.

Exemple : on souhaite que chaque débordement du Timer 0 conduise à une interruption.

SREG.7=1 (bit général d'activation des IT, sans lui aucune IT)
TIMSK0.0 (TOIE0)=1 (interruption sur débordement du timer 0)

4.4 Timers/Counters de ATMega328

Les microcontrôleurs AVR disposent de modules de temporisation/comptage internes, fonctionnant pour certains avec des registres de comptage sur 8 bits, et pour d'autres sur 16 bits. Dans tous les cas, chaque événement de comptage conduit à une modification du registre de comptage (+1 ou -1). L'événement de comptage peut être un "tick" de l'horloge du microcontrôleur, ce qui revient à mesurer l'écoulement du temps. L'évenement de comptage peut aussi être un front sur une broche d'entrée du microcontrôleur (les broches T0 et T1 peuvent servir d'entrée de comptage).

Fonction Temporisateur : lorsque l'on compte des "ticks" de l'horloge qui cadence le microcontrôleur, on mesure du temps. Les modules Timers/Counters permettent de compter les ticks du signal d'horologe ou un signal de fréquence plus faible obtenu par un diviseur appelé **prescaler**.

Fonction Compteur : lorsque l'on compte des fronts sur une entrée de comptage (broches T0 ou T1), on utilise alors la fonction "compteur" du module.

Le choix entre fonction de temporisation (avec prédiviseur de fréquence ou non) et fonction de comptage se fait par paramétrage de registres dédiés à la gestion des modules Timers/Counters.

Génération de signaux périodiques : les modules Timers/Counters sont assez complexes et chacun de ces modules peut générer deux signaux PWM dont le rapport cyclique est facilement modifiable.

Remarque : ces périphériques intégrés sont assez complexes (environ 70 pages du datasheet ATMega). Seule une vision simplifiée est fournie ici.

4.4.1 Timer/Counter 0 (comptage 8 bits)

C'est un module Timer/Counter avec registre de comptage 8 bits. En utilisant l'IDE Arduino, le timer 0 est implicitement utilisé par les fonctions de delay (ainsi que l'interruption correspondante). Ce module Timer/Counter n'est donc pas utilisable directement avec la carte ARDUINO.

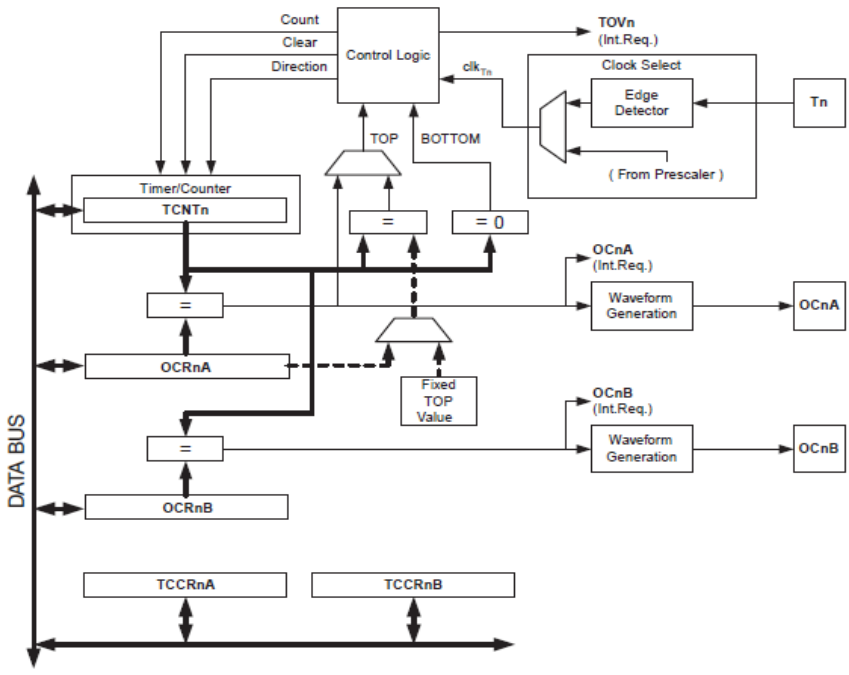
4.4.2 Timer/Counter 2 (comptage 8 bits)

C'est un module Timer/Counter avec registre de comptage 8 bits.
Note : le bit PRTIM2 (registre PRR bit 6) doit être à 0 pour activer le module Timer/Counter2

La structure générale du module Timer/Counter 0 est donnée ci-dessous. Le registre de comptage est TCNT2 (registre 8 bits).

Points importants :

- détection et IT sur débordement
- entrée de comptage interne = signal d'horloge interne avec prédivision ou non
- possibilité de comparer TCNT2 à deux registres de comparaison OCR2A/OCR2B
- l'égalité TCTN2=OCR2A peut déclencher une IT
- l'égalité TCTN2=OCR2B peut déclencher une IT également
- Les broches OC2A(PB3) et OC2B (PD3) peuvent être activées par le Timer/Counter 2 pour génération de signaux périodiques (PWM).



Registres du module Timer/Counter 2

TCNT2 – Timer/Counter Register

Bit	7	6	5	4	3	2	1	0	
(0xB2)	TCNT2[7:0]								TCNT2
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

OCR2A – Output Compare Register A

Bit	7	6	5	4	3	2	1	0	
(0xB3)	OCR2A[7:0]								OCR2A
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

The Output Compare Register A contains an 8-bit value that is continuously compared with the counter value (TCNT2). A match can be used to generate an Output Compare interrupt, or to generate a waveform output on the OC2A pin.

OCR2B – Output Compare Register B

Bit	7	6	5	4	3	2	1	0	
(0xB4)	OCR2B[7:0]								OCR2B
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

TCCR2A – Timer/Counter Control Register A

Bit	7	6	5	4	3	2	1	0	
(0xB0)	COM2A1	COM2A0	COM2B1	COM2B0	–	–	WGM21	WGM20	TCCR2A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	

TCCR2B – Timer/Counter Control Register B

Bit	7	6	5	4	3	2	1	0	
(0xB1)	FOC2A	FOC2B	–	–	WGM22	CS22	CS21	CS20	TCCR2B
Read/Write	W	W	R	R	R	R	R/W	R/W	

TIMSK2 – Timer/Counter2 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x70)	–	–	–	–	–	OCIE2B	OCIE2A	TOIE2	TIMSK2
Read/Write	R	R	R	R	R	R/W	R/W	R/W	

TIFR2 – Timer/Counter2 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x17 (0x37)	–	–	–	–	–	OCF2B	OCF2A	TOV2	TIFR2
Read/Write	R	R	R	R	R	R/W	R/W	R/W	

Les bits WGM22:20 définissent le mode de fonctionnement du module.

Table 17-8. Waveform Generation Mode Bit Description

Mode	WGM2	WGM1	WGM0	Timer/Counter Mode of Operation	TOP	Update of OCRx at	TOV Flag Set on[1][2]
0	0	0	0	Normal	0xFF	Immediate	MAX
1	0	0	1	PWM, Phase Correct	0xFF	TOP	BOTTOM
2	0	1	0	CTC	OCRA	Immediate	MAX
3	0	1	1	Fast PWM	0xFF	BOTTOM	MAX
4	1	0	0	Reserved	–	–	–
5	1	0	1	PWM, Phase Correct	OCRA	TOP	BOTTOM
6	1	1	0	Reserved	–	–	–
7	1	1	1	Fast PWM	OCRA	BOTTOM	TOP

Notes: 1. MAX= 0xFF
2. BOTTOM= 0x00

Selon le mode choisi par les bits WGM0:2 on a les options suivantes (mode PWM Phase correct non décrit)

Bits 7:6 – COM2A1:0: Compare Match Output A Mode. Ces bits configurent le rôle de la broche OC2A/PB3. Si l'un des bits COM2A1:0 vaut 1, la broche OC2A a une fonction alternative (associée au module Timer/Counter 2). Néanmoins, le registre DDR doit être tel que OC2A/PB3 **soit une sortie (cf 4.2).**

Table 17-2 : Mode Normal ou CTC (non-PWM mode)

Table 17-2. Compare Output Mode, non-PWM Mode

COM2A1	COM2A0	Description
0	0	Normal port operation, OC0A disconnected.
0	1	Toggle OC2A on Compare Match
1	0	Clear OC2A on Compare Match
1	1	Set OC2A on Compare Match

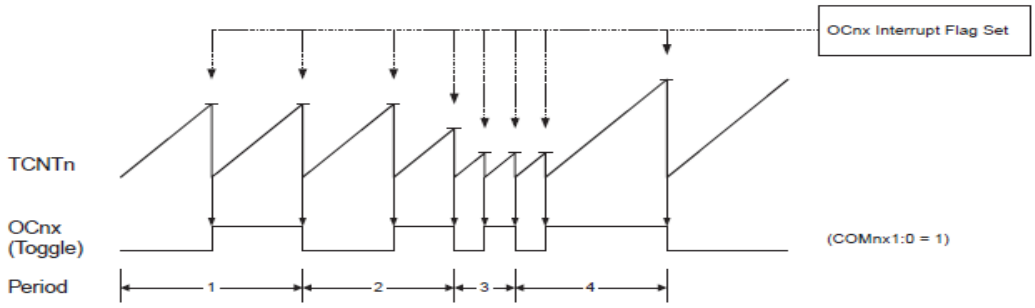
Table 17-3 : Fast-PWM mode

Table 17-3. Compare Output Mode, Fast PWM Mode[1]

COM2A1	COM2A0	Description
0	0	Normal port operation, OC2A disconnected.
0	1	WGM22 = 0: Normal Port Operation, OC0A Disconnected. WGM22 = 1: Toggle OC2A on Compare Match.
1	0	Clear OC2A on Compare Match, set OC2A at BOTTOM, (non-inverting mode).
1	1	Set OC2A on Compare Match, clear OC2A at BOTTOM, (inverting mode).

Clear Timer on Compare Match (CTC) Mode
En mode CTC (WGM22:0 = 2), le registre OCR2A règle la résolution. Le compteur TCTN2 est remis à zéro après l'égalité (match) TCTN2=OCR2A. Le registre OCR2A definit la valeur maximale pour le compteur, et donc sa résolution. On peut configurer le module 2 pour inverser l'état de la sortie OC2A (PB3) il faut alors (COM2A1:0 = 1).

Figure 17-5. CTC Mode, Timing Diagram



Prescaler Timer 2 (réglage de la vitesse de débordement en fonction de l'horloge)

Table 17-9. Clock Select Bit Description

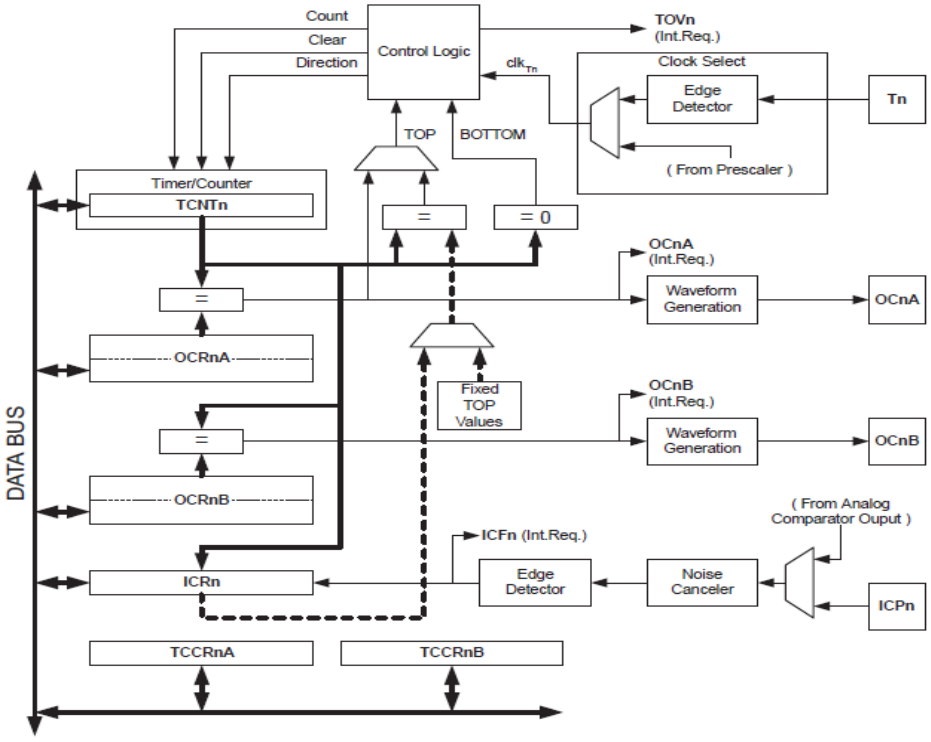
CS22	CS21	CS20	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	clk_{T2S}/(No prescaling)
0	1	0	clk_{T2S}/8 (From prescaler)
0	1	1	clk_{T2S}/32 (From prescaler)
1	0	0	clk_{T2S}/64 (From prescaler)
1	0	1	clk_{T2S}/128 (From prescaler)
1	1	0	clk_{T2S}/256 (From prescaler)
1	1	1	clk_{T2S}/1024 (From prescaler)

4.4.3 Timer/Counter 1 (comptage 16 bits)

Le registre de comptage TCNT1, ainsi que les registres de comparaison OCR1A et OCR1B, sont cette fois-ci sur 16 bits.

Note: en langage d'assemblage, il faut deux accès 8 bits pour lire/écrire ces registres 16 bits. En langage C, on peut manipuler symboliquement des données 16 bits via les symboles TCNT1, OCR1A et OCR1B sans se soucier de la façon dont le code sera généré.

Figure 15-1. 16-bit Timer/Counter Block Diagram[1]



Registres du module Timer/Counter 1

TCNT1H and TCNT1L – Timer/Counter1

Bit	7	6	5	4	3	2	1	0	
(0x85)	TCNT1[15:8]								TCNT1H
(0x84)	TCNT1[7:0]								TCNT1L
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

OCR1AH and OCR1AL – Output Compare Register 1 A

Bit	7	6	5	4	3	2	1	0	
(0x89)	OCR1A[15:8]								OCR1AH
(0x88)	OCR1A[7:0]								OCR1AL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

OCR1BH and OCR1BL – Output Compare Register 1 B

Bit	7	6	5	4	3	2	1	0	
(0x8B)	OCR1B[15:8]								OCR1BH
(0x8A)	OCR1B[7:0]								OCR1BL
Read/Write	R/W	R/W	R/W	R/W	R/W	R/W	R/W	R/W	

TIMSK1 – Timer/Counter1 Interrupt Mask Register

Bit	7	6	5	4	3	2	1	0	
(0x6F)	–	–	ICIE1	–	–	OCIE1B	OCIE1A	TOIE1	TIMSK1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	

TIFR1 – Timer/Counter1 Interrupt Flag Register

Bit	7	6	5	4	3	2	1	0	
0x16 (0x36)	–	–	ICF1	–	–	OCF1B	OCF1A	TOV1	TIFR1
Read/Write	R	R	R/W	R	R	R/W	R/W	R/W	

TCCR1A – Timer/Counter1 Control Register A

Bit	7	6	5	4	3	2	1	0	
(0x80)	COM1A1	COM1A0	COM1B1	COM1B0	–	–	WGM11	WGM10	TCCR1A
Read/Write	R/W	R/W	R/W	R/W	R	R	R/W	R/W	

TCCR1B – Timer/Counter1 Control Register B

Bit	7	6	5	4	3	2	1	0	
(0x81)	ICNC1	ICES1	–	WGM13	WGM12	CS12	CS11	CS10	TCCR1B
Read/Write	R/W	R/W	R	R/W	R/W	R/W	R/W	R/W	

Selon le mode choisi par les bits WGM10:3 on a les options suivantes (mode PWM Phase correct non décrit)

Table 15-4. Waveform Generation Mode Bit Description(1)

Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	0x03FF	TOP	BOTTOM
4	0	1	0	0	CTC	OCR1A	Immediate	MAX
5	0	1	0	1	Fast PWM, 8-bit	0x00FF	BOTTOM	TOP
6	0	1	1	0	Fast PWM, 9-bit	0x01FF	BOTTOM	TOP
7	0	1	1	1	Fast PWM, 10-bit	0x03FF	BOTTOM	TOP
8	1	0	0	0	PWM, Phase and Frequency Correct	ICR1	BOTTOM	BOTTOM
9	1	0	0	1	PWM, Phase and Frequency Correct	OCR1A	BOTTOM	BOTTOM
10	1	0	1	0	PWM, Phase Correct	ICR1	TOP	BOTTOM
11	1	0	1	1	PWM, Phase Correct	OCR1A	TOP	BOTTOM
12	1	1	0	0	CTC	ICR1	Immediate	MAX
13	1	1	0	1	(Reserved)	–	–	–
14	1	1	1	0	Fast PWM	ICR1	BOTTOM	TOP
15	1	1	1	1	Fast PWM	OCR1A	BOTTOM	TOP

Note: 1. The CTC1 and PWM11:0 bit definition names are obsolete. Use the WGM12:0 definitions. However, the functionality and location of these bits are compatible with previous versions of the timer.

Table 15-1 : Mode Normal ou CTC (non-PWM mode)

Table 15-1. Compare Output Mode, non-PWM

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	Toggle OC1A/OC1B on Compare Match.
1	0	Clear OC1A/OC1B on Compare Match (Set output to low level).
1	1	Set OC1A/OC1B on Compare Match (Set output to high level).

Table 15-2 : Fast-PWM mode

Table 15-2. Compare Output Mode, Fast PWM(1)

COM1A1/COM1B1	COM1A0/COM1B0	Description
0	0	Normal port operation, OC1A/OC1B disconnected.
0	1	WGM13:0 = 14 or 15: Toggle OC1A on Compare Match, OC1B disconnected (normal port operation). For all other WGM1 settings, normal port operation, OC1A/OC1B disconnected.
1	0	Clear OC1A/OC1B on Compare Match, set OC1A/OC1B at BOTTOM (non-inverting mode)
1	1	Set OC1A/OC1B on Compare Match, clear OC1A/OC1B at BOTTOM (inverting mode)

Note: 1. A special case occurs when OCR1A/OCR1B equals TOP and COM1A1/COM1B1 is set. In this case the compare match is ignored, but the set or clear is done at BOTTOM. See Section "15.9.3" on page 126. for more details.

Prescaler Timer 1 (réglage de la vitesse de débordement en fonction de l'horloge)

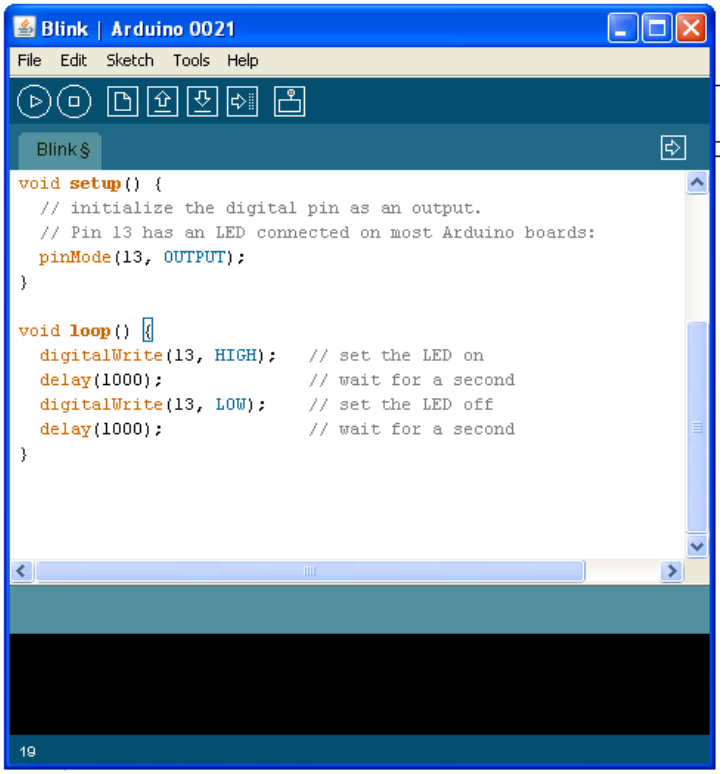
Table 15-5. Clock Select Bit Description

CS12	CS11	CS10	Description
0	0	0	No clock source (Timer/Counter stopped).
0	0	1	$clk_{I/O}$/1 (No prescaling)
0	1	0	$clk_{I/O}$/8 (From prescaler)
0	1	1	$clk_{I/O}$/64 (From prescaler)
1	0	0	$clk_{I/O}$/256 (From prescaler)
1	0	1	$clk_{I/O}$/1024 (From prescaler)
1	1	0	External clock source on T1 pin. Clock on falling edge.
1	1	1	External clock source on T1 pin. Clock on rising edge.

5 Exemples

On fournit ici quelques exemples simples testés avec la carte ARDUINO UNO.

5.1. Caractéristiques du développement ARDUINO



ARDUINO fournit un environnement de développement avec un éditeur de source, les opérations de compilation et de chargement dans la mémoire du microcontrôleur étant ramenées à des clicks sur des boutons dans l'IHM (très simple).

La communication entre le PC et la carte se fait via le port USB, moyennant installation d'un driver adapté (fourni par ARDUIINO).

Structure d'un projet ARDUINO

L'outil impose de structurer l'application de façon spécifique. Le compilateur utilisé est AVR GCC (compilateur C/C++ pour processeur AVR). La particularité est que une structure de programme est imposée au programmeur. La fonction main() est imposée, non modifiable, et décrite ci-dessous. En d'autres termes, les seules parties que l'on développe spécifiquement sont :

- la fonction setup() : doit contenir les initialisations (times, interrupts...)
- la fonction loop() : fonction répétée indéfiniment

```
#include <WProgram.h>

int main(void)
{
     init();   // initialisations ARDUINO pour fonctions
               // utiles : delays, PWM ...
     setup();
     for (;;)  loop();
     return 0;
}
```

UN PROGRAMME ARDUINO = une fonction setup() + une fonction loop()

5.2. Exemples ARDUINO UNO

Types/Taille :

boolean : true/false (8 bits)
char = entier 8 bits signé [-128,+127] (stocke aussi des codes ASCII)
byte / **unsigned char** : entiers 8 bits non signés [0,255]
Note: des constantes sont définies et permettent d'écrire `byte b = B10010;`
int : entiers 16 bits signés [-32768,+32767]
word / **unsigned int** : entier 16 bits non signé [0,65535]
long : entiers 32 bits signé
unsigned long : entiers 32 bits non signés
float /double : flottant format IEEE 32 bits (float et double = même type ici)

```
void setup()
{
   pinMode(13, OUTPUT);  //broche PB5 en sortie
}

void loop()
{
  digitalWrite(13,HIGH);  // set PB5
  delay(200);      // délai 200 ms
  digitalWrite(13,LOW);      // clear PB5
  delay(1000);     // délai 1 s
}
```

L'exemple le plus simple, fourni par ARDUINO, consiste à faire clignoter la LED raccordée à la broche PB5 du microcontrôleur, broche également disponible sur les connecteurs de la carte et portant le numéro 13.

La fonction setup() configure la broche PB5 (connexion n°13 sur la carte) en sortie.
La fonction loop() décrit ce qui sera répété indéfiniment : mettre PB5 à 1 pendant 200ms puis mettre PB5 à 0 pendant 1s, et ainsi de suite.

Fonctions utiles fournies par ARDUINO

Constantes : **HIGH, LOW, INPUT, OUTPUT**

Digital I/O

pinMode(pin,mode) : pin = numéro sur connecteur, mode = INPUT/OUTPUT
digitalWrite(pin,value) : pin = numéro sur connecteur, value= HIGH/LOW
int digitalRead(pin) : pin = numéro sur connecteur, retourne la valeur

Temporisations

delay(unsigned long ms) : delai de ms milli secondes avec ms sur 32 bits
delayMicroseconds(unsigned int ms) : delai de ms microsecondes avec ms sur 16 bits
unsigned long micros() : nombre de micro secondes depuis démarrage. Revient à zéro tous les 70mn environ.
unsigned long millis() : nombre de milli secondes depuis démarrage. Revient à zéro tous les 50 jours environ.

Exploitation des registres de l'ATMega328

Faire clignoter la LED (PB5) sans utiliser les fonctions ARDUINO (pour la gestion des broches I/O) nécessite l'accès aux registres de configuration des ports E/S, voir section 4.2.

Tous les registres sont accessibles via leur nom en majuscule.

```
// Exemple ARDUINO UNO
//    LED <- PB5 : LED allumée quand PB5=1

void setup() {
  // configurer broche PB5 en sortie
  DDRB |= B100000;   // B100000 représente la constante 0x20
      // OU DDRB|=0x20;  // équivalent
      // OU DDRB |= 32;  // équivalent quoique moins clair
  PORTB &= ~(1<<PORTB5);   // PORTB.5 <- 0
}

void loop() {
  PORTB |= (1<<PORTB5);  // PORTB.5 <- 1
  delay(200); // 200 ms
  PORTB &= ~(1<<PORTB5); // PORTB.5 <- 0
  delay(1000); // 1s
}
```

Remarques :

Les exemples ATMega328 en langage C trouvés sur internet utilisent des "styles" d'écriture parfois déroutants. Notamment, lors de la gestion des ports I/O, on doit souvent faire des opérations logiques (&,|,~,^) pour mettre des bits à 0, 1 ou inverser des bits.

Certaines écritures peuvent être déroutantes, par exemple

```
PORTB &= ~(1<<PORTB5);  // Mettre le bit 5 de PORTB à 0
```

Explication : PORTB est une contante qui vaut 5.

1<<PORTB5 signifie "décaler la valeur (0000 0001)b de 5 bits vers la gauche"

Donc, (1<<PORTB5) représente (0010 0000)b

Ensuite ~(1<<PORTB5) représente l'inverse bit à bit soit (1101 1111)b

Enfin, PORTB &= ~(1<<PORTB5) signifie PORTB = PORTB & B11011111

L'opération ET logique laisse inchangés tous les bits de PORTB **sauf le bit 5** qui est mis à 0 (en raison du bit à 0 de la valeur binaire (1101 1111)b = ~(1<<PORTB5)).

Exploitation des timers et des interruptions
L'exemple suivant illustre l'utilisation du timer 2 et de l'interruption associée à son débordement.

```
// ARDUINO UNO - IT Timer 2
volatile unsigned char cpt=0; // compteur d'IT

// Fonction de traitement IT n°10 = Timer 2 OverFlow
ISR(TIMER2_OVF_vect){
      cpt++;
      if(cpt==61){
            PORTB ^=(1<<PORTB5);
            cpt=0;
      }
}

void setup(){
      // Configuration Timer 2
      TCCR2A=0;        // Mode Normal (pas PWM)
      TCCR2B=B111;     // Prescaler 1024 (Clock/1024)
      TIMSK2=0x01;     // IT Timer2 Over Flow Active
      //Configuration PORTB.5
      DDRB |= B100000;       // PB5 en sortie
      PORTB &= ~(1<<PORTB5); // PORTB.5 <-0

      sei();      // activation des IT (SREG.7=1)
}

void loop() {    // aucun traitement          }
```

Paramètres Timer 2 (voir section 4.4.2) :
Mode 0 (Normal) : WGM2=0 WGM1=0 WGM0=0 [TCCR2A=0]
Prescaler = 1024 : CS22=1 CS21=1 CS20=1 [TCCR2B=0x07=(111)b]

Interruptions
Masque d'IT : Interruption sur Overflow = TIMSK2.0 =1
Autorisation générale des IT : SREG.7=1 [activé par sei()]

Digital I/O = Broche PORTB.5 en sortie.

Principe : après la fonction setup(), le registre **TCNT2** (8bits) est incrémenté à chaque tick du signal périodique clock/1024. A chaque débordement du registre TCNT2, le débordement déclenche l'interruption n°10 appelée "Timer 2 Over Flow". Tous les 60 appels de cette fonction, la broche PB5 (LED) change d'état. La LED clignote donc.

Quelle fréquence de clignotement ?
Clock = signal périodique de 16MegaHz (16 millions de ticks par seconde)
Prescaler = 1024 -> la fréquence d'incrément de TCNT2 = (16/1024) MegaHz
Fréquence de débordement : TCNT2 ne déborde que tous les 256 incréments
c'est-à-dire à la fréquence de 16/(1024*256) MegaHZ ≈ 61 Hz
Il y a 1 interruption Timer2 Over Flow tous les 1/61 secondes.
Il faut 61 Interruptions pour que la LED change d'état

La LED change d'état (0->1 1->0) à intervalles de environ 1 seconde

Exploitation des timers et des interruptions

On utilise ici le mode CTC du Timer 2. A chaque fois que TCNT2=CR2A, TCNT2 est remis à 0 et l'égalité déclenche une interruption.

```
// ARDUINO UNO - IT Timer 2
// Mode Clear Timer On Compare

volatile unsigned char cpt;

// Fonction de traitement IT n°10 = Timer 2 OverFlow
ISR(TIMER2_COMPA_vect){
      cpt++;
        if(cpt==40) PORTB ^=(1<<PORTB5);
        if(cpt==50){
            PORTB ^=(1<<PORTB5);
            cpt=0;
         }
}

void setup(){
      // Configuration Timer 2
      TCCR2A=B010;    // Mode CTC (Clear Timer On Compare)
      OCR2A=156;      // Registre de comparaison A = 156
      TCCR2B=B111;    // Prescaler 1024 (Clock/1024)
      TIMSK2=B010;    // IT Timer2 Quand TCNT2=OCR2A

      //Configuration PORTB.5
      DDRB |= B100000;  // PB5 en sortie
      PORTB &= ~(1<<PORTB5); // PORTB.5 <-0

      sei();     // activation des IT (SREG.7=1)
}

void loop() {      // aucun traitement  }
```

Quelle fréquence de clignotement ?

Clock = signal périodique de 16MegaHz (16 millions de ticks par seconde)
Prescaler = 1024 -> la fréquence d'incrément de TCNT2 = (16/1024) MegaHz
Fréquence de débordement :
TCNT2 ne déborde que tous les 156 incréments (valeur de OCR2A)
c'est-à-dire à la fréquence de 16/(1024*156) MegaHZ ≈ 100 Hz
Il y a 1 interruption Timer2 On Compare A environ tous les 1/100 secondes.

La LED s'allume 1/10 de seconde puis s'éteint 4/10.de seconde. Elle s'allume brièvement 2 fois par seconde.

Remarque (noms des sources d'interruption) : lorsqu'on définit la fonction associée à une source d'interruption

```
ISR(xxx){
      // fonction assiociée à la source xxx d'interruption
}
```

le nom de la source d'interruption est un symbole défini parmi les valeurs suivantes

```
#define INT0_vect          _VECTOR(1)   /* External Interrupt Request 0 */
#define INT1_vect          _VECTOR(2)   /* External Interrupt Request 1 */
#define PCINT0_vect        _VECTOR(3)   /* Pin Change Interrupt Request 0 */
#define PCINT1_vect        _VECTOR(4)   /* Pin Change Interrupt Request 0 */
#define PCINT2_vect        _VECTOR(5)   /* Pin Change Interrupt Request 1 */
#define WDT_vect           _VECTOR(6)   /* Watchdog Time-out Interrupt */
#define TIMER2_COMPA_vect  _VECTOR(7)   /* Timer/Counter2 Compare Match A */
#define TIMER2_COMPB_vect  _VECTOR(8)   /* Timer/Counter2 Compare Match A */
#define TIMER2_OVF_vect    _VECTOR(9)   /* Timer/Counter2 Overflow */
#define TIMER1_CAPT_vect   _VECTOR(10)  /* Timer/Counter1 Capture Event */
#define TIMER1_COMPA_vect  _VECTOR(11)  /* Timer/Counter1 Compare Match A */
#define TIMER1_COMPB_vect  _VECTOR(12)  /* Timer/Counter1 Compare Match B */
#define TIMER1_OVF_vect    _VECTOR(13)  /* Timer/Counter1 Overflow */
#define TIMER0_COMPA_vect  _VECTOR(14)  /* TimerCounter0 Compare Match A */
#define TIMER0_COMPB_vect  _VECTOR(15)  /* TimerCounter0 Compare Match B */
#define TIMER0_OVF_vect    _VECTOR(16)  /* Timer/Couner0 Overflow */
#define SPI_STC_vect       _VECTOR(17)  /* SPI Serial Transfer Complete */
#define USART_RX_vect      _VECTOR(18)  /* USART Rx Complete */
#define USART_UDRE_vect    _VECTOR(19)  /* USART, Data Register Empty */
#define USART_TX_vect      _VECTOR(20)  /* USART Tx Complete */
#define ADC_vect           _VECTOR(21)  /* ADC Conversion Complete */
#define EE_READY_vect      _VECTOR(22)  /* EEPROM Ready */
#define ANALOG_COMP_vect   _VECTOR(23)  /* Analog Comparator */
#define TWI_vect           _VECTOR(24)  /* Two-wire Serial Interface */
```

Exploitation des interruptions externes

```
// ARDUINO UNO - IT Externe 0

// Fonction de IT externe 0 (broche PD2)
ISR(INT0_vect){
      PORTB ^= (1<<PORTB5);
}

void setup(){
      //Configuration PORTB.5
      DDRB |= B100000;       // PB5 en sortie
      PORTB &= ~(1<<PORTB5); // PORTB.5 <-0
      // Configuration PORTD.2/INT0
      DDRD &= B11111011;     // PD2 en entrée
      PORTD |= (1<<PORTD2);  // PORTD.2=1
                             // active pull-up

      EIMSK |= 1;    //IT INT0 prise en compte
      EICRA = B10;  // IT = front négatif sur PD2
      sei();     // activation des IT (SREG.7=1)
}

void loop() {    // aucun traitement }
```

Pour le programme précédent, on branche un bouton poussoir entre PD2 et la masse GND (numéro 2 et 7 sur connecteur de la carte ARDUINO UNO).

Quand on enfonce le bouton poussoir, on force l'entrée à 0. On utilise ici la résistance de pull-up interne pour retirer le niveau à 1 quand le bouton est relâché.

Principe : chaque fois qu'on fait passer le niveau de 1 à 0 sur l'entrée INT0(PD2), la fonction d''interruption associée à INT0 est exécutée. Cette action a pour effet d'inverser l'état de la LED.

Configuration

PORTB.5 en sortie = gestion de le LED

PORTD.2 en entrée avec pull-up interne activée = gestion du bouton poussoir
soit DDRD.2=0 ET PORTD.2=1 (cf 4.2 DIGITAL I/O)

Autorisation de la source d'IT INT0 : EIMSK= (0000 0001)b
Interruption sur front négatif sur INT0 : EICRA= (0000 0010)b soit ISC01=1 ISC00=0

Interruption "PIN CHANGE"

Cette interruption a lieu sur un changement d'état d'une des broches PCINTxx. On connecte un bouton poussoir entre AI5 (entrée analogique 5 PC5/PCINT13) et GND. Ca correspond à la source PCINT13 ("Pin Change 13").
Chaque changement d'état sur la broche PC5 conduit à l'interruption PCINT1. Attention, l'interruption PCINT1 correspond à tout changement d'état sur les broches PCINT8 à PCINT14 pour lesquelles le bit correspondant dans PCMSK1 est à1.

```
// ARDUINO UNO - IT Pin Change PCINT13

// Fonction de traitement PCINT1
ISR(PCINT1_vect)
{
      PORTB ^= (1<<PORTB5);
}

void setup(){
   DDRB |= B100000;          // PB5 en sortie
   PORTB &= ~(1<<PORTB5);    // PORTB.5 <-0
   DDRC &= B11011111;      // PC5 en entrée
   PORTC |= (1<<PORTC5);  // pull up active
   PCICR |= B10;     // groupe PCIE1=1 (PCINT14..8)
   PCMSK1|= B00100000; //PCINT13 autorisée

   sei();      // activation des IT (SREG.7=1)
}

void loop() {    // aucun traitement    }
```

Remarque : sur AVR, une fonction d'interruption ne peut pas elle-même être interrompue.