Arduino gestion du temps bases

Plan

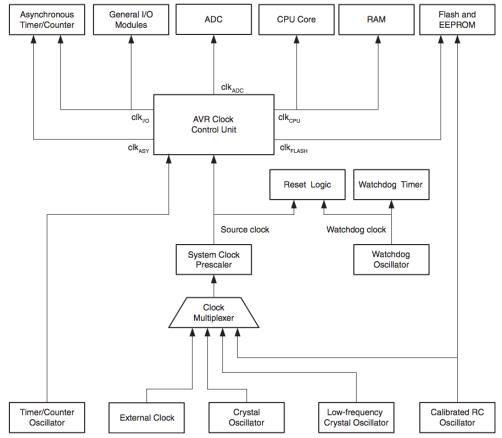
- Besoins en gestion de temps
- Matériel d'horlogeries
- Gestion du temps dans les programmes
- API Arduino basique
- Timers du micro-contrôleur ATmega
- Interruptions externes

Besoins en gestion de temps

- Faire une attente entre deux actions.
- Faire une action périodiquement.
- Faire une séquence d'actions à des dates précises.
- Mesurer le temps qui sépare deux actions.
- Veillez qu'une action ait été effectuée avant une échéance.
- Démarrer une action dans un délai maximum après un événement.

Matériel d'horlogerie ATmega

Figure 10-1. Clock Distribution.



http://www.atmel.com/Images/doc2549.pdf page 40

Horloges générées

- **clk cpu**
  horloge utilisée le processeur interne.
- **clk io**
  horloge utilisée par les périphériques (usart, spi, timer)
- **clk flash**
  horloge utilisée pour la reprogrammation des mémoires flash
- **clk asy**
  horloge asynchrone utilisée pour un timer temps réel
- **clk adc**
  horloge utilisée par le convertisseur analogique digital, pour réduire les bruits des horloge cpu et io

Sources d'horloges

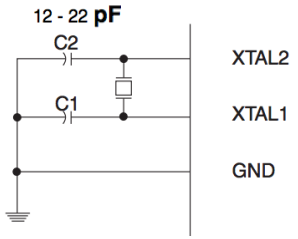
Table 10-1. Device Clocking Options Select(1)

Device Clocking Option	CKSEL3:0
Low Power Crystal Oscillator	1111 - 1000
Full Swing Crystal Oscillator	0111 - 0110
Low Frequency Crystal Oscillator	0101 - 0100
Internal 128kHz RC Oscillator	0011
Calibrated Internal RC Oscillator	0010
External Clock	0000
Reserved	0001

- La sélection des sources d'horloge se fait par des "fusibles" flash. On ne va pas regarder le détail maintenant, il faut comprendre que le choix est imposé par la source matérielle.
- Il faut aussi choisir le délai de démarrage (power-up ou reset).

http://www.atmel.com/Images/doc2549.pdf page 41

Oscillateurs quartz



Horloge et mode sommeil

- Un microcontrôleur doit pouvoir s'endormir pour économiser l'énergie lorsqu'il n'y a pas de traitement à effectuer.
- Mais, certaines horloges restent actives pour permettre le réveil et continuer à compter le temps ou des événements.
- Le micro-contrôleur propose plusieurs niveau d'endormissement plus ou moins profond qui seront détaillés dans un cours prochain.

Table 11-1. Active Clock Domains and Wake-up Sources in the Different Sleep Modes.

	Active Clock Domains					Oscillators		Wake-up Sources						
Sleep Mode	clk_{CPU}	clk_{FLASH}	clk_{IO}	clk_{ADC}	clk_{ASY}	Main Clock Source Enabled	Timer Osc Enabled	INT7:0 and Pin Change	TWI Address Match	Timer2	SPM/ EEPROM Ready	ADC	WDT Interrupt	Other I/O
Idle			X	X	X	X	X(2)	X	X	X	X	X	X	X
ADCNRM				X	X	X	X(2)	X(3)	X	X(2)	X	X	X	
Power-down								X(3)	X				X	
Power-save					X		X(2)	X(3)	X	X			X	
Standby(1)						X		X(3)	X				X	
Extended Standby					X(2)	X	X(2)	X(3)	X	X			X	

Notes: 1. Only recommended with external crystal or resonator selected as clock source.
2. If Timer/Counter2 is running in asynchronous mode.
3. For INT7:4, only level interrupt.

Gestion du temps dans les programmes

- Boucles d'attentes actives
  - logicielle : for (i=0; i<delay;i++) asm volatile("nop");
  - matérielle : avec un compteur
- Scrutation d'un compteur permanent
  - Un compteur (time-stamp-counter) compte les cycles ou le temps en permanence et le programme peut consulter sa valeur pour connaître la durée écoulée.
- Timers
  - Un compteur programmable (dé)compte une valeur en cycles et lève un drapeau lorsqu'il atteind un seuil.
  - Ce drapeau peut être scruté ou interrompre le progamme si le processeur est interruptible pour exécuter une routine d'interruption (ISR)

Adéquation des techniques et des besoins

	delay attente active	time stamp counter	timers
Attente entre 2 actions	oui mais	oui pas précis	oui
Action périodique	non	oui pas précis	oui
Séquences d'actions temporelles	oui	oui pas précis	oui
Action avant échéance	non	oui pas précis	oui
Action après événement	non	oui pas précis	oui

Il n'y a pas de solution universelle, la bonne solution dépend du besoin.

API Arduino basique

Attente active : le programme n'avance plus mais le processeur est toujours sensible aux interruptions.

- delay(unsigned long ms)
  - 1ms à 4Gi ms (= 50 jours)
- delayMicroseconds(unsigned int us)
  - 3 µs à 16883µs (= 16ms)

Time Stamps (utilise le timer0 interne)

- unsigned long millis()
  - retourne le nombre de ms écoulées depuis le démarrage (max 50j)
- unsigned long micros()
  - retourne le nombre de µs écoulées depuis le démarrage (max 1h11m)

API Arduino basique exemples

```
int led = 13;

void setup() {
     pinMode(led, OUTPUT);
}

void loop() {
     digitalWrite(led, HIGH);
     delay(1000);
     digitalWrite(led, LOW);
     delay(1000);
}
```

```
int led = 13;
unsigned long prevMillis = 0;
int etatLed = LOW;

void setup() {
     pinMode(led, OUTPUT);
}

void loop() {
     unsigned long currMillis = millis();
     if (currMillis - prevMillis > 1000) {
          etatLed = (etatLed==LOW) ? HIGH:LOW;
          digitalWrite(led, etatLed);
     }
}
```

Comment faire clignoter deux leds

présentation Timers de l'ATmega

- L'ATmega2560 possède 6 timers programmables
  - 2 timers 8 bits (0 à 255)
  - 4 timers 16 bits (0 à 65355)
- Le timer0 est utilisé par Arduino (delays &co), les autres timers sont utilisé par des bibliothèques spécifiques.
- Un timer peut compter l'horloge interne ou un signal d'une broche.
- Lorsque le timer atteint sa valeur max ou un seuil défini, il peut
  - lever une interruption
  - changer l'état d'une broche
  - se réarmer
- Pour augmenter la plage de comptage, les timers sont
  - précédés de prédiviseur (prescaler) ou postdiviseur (postscaler)

http://www.atmel.com/images/doc2549.pdf page 118...

Programmation des timers

Il n'y a pas d'API Arduino programmer directement les timers, il faut écrire dans les registres de contrôle de ces périphériques pour obtenir la fonction souhaitée.

Toutefois, certaines bibliothèques les utilise et nous les verrons ultérieurement:

- analogWrite(pin, value)
- tone(pin, freq_hz,duration_ms)
- fonctions de Servo (moteurs pour positionnement angulaire)

Interruptions externes

Nous aurrons l'occasion d'utiliser les interruptions. Pour commencer, il est possible de demander l'exécution d'une fonction lorsqu'une broche subit un événement.

- attachInterrup(interrupt, function, mode)
  - pour l'ATmega2560 : int 0 = pin 2, 1→3, 2→21, 3→20, 4→19, 5→18
- detachInterrupt(interrupt)

Masquage des interruptions

- interrupt()
- noInterrupt()