Journée 3
Systèmes embarqués, développement sur la plateforme Arduino

Dr. Ing. Chiheb Ameur ABID

Contact : chiheb.abid@gmail.com

Janvier 2020

Plan

L'intérêt des interruptions

Comment vérifier une condition/évènement ?

Par exemple, on se propose de vérifier qu'une broche d'entrée prends la valeur 1 !?

1. Attente active

```
for(;;) {
 while (digitalRead(broche)==0);
 Action ;
 Action 1;
 Action 2;
 Action 3;}
```

✖ Le processeur est bloqué sur la vérification de l'évènement

L'intérêt des interruptions

Comment vérifier une condition/évènement ?

Par exemple, on se propose de vérifier qu'une broche d'entrée prends la valeur 1 !?

2. Lecture par scrutation (Polling)

```
for(;;) {
  Action 1;
  Action 2;
  if (digitalRead(broche)) Action();
  Action 3; }
```

✔ Simple à mettre en oeuvre
✖ Il est possible de louper certains évènements
✖ Consomme du temps processeur

L'intérêt des interruptions

Comment vérifier une condition/évènement ?

Par exemple, on se propose de vérifier qu'une broche d'entrée prends la valeur

❸ Les interruptions

```
for(;;) {
 Action 1;
 Action 2;
 Action 3;}
 ISR() {
 Action; }
```

- ✔ La fonction `ISR` est appelée à chaque demande d'interruption (IRQ)
- ✔ La charge de vérification du changement de la valeur est déléguée au contrôleur d'interruptions

Les interruptions externes/internes

Les interruptions internes

➥ Elles sont souvent appelées **exceptions**. Elles surviennent lors dune erreur dexécution du programme (overflow, watchdog, Timer, adressage, etc.)

Les interruptions externes

➥ Elles concernent essentiellement les demandes de service des périphériques. Elles arrivent de manière asynchrone avec les instructions en cours dexécution.

Vecteurs d'interruptions

Vecteurs d'interruptions

➥ Plusieurs sources d'interruptions ?: quelle ISR exécuter ? Où peut-on trouver l'adresse de l'ISR adéquate ?

➥ Deux interruptions se produisent en même temps : laquelle exécutée ?

Z Les vecteurs d'interruptions

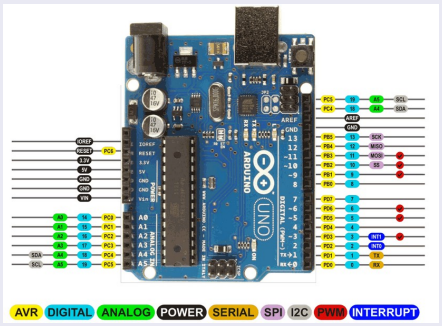
➥ Pour une source d'interruption, un vecteur d'interruption précise l'adresse de son ISR

➥ Les vecteurs d'interruptions sont rangés dans une table

➥ L'ordre d'apparence dans des vecteurs d'interruptions dans la table précise les priorités

Les interruptions avec Arduino

Les entrées sorties de l'Arduino Uno



Les interruptions avec Arduino

Table des vecteurs d'interruptions de l'Arduino Uno (1/2)

Numéro	Adresse	Source	Définition de l'interruption
1	0x0000	RESET	External Pin, Poweron Reset, Brownout 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 Timeout 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/Counter1 Compare Match B

Les interruptions avec Arduino

Table des vecteurs d'interruptions de l'Arduino Uno (2/2)

Numéro	Adresse	Source	Définition de l'interruption
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
23	0x002C	EE READY	EEPROM Ready
24	0x002E	ANALOG COMP	Analog Comparator
25	0x0030	TWI	2-wire Serial Interface
26	0x0032	SPM READY	Store Program Memory Ready

Les interruptions avec Arduino

Les interruptions externes

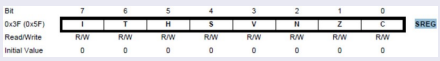
- Elle permettent de déclencher une séquence de code lors dun signal électrique sur lune des broches dentrée du micro-contrôleur
- Arduino Uno peut gérer jusqu'à 5 interruptions externes
  - `INT0` sur la broche `PD2` et `INT1` sur la broche `PD3`
  - Les interruptions `PCINT0`, `PCINT1` et `PCINT2` s'appliquent respectivement aux ports `B`, `C` et `D`

✍ Les broches `PD2` et `PD3` peuvent être gérées individuellement par `INT0` et `INT1`, ou par `PCIE2` au niveau du port `D`

Les interruptions avec Arduino

Registre de contrôle

- Le registre `SREG` (Status Register) permet d'activer/désactiver la gestion des interruptions



  - `I` → 1 : Activer la gestion des interruptions. L'activation
    L'activation peut s'effectuer en utilisant la fonction `sei()`
  - `I` → 0 : désactiver toutes les interruptions
    La désactivation peut s'effectuer en utilisant la fonction `cli()`

✍ Dés qu'une interruption se déclenche, elle met automatiquement le bit `I` à 0 bloquant ainsi la possibilité d'avoir plusieurs interruptions simultanées. Une fois l'interruption est gérée, le bit `I` est systématiquement repositionné à 1.

Gestion des interruptions externes

Vecteur d'interruptions

- Les vecteurs d'interruptions associées aux interruptions externes

Numéro	Adresse	Source	Définition de l'interruption
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

Gestion des interruptions externes

Gestion des interruptions `INT0` et `INT1`

- Le registre `EICRA` (External Interrupt Control Register A) permet de spécifier le mode de déclenchement d'une interruption

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	

- Les bits `ISC0x` sont associés à `INT0`, et les bits `ISC1x` sont associés `INT1`

ISC11 (*ISC01*)	ISC10 (*ISC00*)	Description
0	0	Déclenchement sur un niveau bas
0	1	Déclenchement sur changement de niveau
1	0	Déclenchement sur front descendant
1	1	Déclenchement sur front montant

Gestion des interruptions externes

Gestion des interruptions `INT0` et `INT1`

- Le registre `EIMSK` (External Interrupt Mask Register) permet d'activer ou désactiver une interruption

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	

  - `INTx` → 1 : l'interruption est activée
    `INTx` → 0 : l'interruption est désactivée
- Le registre `EIFR` (External Interrupt Flag Register) indique si une interruption s'est produite

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	

  - `INTFx` → 1 : une interruption de type `INTx` s'est produite
    `INTFx` → 0 : pas d'interruption de type `INTx`

Gestion des interruptions externes

Activer/Désactiver la gestion des interruptions

- Activer la gestion des interruptions
  `sei()`
- Désactiver la gestion des interruptions
  `cli()`

Spécifier l'ISR

- L'ISR pour une interruption est spécifiée à travers la macro `ISR`
  `ISR(NOMINTERRUPTION_vect)`

Gestion des interruptions externes

Bibliothèque Wiring

- On utilise la fonction `attachInterrupt` qui permet de spécifier l'ISR pour une interruption externe
  `void attachInterrupt(digitalPinToInterrupt(pin), ISR, mode)`
  - `pin` spécifie la broche sur laquelle l'évènement est produit
  - `ISR` la fonction ISR à exécuter
  - `mode` sélectionne l'évènement induisant une demande d'interruption `LOW`,`HIGH`, `RISING` ou `FALLING`
- La fonction `detachInterrupt(digitalPinToInterrupt(pin))` permet de désactiver la gestion des interruptions pour la broche spécifiée

Avertissement

Les variables modifiées dans des routines ISR doivent être déclarées avec la directive `volatile`

Gestion des interruptions externes

Mise en application : objectif

- On se propose d'utiliser l'interruption `INT0` pour inverser l'état d'une diode LED à chaque appuie sur un bouton poussoir. En parallèle, une deuxième diode LED change d'état à chaque seconde.
- On se propose de réaliser deux versions
  1. En utilisant les registres
  2. En utilisant les fonctions de la bibliothèque `Wiring`

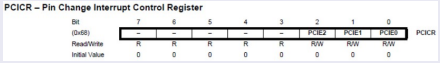
Gestion des interruptions externes

Les interruptions `PCIE0`, `PCIE1` et `PCIE2`

- Les interruptions `PCIE0`, `PCIE1` et `PCIE2` sont produites sur un changement d'état respectivement sur les broches des ports `B`, `C` et `D`
- Il est possible de configurer les broches dont le changement d'état produit ou non une interruption

Gestion des interruptions `PCIE0`, `PCIE1` et `PCIE2`

- Le registre `PCICR` (Pin Change Interrupt Control Register) permet d'activer ou désactiver une interruption

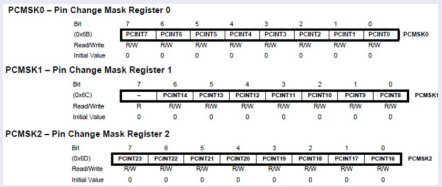


  - `PCIEx` → 1 : l'interruption est activée
    `PCIEx` → 0 : l'interruption est désactivée

Gestion des interruptions externes

Gestion des interruptions `PCIE0`, `PCIE1` et `PCIE2`

- Le registre `PCMSKx` (Pin Change MaSK) permet d'activer ou désactiver une interruption

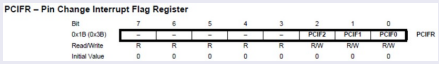


  - `PCINTy` ← 1 : le changement d'état de la broche `y` produit une interruption
    `PCINTy` ← 0 : le changement d'état de la broche `y` ne produit pas une interruption

Gestion des interruptions externes

Gestion des interruptions PCIE0, PCIE1 et PCIE2

- Le registre PCIFR (Pin Change Interrupt Flag Register) indique si une interruption de type PCIEx s'est produite



- Z PCIFx ← 1 : une interruption PCIEx s'est produite

Exemple

On souhaite qu'un changement d'état sur la broche PB2 conduise à une interruption

- L'interruption à produire est de type PCIE0
  1. SREG(7) ← 1 : activer la gestion des IT
  2. PCICR(0) ← 1 : activer la production de l'interruption PCIE0
  3. PCMSK0(2)=PCINT2 ← 1 : le changement d'état sur la broche PB2 est pris en compte

Plan

Les timers

Qu'est-ce qu'un Timer ?

- Un timer est un circuit matériel qui modélise un compteur dont le modulo et la période sont reconfigurables
- Un timer est généralement utilisé pour les fonctions suivantes
  - Z Fonction Temporisateur
  - Z Fonction Compteur d'évènements
  - Z Génération de signaux périodiques (tel que PWM)

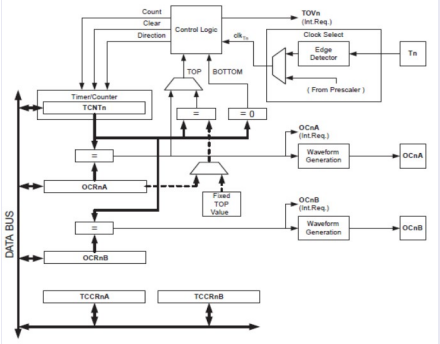
Les timers

Les timers de la carte Arduino Uno

- La carte Arduino Uno possède trois timers
  1. Timer 0 : timer 8 bits
     Utilisé par les fonctions delay(), millis() et micros()
  2. Timer 1 : Timer 16 bits
     Utilisé par la librairie Servo
  3. Timer 2 : timer 8 bits
     Utilisé par la bibliothèque tone
- Les trois Timers fonctionnent en utilisant le signal horloge de la carte, à savoir avec une fréquence de **16 Mhz**

Structure d'un Timer

Structure d'un Timer

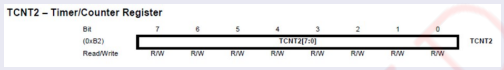


- *Tn* représente le signal horloge à utiliser par le Timer/Counter
- La sortie `TOVn` (Timer Over Flow) est mis à 1 chaque fois que le comparateur atteint sa valeur maximale
- Chaque Timer possède deux canaux `OCnA` et `OCnB` sur lesquels on peut produire des signaux PWM

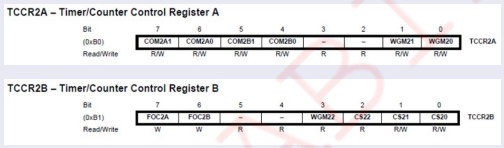
Structures des registres des Timers (0 et 2)

Les principaux registres des timers

- TCNTx - Timer/Counter Register : valeur courante du compteur



- TCCRxA/B - Timer/Counter Configuration Register : registres de reconfiguration

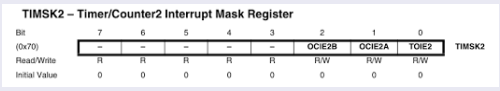


CSx[2:0] spécifie le diviseur de fréquence à utiliser (001 ➡ 1, 010 ➡ 8, 011 ➡ 32,100 ➡ 64,101 ➡ 128,110 ➡ 256,111 ➡ 1024)
WGM[2:0] spécifie le mode de fonctionnement de compteur
COMxA[2:0] (resp. COMxB[2::0]) spécifie le comportement des sorties OCxA (resp. OCxB)

Structures des registres des Timers (0 et 2)

Les principaux registres des timers

- OCRxA/B - Output Compare Register : valeur de comparaison
- TIMSKx - Timer Interrupt MaSK : activation/désactivation des ITs



- `OCIExB` ← 1 : générer une interruption lorsque TCNTx=OCRxB
- `OCIExA` ← 1 : générer une interruption lorsque TCNTx=OCRxA
- `TOIEx` ← 1 : générer une interruption lorsque TCNTx atteint sa valeur maximale

Fonctionnement des Timers

Fonction temporisateur : mesurer une durée

1. La fréquence de fonctionnement est fixer à travers la fréquence d'horloge utilisé et un pré-diviseur de fréquence (prescaler)
2. Initialiser les registres du Timer en fixant une valeur à atteindre par le compteur
3. Attendre lactivation de TOV (Timer Overflow interrupt flag) : la valeur du compteur est égale à la valeur fournie
4. On prend une décision : e.g. on ne repart pas de 0 mais dune autre valeur => temps plus court pour arriver à 255.
5. Remettre à zéro TOV
6. Réinitialiser TCNT pour la durée voulue

Mise en application : utilisation d'un Timer

Objectif

On se propose de clignoter la LED intégrée à la carte Arduino (connectée via la broche 13) à chaque seconde en utilisant le Timer 2.

Déterminer les paramètres du Timer

- Quel est le diviseur de fréquence le plus adéquat ?

Journée 3 Chiheb Ameur ABID 29/ 56
└Les Timers

Mise en application : utilisation d'un Timer

Objectif

On se propose de clignoter la LED intégrée à la carte Arduino (connectée via la broche 13) à chaque seconde en utilisant le Timer 2.

Déterminer les paramètres du Timer

- Quel est le diviseur de fréquence le plus adéquat ?
  Le plus grand diviseur autrement dit 1024
- Comment spécifier ce diviseur ?

Mise en application : utilisation d'un Timer

Sketch

```
volatile unsigned char cpt=0; // compteur d'IT
// Fonction de traitement IT 10 = Timer 2 OverFlow
ISR(TIMER2_OVF_vect){
   cpt++;
   if(cpt==61) {
      PORTB ^=(1<<PORTB5);
      cpt=0;}
}
void setup() {
   // Mode Normal ; Prescaler 1024 (Clock/1024)
   TCCR2A=0; TCCR2B=B111;
   // IT Timer2 Over Flow Active
   TIMSK2=0x01;
   DDRB |= 0b100000;// PB5 en sortie
   PORTB &= ~(1<<PORTB5); // PORTB.5 <-0
   // Activation des IT (SREG.7=1)
   sei();}

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

Journée 3 Chiheb Ameur ABID 31/ 56
└Les Timers

Mise en application : utilisation d'un Timer (1/2)

Sketch d'implémentation de la fonction delay_ms()

```
volatile unsigned int val_g,cpt=0;
volatile bool mutex=false;

ISR(TIMER2_OVF_vect){
   cpt++;
   if(cpt==(8*val_g)) {
      cpt=0;
      mutex=true;
   }
}
void delay_ms(int duree) {
   TIMSK2|=0b1;
   val_g=duree;
   while (mutex==false);
   TIMSK2^=0b1;
   mutex=false;
}
```

Mise en application : utilisation d'un Timer (2/2)

Sketch d'implémentation de la fonction delay_ms()

```
void setup() {
   Serial.begin(9600);
   TCCR2A=0; TCCR2B=0b010;
   sei();
}

void loop() {
   Serial.println("Message");
   delay_ms(100);
}
```


Plan

Timer et PWM

Timer et PWM

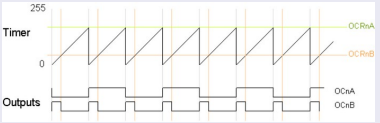
- Les Timers sont utilisés pour générer les signaux PWM en utilisant la fonction `analogWrite()`
  - Timer 0 → broches 5 (PD5 - OC0B) et 6 (PD6 - OC0A)
  - Timer 1 → broches 9 (PB1 - OC1A) et 10 (PB2 - OC1B)
  - Timer 2 → broches 3 (PD3 - OC2B) et 11 (PB3 - OC2A)

  Ces broches PWM sont connectées aux sorties des Timers OCxA et OCxB
- L'utilisation directe des Timers pour générer un signal PWM
  - Meilleur contrôle de la résolution du rapport cyclique en utilisant le Timer 1
  - Changer la fréquence de génération du signal PWM

Timer et PWM

Modes de fonctionnement des timers pour les signaux PWM

1. Fast PWM

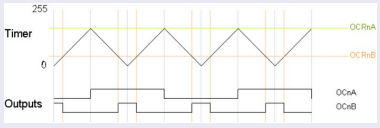


  - WGM[2:0] ← 011, COMxA ← 01 (Toggle on compare match) et COMxB ← 10 (Clear on compare match, Set on bottom)

Timer et PWM

Modes de fonctionnement des timers pour les signaux PWM

❷ Phase correct PWM



Z WGM[2:0] ← 001, COMxA ← 01 (Toggle on compare match) et COMxB ← 10 (Clear on compare match, Set on bottom)

Structures des registres du Timer 1

Les principaux registres du Timer 1

➥ TCNT1 : valeur courante du compteur sur 16 bits
TCNT1H et TCNT1L constituent le registre symbolique TCNT1

➥ Les registres OCR1A et OCR1B sont sur 16 bits

➥ ICR1 (Input Compare Register) : peut être utilisé pour définir la valeur maximale à atteindre par le compteur TCNT1

➥ TIMSK1 - Timer Interrupt MaSK : activation/désactivation des ITs

TIMSK1 – Timer/Counter1 Interrupt Mask Register

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

Z `OCIE1B` ← 1 : générer une interruption lorsque TCNT1=OCR1B
Z `OCIE1A` ← 1 : générer une interruption lorsque TCNT1=OCR1A
Z `TOIE1` ← 1 : générer une interruption lorsque TCNT1 atteint sa valeur maximale
Z `ICIE1` ← 1 : générer une interruption lorsque TCNT1=ICR1

Structures des registres du Timer 1

Les principaux registres du Timer 1

➥ La structure des registres de contrôle TCCR1A et TCCR1B

TCCR1A – Timer/Counter1 Control Register A

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

➥ Sélection de diviseur de fréquence

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.

Structures des registres du Timer 1

Les principaux registres du Timer 1

➥ Modes de génération des signaux de sorties

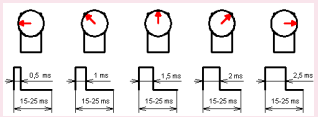
Mode	WGM13	WGM12 (CTC1)	WGM11 (PWM11)	WGM10 (PWM10)	Timer/Counter Mode of Operation	TOP	Update of OCR1x at	TOV1 Flag Set on
0	0	0	0	0	Normal	0xFFFF	Immediate	MAX
1	0	0	0	1	PWM, Phase Correct, 8-bit	0x00FF	TOP	BOTTOM
2	0	0	1	0	PWM, Phase Correct, 9-bit	0x01FF	TOP	BOTTOM
3	0	0	1	1	PWM, Phase Correct, 10-bit	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

Mise en application

Mise en application : commande d'un servomoteur

- L'objectif est de commander le servomoteur `SG90 9 g Micro Servo`

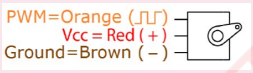
- Un servomoteur est un système qui a pour but de produire un mouvement précis en réponse à une commande externe.
- Un servomoteur est asservi en position angulaire à travers un signal PWM



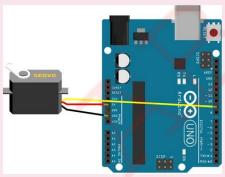
Mise en application

Mise en application : commande d'un servomoteur

- Les connexions d'un servomoteur



- Montage



Mise en application

Mise en application : commande d'un servomoteur

- On utilisera la broche 10 (connectée à OC1B) pour la génération du signal PWM
- Mode de génération des signaux
  WGM[3:0] ← 1110 (Fast PWM avec ICR1 valeur maximale du compteur)
- COM1B ← 10 (Clear on compare match, set on bottom)
- Choix de prescaler `CS1[2:00]` ← 010 ⇒ $clk/8$ ⇒ fréquence=2 Mhz
- `ICR1` ← 40000 ⇒ fréquence de génération sur `OC1B` = 50 Hz
- La valeur spécifiée dans `OCR1B` définit la largeur de la pulsation (entre 1000 et 5000)

TCCR1A – Timer/Counter1 Control Register A

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

TCCR1B – Timer/Counter1 Control Register B

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

Mise en application

Mise en application : commande d'un servomoteur

```
1 void setup() {
2    pinMode(10,OUTPUT);
3    TCCR1A = 0b00100010; // Set OC1B; WMG1[1]
4    TCCR1B = 0b00011000; // Set WMG1[2] & WMG1[3]
5    TCNT1 = 0; // clear counter
6    ICR1 = 40000; // count of 40000 for 50 Hz cycle
7    OCR1B = 1000; // initial position
8    TCCR1B|=0b10; // activate clock
9 }
```

Mise en application

Mise en application : commande d'un servomoteur

```
void loop() {
   for (uint16_t ii=1050;ii<4800;ii+=10)
   {
      OCR1B = ii;
      delay(50);
   }
   delay(2000);
}
```


Timer et PWM

La bibliothèque TimerOne

- Cette bibliothèque offre des fonctions pour la génération d'un signal PWM avec Timer1
- Installation de la bibliothèque TimerOne
  1. Aller au Menu `Croquis` ➡ `Inclure une bibliothèque` ➡ `Gérer les bibliothèques`, ou tout simplement par les raccourcis clavier `Ctrl+MAJ+I`
  2. Dans le champ de recherche `Filtrez votre recherche`, taper `timerone`
  3. Cliquer sur le bouton `Installer`

Timer et PWM

Les fonctions Arduino de Timer1

- `void TimerOne::initialize(unsigned long microseconds=1000000)` : spécifie la durée en microsecondes de la période du signal à générer
- `void TimerOne::pwm(char pin, unsigned int duty)` : spécifie le rapport cyclique du signal PWM à générer sur la broche `pin`
  `pin`$\in${`9,10`} et `duty`$\in${`0,...,1023`}

Mise en application

Mise en application : commande d'un servomoteur

```
#include "TimerOne.h"
TimerOne t1;
void setup() {
   pinMode(10,OUTPUT);
   t1.initialize(40000);
}

void loop() {
   for (int jj=0;jj<180;jj++) {
      int ii = map(jj,0,180,13,64);
      t1.pwm(10,ii);
      delay(50);
   }
}
```

Plan

Les machines à états

Les machines à états

- Un automate fini ou machine à états finis (finite state machine) est un modèle mathématique pour la modélisation de comportement de systèmes numériques
  - Son rôle, est de décrire le fonctionnement dune machine (ou dun objet) ayant un comportement **séquentiel**
  - Modèle graphique
  - Utilisé dans de nombreux domaines : conception de programmes informatiques, protocoles de communication, contrôle des processus, analyse linguistique, etc.

Les machines à états

Les machines à états

- Dans un système séquentiel, le comportement dune machine (ou dun objet) dépend non seulement des évènements quil reçoit mais aussi de ce qui sest passé avant ces évènements.
  - Par exemple, on ne peut pas faire descendre un ascenseur s'il est déjà en bas
  - Les machines ayant un fonctionnement séquentiel passent par un nombre de situations (états) limitées et clairement identifiées. Nous disons alors que ce sont des machines à états **finis**.

Les machines à états

Les machines à états

- Une machine à états est construite de quatre éléments de base
  - Des états
  - Des évènements
  - Des transitions
  - Des actions

États

- Un état est une situation stable qui possède une certaine durée pendant laquelle un objet exécute une activité ou attend un évènement
- Un état représenté par un cercle dans lequel le nom de l'état est inscrit

Les machines à états

Transitions

- Une transition définit la réponse d'un objet à l'arrivée d'un évènement. Elle indique qu'un objet qui se trouve dans un état peut ń transiter ż vers un autre état en exécutant éventuellement certaines activités
- Une transition est représentée par une flèche allant de l'état de départ vers l'état d'arrivée

Évènements

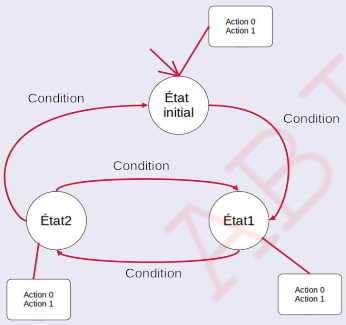
- Un évènement est un fait qui déclenche le changement d'état, qui fait donc passer un objet dun état à un autre état
- Un évènement se produit à un instant précis et est dépourvu de durée. Quand un événement est reçu, une transition peut être déclenchée et faire basculer l'objet dans un nouvel état.
- Un évènement est représenté par une expression inscrite sur une transition

Actions

- Une action consiste à envoyer un signal, à faire appel à une méthode, à affecter une valeur à un attribut, etc.

Les machines à états

Représentation graphique d'une machine à états



Les machines à états

Traduction d'une machine à états en C++

- Définir un type énumératif

```
enum State {etatInitial, etat1, etat2, ...};
```

- Définir une variable globale pour mémoriser l'état courant

```
State etat;
```

- Spécifier dans la fonction `setup()` la valeur initiale de la variable `etat` : l'état initial

```
etat=etatInitial;
```

- Dans la fonction `loop()`, on spéicifie à l'aide d'un `switch` le comportement de la machine à états

```
switch(etat) {
case etatInitial: // Actions de etatInitial
  if (cond) { // Condition de transition
    // Actions juste avant la transition
    etat=etatSuivant;
  }
  break;
case etat0: ...
}
```


Les machines à états

Mise en application

- On se propose de réaliser un système d'alarme. Ce système se déclenche suite à l'appuie sur un bouton poussoir en activant un signal sonore à partir d'un Buzzer actif. L'alarme se désactivera automatiquement après 10 secondes ou suite à l'appuie d'un deuxième bouton poussoir.

- Donner le schéma du montage
- La machine à états permettant de modéliser le comportement de ce système
- Réaliser ce système d'alarme

MERCI POUR VOTRE ATTENTION

Questions ?