PORT, PIN BROCHES

PORTX : Permet d'assigner des bits aux PIN du port X sur 8 bits. Ex : `PORTX = 0b01;` //Donne la couleur rouge de la DEL dans une certaine assignation.

PINX : Permet de lire les bits aux PIN du PORTX. On obtient donc la valeur logique de la tentions présente sur la broche.

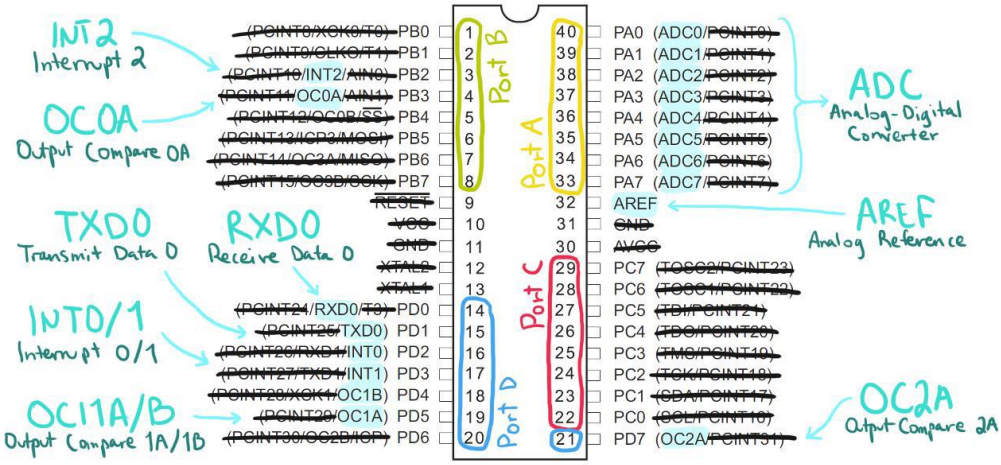
DDRX : Indique si la direction du port est en entré ou en sortie. [7 :0]

> **Entré <0x00>**: les PINs seront réservés aux éléments intégrés au mobo. Ex : DDRD = 0x00; permet l'obtention de l'état du bouton poussoir lorsque qu'il y a un cavalier sur IntEn. Résultat peut s'obtenir par scrutation de la PIN PD2 : while(PIND & 0x04).
>
> **Sortie <0xFF>** : Dépendamment du mode utilisé, les PIN sortes les valeurs dictées par l'instance étant reliée à cette dernière.

BROCHES vs. PIN:

- Les broches sont les éléments physiques sur la carte mère et sont numérotés de [8 : 1]. Les pins sont les éléments tenus dans la documentation Atmel et réfèrent donc aux broches mais sont plutôt numérotés de [7 : 0] et est nommé PXn.
    - Exemple : La PIN selon le documents Atmel pour le bouton poussoir de la carte mère est le PD2 et est situé sur la carte mère à la broche 3 du PORT D.

REMARQUE : Les bits [4:7] du port B (broches 5, 6, 7 et 8) du ATmega324PA sont utilisée par le ATmega8 lorsque ce dernier programme le ATmega324PA lorsque la commande make install est lancée pour que l'exécutable sur le PC se retrouve en mémoire flash.



FONCTIONS INITIALISATION/ISR

On y retrouve la direction des ports et les paramètres de base d'interruptions.

Initialisation/FONCTIONS	
Fonction	**Description**
`cli ()`	Routine qui bloque toutes les interruptions
`sei ()`	Permet de recevoir à nouveau des interruptions.
Initialisation/REGISTRES	
Registres	**Description**
`EIMSK`	Permet les interruptions externes
`EICRA`	Sensibilise les interruptions externes aux changements de niveau du bouton-poussoir
`EIFR`	Si un second signal d'interruption arrive durant l'exécution de ISR, l'AVR s'en souvient (le bit INTF0 est activé dans le EIFR) et la routine ISR sera exécutée une seconde fois, une fois la première terminée
`ISR ()`/**VECTEURS** Cavalier en INTen pour bouton poussoir &dbgEN pour communication USART entre les micro controlleurs & MemEn pour utiliser la mémoire	
Vecteurs (utilisé XXX_vect)	**Description**
INT0	External Interrupt Request 0
PCINT0	Pin Change Interrupt Request 0
TIMER1_CAPT	Timer/Counter1 Capture Event
TIMER1_COMPA	Timer/Counter1 Compare Match A
TIMER1_COMPB	Timer/Counter1 Compare Match B
TIMER1_OVF	Timer/Counter1 Overflow
USART0_RX	USART0 Rx Complete
USART0_UDRE	USART0 Data Register Empty
USART0_TX	USART0 Tx Complete
ANALOG_COMP	Analog Comparator
ADC	ADC Conversion Complete
EE_READY	EEPROM Ready

Exemple TP4/pb2:

```
void initialisation ( void ) {

cli ();

  DDRB = 0xff; // PORT B est en mode sortie
  DDRD = 0x00; // PORT D est en mode ENTREE

EIMSK |= (1 << INT0) ; Interruptions externes enable

EICRA |= (1 << ISC01) | (1 << ISC00); //Any edge genere asynchronously interrupt
 /*
0 0 The low level of INTn generates an interrupt request
0 1 Any edge genere asynchronously interrupt
1 0 The falling edge of INTn generates asynchronously an interrupt request
1 1 The rising edge of INTn generates asynchronously an interrupt request
*/

sei ();

}
```

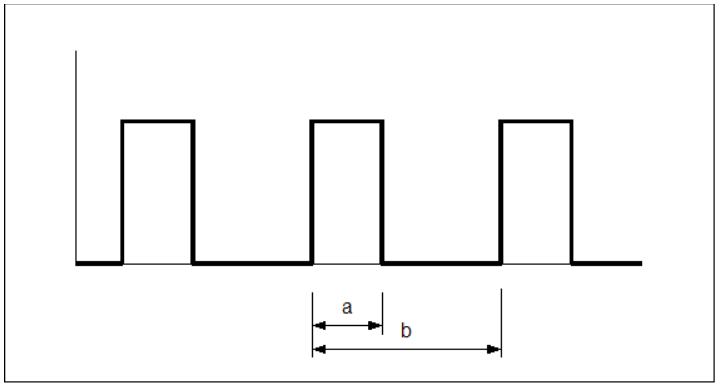
MAKEFILE

Tableau X : Termes relatifs aux Makefiles par sections

Détails de la cible		
Nom	Description	Exemple/Cas
MCU	Nom du microcontrôleur cible	atmega324pa
PROJECTNAME	Nom de votre projet	monprojet
PRJSRC	Fichiers sources	probleme2.cpp
INC	Inclusions additionnels	-I/path/to/mydir
LIBS	Libraires à lier	-lmylib
OPTLEVEL	Niveau d'optimization Utilisez s (size opt), 1, 2, 3 ou 0 (off)	s
AVRDUDE_PROGRAMMERID	Programmer ID - Liste complète des IDs disponible avec avrdude	usbasp
Détails d'implémentation		
CC	Compilateur	avr-gcc
OBJCOPY	Copier le contenu d'un fichier objet vers un autre	avr-objcopy
AVRDUDE	Permet le transfert vers le microcontrôleur	avrdude
REMOVE	Permet de supprimer les fichiers w/make clean	rm -f
HEXFORMAT	Format pour les fichiers produient .hex	ihex
Options de compilation		
CFLAGS	Flags pour le compilateur en C	Voir Makefile
CXXFLAGS	Flags pour le compilateur en C++	-fno-exceptions
LDFLAGS	Linker pour lier les librairies utilisees	-Wl,-Map,$(TRG).map -mmcu=$(MCU)
Utilisations		
make	Compile	make
install	Compile + installe	make install
clean	Supprime les fichers	make clean

LE PWM/PONT EN H

Le PWM : c'est un signal périodique qui n'est que soit à zéro ou soit à un. Il est au niveau haut que durant une certaine fraction de la période totale de l'onde (souvent mesuré en pourcentage). Cette proportion de temps passé au niveau haut est souvent plus importante que la fréquence totale de l'onde. Le front à « true (1) » est représenté par « a », la période « T » est représentée par « b » et le front à « false (0) » est représenté par c.



2 façons de le faire :

- **Logiciel** : Par calcul de chaque tranche et l'attribution une dans une boucle for()
    - Utilise la librairie « util/delay » et peut être fastidieuse à la longue. En effet, pendant la boucle for(), le robot ne peut faire autre chose que le PWM, ce qui peut être un désavantage s'il doit accomplir d'autres tâches.
- **Matériel**: TIMER/COUNTER (Waveform Generator [WGMnX]) & OCRnX
    - Est plus avantageux que la façon logicielle puisqu'il s'agit d'un module indépendant du processeur et qui ne nécessite pas de routine d'interruption. Donc, le main continu même si, les registres OCRnX envoient des signaux d'interruptions, puisque ceux-ci seront plutôt utile à la composante matérielle Waveform Generator.

Avantage :

Lorsque votre robot avancera de façon autonome, il sera fastidieux de générer une onde PWM de la façon dont vous l'avez fait la semaine dernière (semaine 4) puisque le microcontrôleur sera occupé à faire bien d'autres choses. Il vaudra mieux utiliser les ressources internes pour arriver au même but.

Le pourcentage donné est l'équation suivante :

$$\frac{a}{b} * 100 = pourcentage$$

Pont H : N’est pas directement connecté aux bornes d'un moteur à une sortie du ATMega324, car pourrait bruler cette dernière.

2 Problèmes :

- Moteur doit aller dans les deux directions
    - Solution : Transistor fait une commutation de la tension et il peut également amplifier un courant et est plus facile à contrôler par des signaux numériques ou analogiques.
- Vitesse d'un moteur électrique pas proportionnel à la tension. Si cette dernière est trop basse les forces électromagnétiques ne sont pas suffisantes pour combattre le frottement.
    - Solution, donner au moteur des périodes de tension à 100% mais moins longtemps (PWM).

DEL : On ne peut déterminer si le robot avance ou recule uniquement en regardant la couleur des DEL. Toutefois, on peut les réglées en Il faut alors prenant un tournevis et en inversant les fils dans l'un ou l'autre des deux connecteurs 5 mm à 2 positions verts du pont en H.

OPÉRATIONS SUR LES BITS

Opérateur	Définition et retour	Exemple
&&	AND, return 0(false) ou 1	(a&&b), si a ou b ==false, return 0 si a et b ==true, return 1. True != 0
||	OR, return 0(false) ou 1	(a||b), si a ou b ==false, return 0 si a ou b ==true, return 1. True != 0
!	Inverse true à false	!0xFF(true) return 0x00(true) !0x00(false) return 0x01(true)
~	Complément à 1	~0xFE return 0x01
^	XOR, return value	a=1001 b=0111; a^b; return 0110
<<	décalage de n bits vers la gauche	0x02 << 2 return 0x08 0b10 <<2 return 0b1000
>>	décalage de n bits vers la droite	0x04 >> 2 return 0x01 0b100 >>2 return 0b1

COULEURS

```
static const uint8_t
       INCOLORE =    0x00, //Peut aussi être 0b11/0b00 |DEL polarisée
       ROUGE    =    0x01,
       VERT     =    0x02;
```

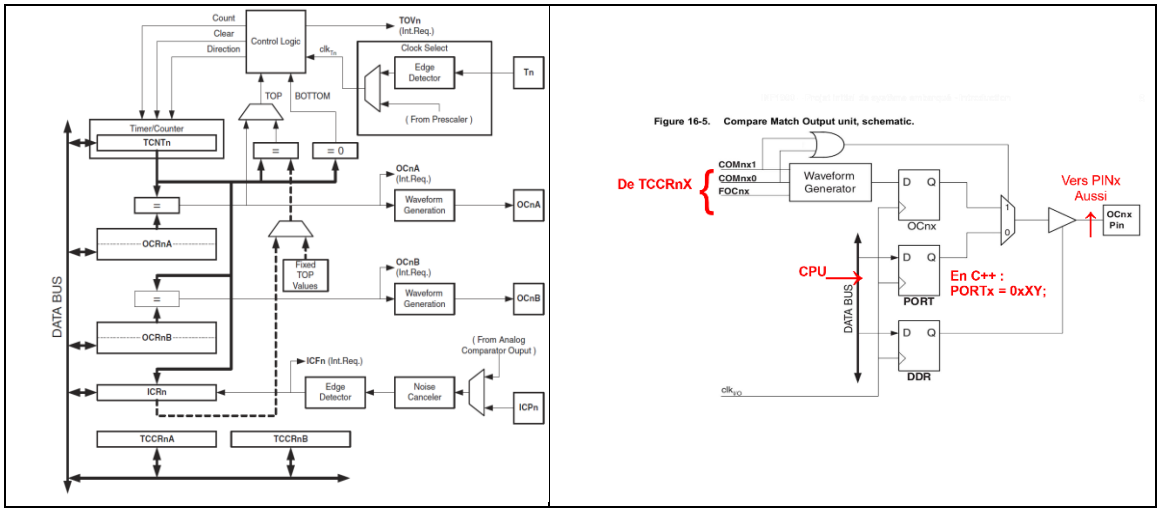
Ambre :

Doit avoir dans l'ordre :

```
#include <avr/io.h>
#define F_CPU 8000000
#include <util/delay.h>
[…]
_delay_ms (double ms) // Ou _delay_us(double) pour microsecondes


Boucle(){
       PORTB = VERT;
       _delay_ms(500.0);
       PORTB = INCOLORE;
       _delay_ms(2000.0);
       PORTB = ROUGE;
```

TIMER1/Interruptions :



Le TIMER1 peut être utilisé avec une fonction ISR, pour obtenir une routine d'interruption avec un timer, ou il peut être utilisé pour générer un PWM.

Lorsqu'il est utilisé pour le temps, on utilise le mode *Clear Timer on Compare* (CTC)

TableauX : Vecteurs relatifs à l'interruption à l'aide d'ISR () :

Vecteurs	Signification
TIMER1_COMPA_vect	Timer/Counter1 Compare Match A
TIMER1_COMPB_vect	Timer/Counter1 Compare MatchB
TIMER1_OVF_vect	Timer/Counter1 Overflow

1) Une fonction ISR n'a pas de valeur de retour.
2) Une fonction ISR peut être appelée directement quand une interruption se présente, peu importe où on en est dans l'exécution du code principal si on a permis les interruptions.
3) On doit faire un réglage pour permettre le bon type d'interruption d'être pris en charge avant que la fonction ISR soit appelée.
4) On ne voit pas, à lire le code principal, un appel direct à une fonction ISR.
5) Une fonction ISR ne devrait pas appeler elle-même de nombreuses autres fonctions pour être complète et bien remplir son rôle de manière efficace.

Registres

TCNT1 :

- Est un compteur allant jusqu'à 16 bits (2^{15}). La vitesse d'incrémentation dépend du clock (clk_{TN}) donné par le prescalor. Il contient deux registres de 8 bits soit **L** ou **H** pour Low (0 à 7) et High(8 à 15).
- **On peut lui assigner une valeur de départ qui est 0 par défaut.**

TCCR1A: Timer/Counter 1 Control Register A.

- Agit sur Compare On Match (**COMnX**[1 :0]) et Waveform Generation Mode**WGMn**[1 :0].
- Le comportement de COMnX dépend de WGMn

TCCR1B: Timer/Counter 1 Control Register B.

- Agit sur **WGMn[3 :2]** et Clock Select **CSn[2:0]**

TCCR1C: Timer/Counter 1 Control Register C.

- Agit sur Force Output CompareA/B (**FOCnA/FOCnB**). Toutefois : Les bits FOCnA/FOCnB ne sont seulement actif quand les bits de WGMn3:0 spécifient un mode non PWM.

OCR1A : Output Compare Register1 A.

- Valeur que doit atteindre TCNT1 pour avoir une interruption/PWM

OCR1B: Output Compare Register1 B.

- Valeur que doit atteindre TCNT1 pour avoir une interruption

OC1A :

- PIN pour PWM (PD5)

OC1B :

- PIN pour PWM (PD6)

TIMSK1: All interrupts are individually masked with the *Timer Interrupt Mask Register* (TIMSKn)

- Met en enable l'interruption associée aux flag dans TIFR1.

TIFR1: Timer/Counter1 Interrupt Flag Register. Agis sur:

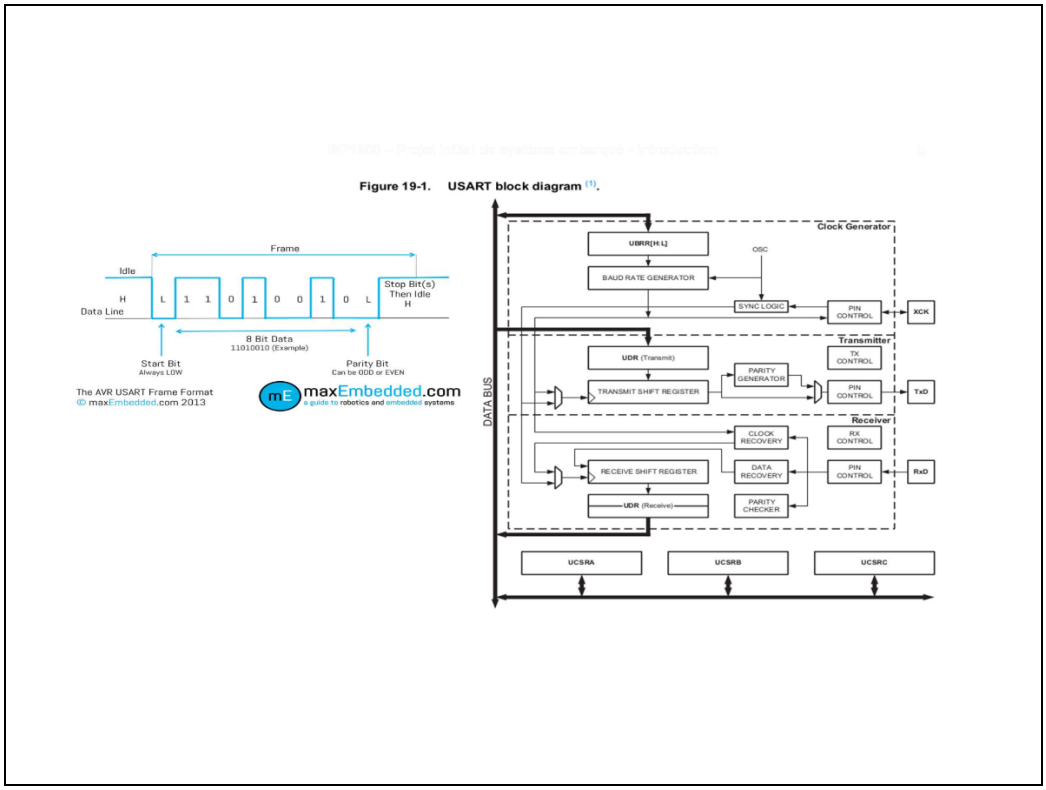
- TOV1: Timer/Counter1, Overflow Flag
- OCF1A: Timer/Counter1, Output Compare A Match Flag
- OCF1B: Timer/Counter1, Output Compare B Match Flag

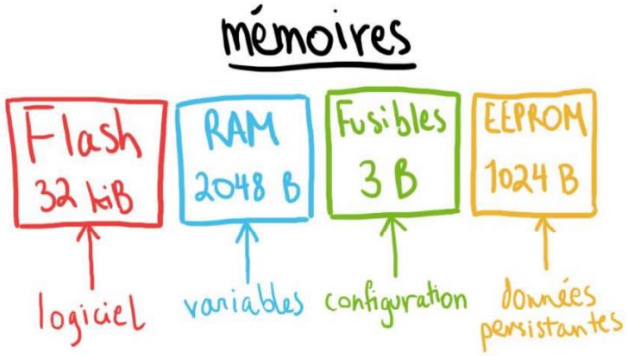
TP4/pb2 : Jeux de réflexes		
Vecteur ISR		Signification
TIMER1_COMPA_vect		Interrupt. Compare Match A
Registres		
Registre	Opération sur les bits	Signification
TCNT1	=0	Timer à 0
TCCR1A	|=(1 << COM1A0)	Active **OCnA/OCnB** en Compare Match.
TCCR1B	|= (1<< CS12) | (0<< CS11) | (1<< CS10)	Clock divisé par 1024
TCCR1C	=0	Set to zero when TCCRnA is written when operating in a PWM mode.
OCR1A	=durée	Valeur de COM fixée
OCR1B		
OC1A	DDRD = 0x00	PORTD est entrée
OC1B		
TIMSK1	1<< OCIE1A	Comparaison enable avec registre OCR1A

Tableau X : Exemple TP4/pb3

TP4/pb3 : PWM PWM 8 bits, phase correcte		
Vecteur ISR		Signification
Aucun		n/a
Registre	Opération sur les bits	Signification
TCNT1	=0	Timer à 0
TCCR1A	|= 1<<COM1A1 | 1<<COM1B1 | 1<<WGM10	PWM 8 bits, phase correcte
TCCR1C	=0	Set to zero when TCCRnA is written when operating in a PWM mode.
TCCR1C	|= 0<<CS12 | 1<<CS11 | 0<<CS10	Clock divisé par 8 (prescaleur)
OCR1A	=durée	Valeur de COM fixée
OCR1B		
OC1A	DDRD = 0x00	PORTD est entrée
OC1A		
TIMSK1	1<< OCIE1A	?

USART





La mémoire externe EEPROM sur notre circuit n'a que 8 broches et doit donc être accédé avec un **protocole série** appelé I²C(TWI). Toutefois, la communication entre les deux puces, soit ATM324 et ATM8 se fait par le protocole RS232 de via leurs U(S)ART.

Protocole série : Demande moins de broches et prend donc moins de place sur une carte. Le protocole d'accès peut cependant être plus complexe étant donné qu'il faut souvent envoyer l'adresse en premier, suivie de la donnée, et ce, bit par bit.

REGISTRES:

UCSR0A: USART Control and Status Register A

- Agit sur les flags Receive complete Transmit Complete

UCSRnB: USART Control and Status Register n B

- Agit sur *Receiver Enable n* **(RXENn)** et *Transmitter Enable n* **(TXENn)**

UCSRnC: USART Control and Status Register n C

- Agit sur le mode (synchrone/asynchrone) *USART Mode Select* **(UMSELn1:0)**
- Agit sur le bit de parité *Parity Mode* **(UPMn1:0)**
- Agit sur le stop bit *Stop Bit Select* **(USBSn)**
- Agit sur la taille des charactères reçu *Character Size* **(UCSZn1:0)**

Configuration de base pour utiliser le USART0 : 2400 baud, 8 bits, none parity, 1 start/stop

Registres	Opération sur les bits	Description
UBRR0H	=0	2400 bauds
UBRR0L	=0xCF	
UCSR0A	=0	Met les flags à 0
UCSR0B	|= _BV(RXEN0) | _BV(TXEN0)	Active transmission et reception
UCSR0C	rien	Est déjà à 8N1 asynchrone
Broches		
Nom	Description	
T_x	Broche qui permet la transmission de données (PD1)	
R_x	Broche qui permet la reception de données (PD0)	

Rôles des micro-processeurs :

- Le ATMega8 effectue la conversion de protocoles entre le RS232 et l'USB.

La tension des signaux à -12 et +12 volts par le passage des signaux.

Transférer des données à :

- PC : cavalier en DbgEn

```
void USART_Transmit( unsigned char data )
{
/* Wait for empty transmit buffer */
while ( !( UCSRnA & (1<<UDREn)) )
;
/* Put data into buffer, sends the data */
UDRn = data;
}
```

- EEPROM : cavalier en MemEn

```
// deux variantes pour la l'ecriture egalement:
```

```
// une donnee a la fois
   uint8_t ecriture(const uint16_t adresse, const uint8_t
donnee);
   // bloc de donnees : longueur doit etre de 127 et moins
   uint8_t ecriture(const uint16_t adresse, uint8_t *donnee,
                    const uint8_t longueur);
```

MATÉRIEL DE MESURE ET TENSION

MULTIMÈTRE :

Utilisation : Puisque l'on utilise le courant continu (CC), on utilise le bouton DC (mV). Pour utiliser le courant continu, on appuis sur le bouton à droite de Ω.

OCILLOSCOPE: Un oscilloscope est essentiellement un voltmètre avec un axe de temps.

Côté de droite pour analogique, gauche pour numérique.

- 4 entrées analogiques
- La petite ligne rouge sur le bord du câble plat donne la position du premier signal.
- On peut opérer en mode analogique et numérique en même temps.
- On peut analyser 16 signaux numériques d'un seul coup.
- On peut ajuster facilement l'amplitude d'un signal analogique à l'écran.
- On peut figer un signal dans temps en appuyant sur le bouton «Stop».
- Appuyer sur le bouton «auto-scale» nous donne d'excellentes chances d'obtenir un signal PWM directement à l'écran si on a fait les bons branchements à l'appareil.
- On ne peut rien supposer a priori sur la forme générale de l'onde ou sur les transitions du mode analogique.

Numérique :

- On peut faire auto-scale, mais si le l'affichage ne nous convient pas on peut quelques modifications :
    - « **trigger** » (complètement à droite) : On choisira la source de déclenchement et le type de front, montant ou descendant, avec les boutons au bas de l'écran.
    - Les modes correspondant aux boutons «*pulse width*», «*pattern*» et «*more*» peuvent devenir complexes.

Analogique :

- Le réglage de l'amplitude d'un signal analogique se fait en utilisant le gros vernier de couleur (jaune, vert, bleu ou rose) correspondant à chaque canal.
- Le réglage de la position verticale se fait par le petit vernier mais l'amplitude reste la même

SOURCE DE TENSION :

- 4 sources (channels) : **combinées** (série ou en parallèle) ou utilisées de manière **indépendante.**
- Il est à noter que l'appareil se met en marche en appuyant sur le bouton «*POWER*». Par contre, aucune des sources ne sera active à moins que le bouton «*OUTPUT*» au-dessus ne soit enfoncé et que la DEL d'indication ne tourne au vert.

Bouton « VOLTAGE » : Ajuste la tension pour le canal. ATTENTION, CE N'EST PAS TOUTES LES SOURCES QUI SONT ÉQUIVALENTES.

Ordre des channels : 4-2-1-3

Channel 1 et 2 vont de 0 à 30v-3A

Channels 3 et 4 vont de 2,2 à 5,2v -1A. Ils sont donc moins flexibles.

Si la DEL entre « CURRENT » et « VOLTAGE » est rouge, ça veut dire que l'ampérage n'est pas assez grand pour obtenir le voltage désiré sur la carte.

CAVALIERS ET CONFIGURATION

Voici un petit tableau qui résume les configurations possibles des cavaliers:

Nom du cavalier	Position	Signification
VtgEN	En place	Le programmeur ISP fournit la tension à la carte mère (carte cible d'où le nom *target*)
	Retiré	La carte mère doit avoir sa source d'alimentation propre durant sa programmation par ISP
PrgSEL	Sur positions 1 et 2	Le ATmega324PA sera programmé par ISP
	Sur positions 2 et 3	Le ATmega8 sera programmé par ISP
	Retiré	La carte mère est un programmeur ISP et peut programmer une autre carte
IntEN	En place	Le bouton-poussoir *Interrupt* soulève l'interruption INT0 du ATmega324PA et utilise le port D2.
	Retiré	Le bouton-poussoir *Interrupt* est inutilisé et le port D2 est disponible pour usage général.
MémEN	En place	La mémoire série de la carte mère est accessible. Les ports C0 et C1 sont utilisés pour accéder à la mémoire.
	Retiré	La mémoire série de la carte mère ne peut être utilisée. Les ports C0 et C1 sont libres pour usage général.
DbgEN	En place	La communication série du ATmega324PA peut être redirigée vers le ATmega8 puis renvoyée vers le PC par USB. Ports D0 et D1 utilisés pour la transmission.
	Retiré	La communication série du ATmega324PA n'est reliée à aucun autre composant. D0 et D1 libres pour utilisation générale.