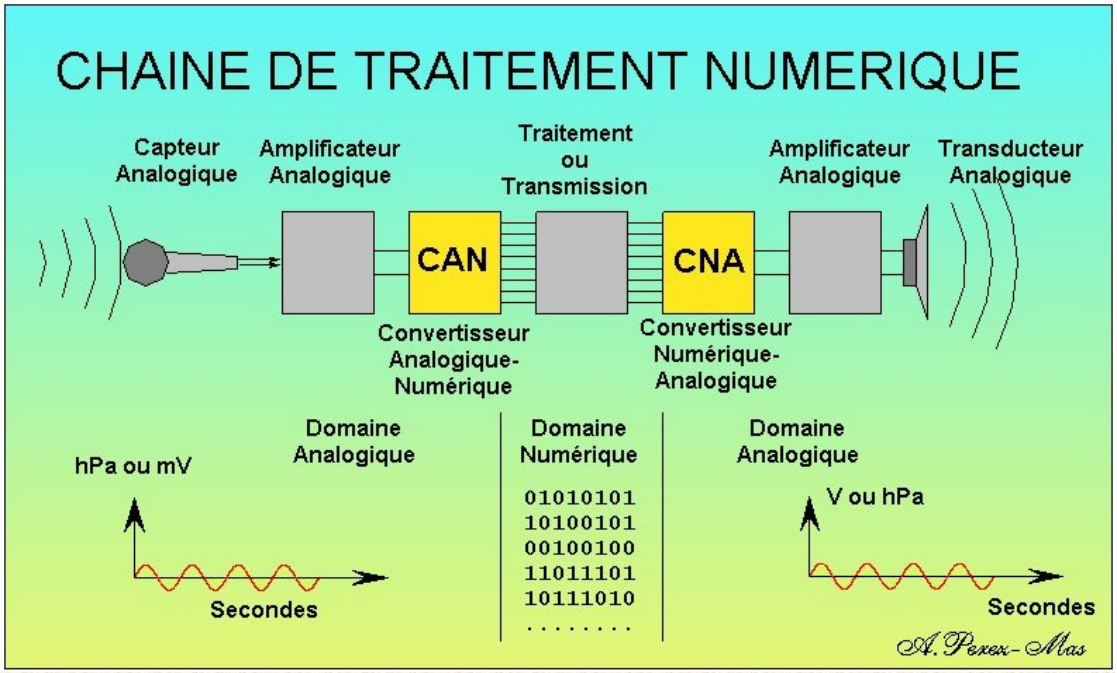
Microcontrôleurs

Chaîne de traitement numérique

Les acteurs

Le micro-contrôleur

- Manipule des valeurs numériques issus de capteurs divers
  - Objectif : Fournir un résultat à l'aide d'un algorithme de traitement ou de commande
- Interagit avec le monde extérieur
  - Convertisseur Analogique-Numérique
  - Convertisseur Numérique-Analogique



Le Convertisseur Analogique Numérique

- En anglais : ADC (Digital to Analog Converter)
- Convertit une tension analogique fournie par un capteur en un mot numérique compréhensible par le microcontrôleur
- Paramètres de l'ADC
  - Résolution, Vitesse de conversion, linéarité...
- L'ADC est intégré en général à l'intérieur du micro-contrôleur
  - Atmega 328p : mot de 10 bits, temps de conversion min 13us

Le Convertisseur Numérique Analogique

- En anglais : DAC (Digital to Analog Converter)
- Convertit un mot numérique fournit par le microcontrôleur en une tension analogique envoyée vers un transducteur
- Paramètres du CNA
  - Résolution, Vitesse de convertion, linéarité...
- Le DAC n'est pas forcément intégré en général à l'intérieur du micro-contrôleur
  - Atmega 328p : PAS de DAC => solution : utilisation d'une PWM

- **Les capteurs**
  - **Capteurs numériques**
    - **Fournissent une valeur numérique au micro via un bus de communication (I2C, SPI, RS232..)**
    - **L'utilisation d'un ADC est inutile dans ce cas**
  - **Capteurs analogiques**
    - **La sortie du capteur varie continûment en fonction de la grandeur physique captée**
    - **Un bloc de conditionnement du signal est souvent nécessaire entre le capteur et le micro**
      - **Exemple : Amplificateur de tension, Amplificateur transimpédance**

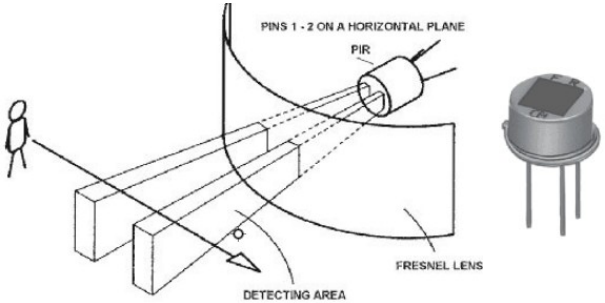
Jauge de contraite

La résistance de sortie varie

en fonction la force exercée

Capteur d'oxygène

Le courant de sortie varie en fonction du taux d'oxygène



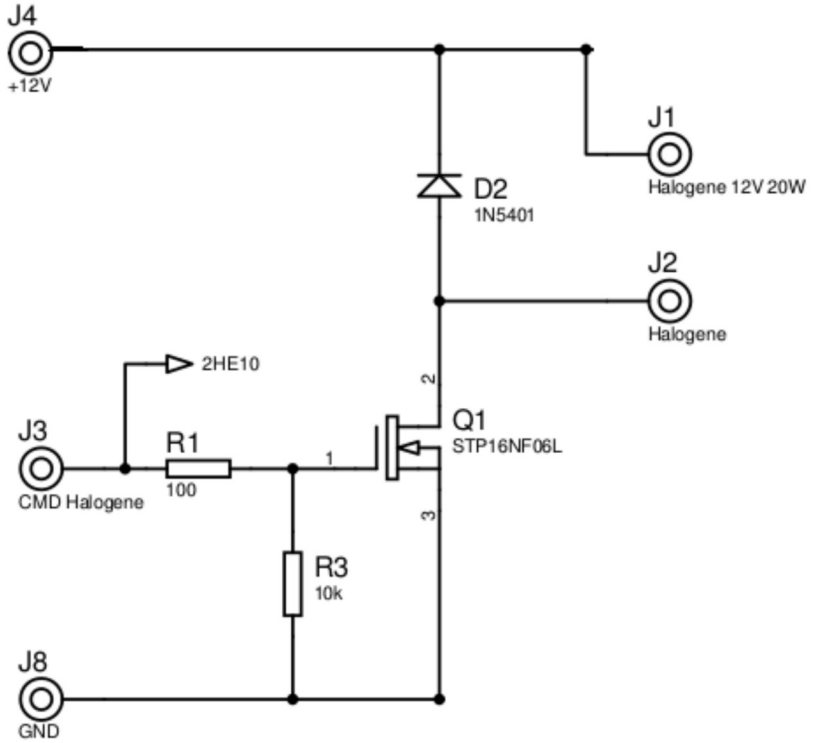
Capteur pyrosenseur

Le courant de sortie varie en fonction de la variation différentielle de température

Transducteur et Actionneurs

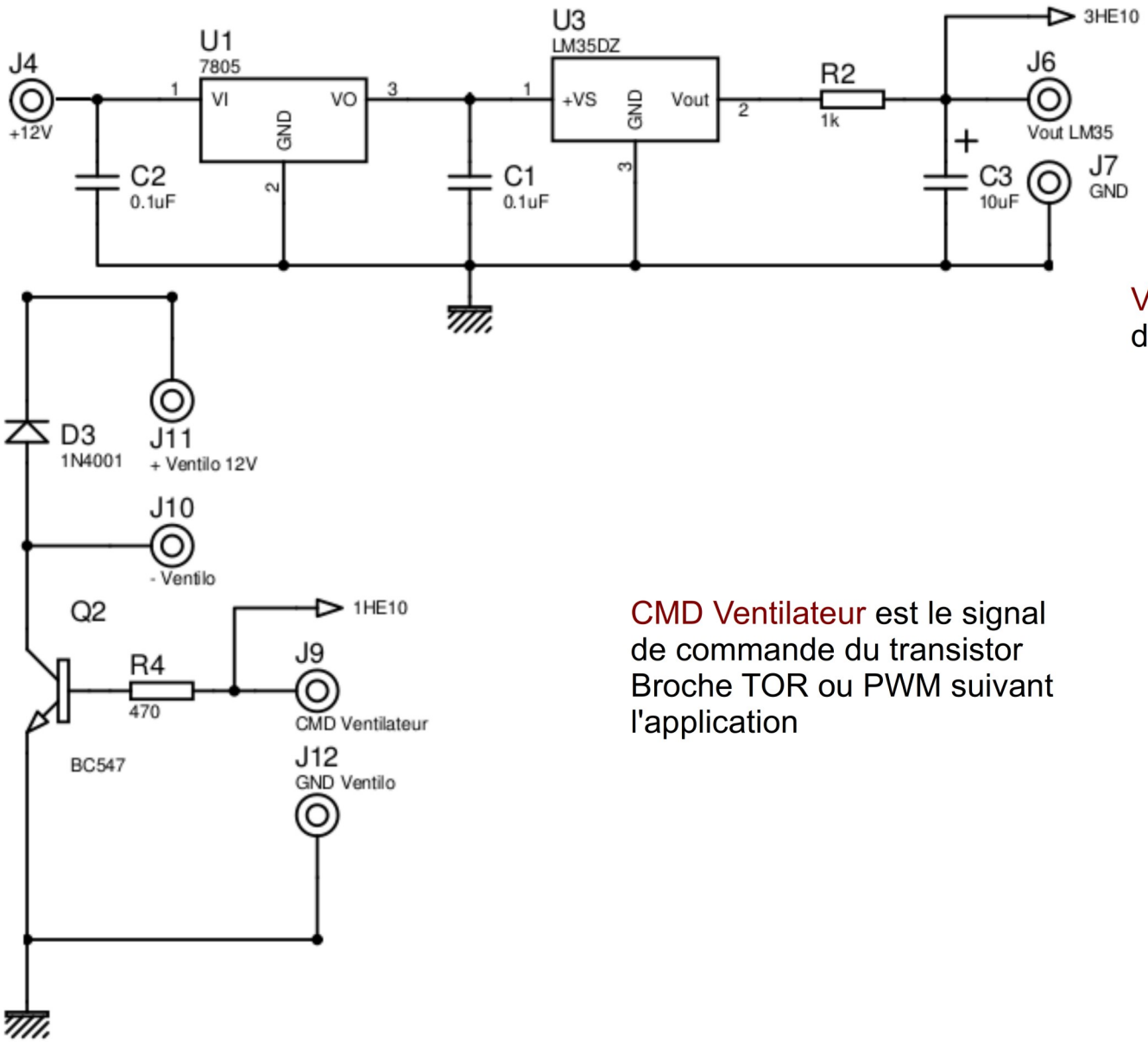
- Transforme les ordres du micro en une action physique
  - Tourner, déplacer, éclairer ...
- Les ordres du micro sont :
  - Un mot issu d'un DAC
  - Des signaux TOR (Tout Ou Rien)
  - Des signaux PWM

Notre Maquette



CMD Halogene est le signal de commande du transistor
Broche TOR ou PWM suivant l'application

Chaîne de traitement numérique



Vout LM35 est le signal image de la température à convertir.

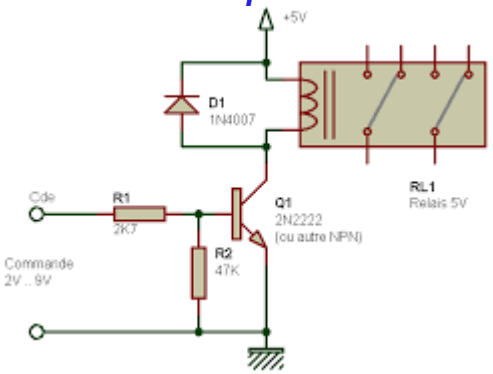
CMD Ventilateur est le signal de commande du transistor Broche TOR ou PWM suivant l'application

Transducteur et Actionneurs (2)

- Un bloc d'interfaçage est en général nécessaire entre l'actionneur et le micro
  - Adaptation en tension , en courant , en puissance
    - Exemple : hacheur, ampli de tension ….

Commande d'un relais

Transistor en bloqué/saturé

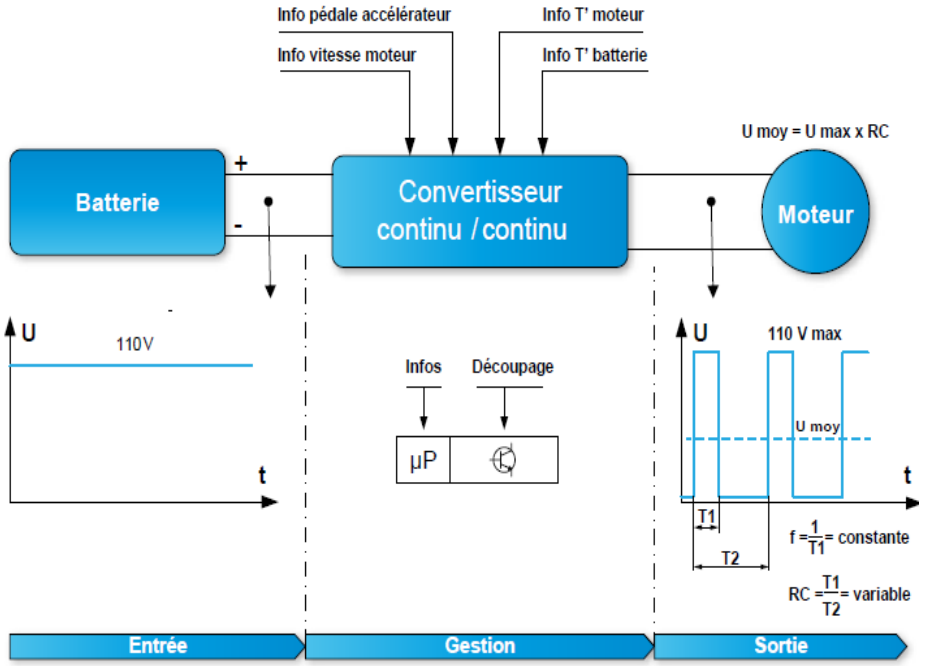


Commande d'un haut-parleur

Amplificateur classe AB

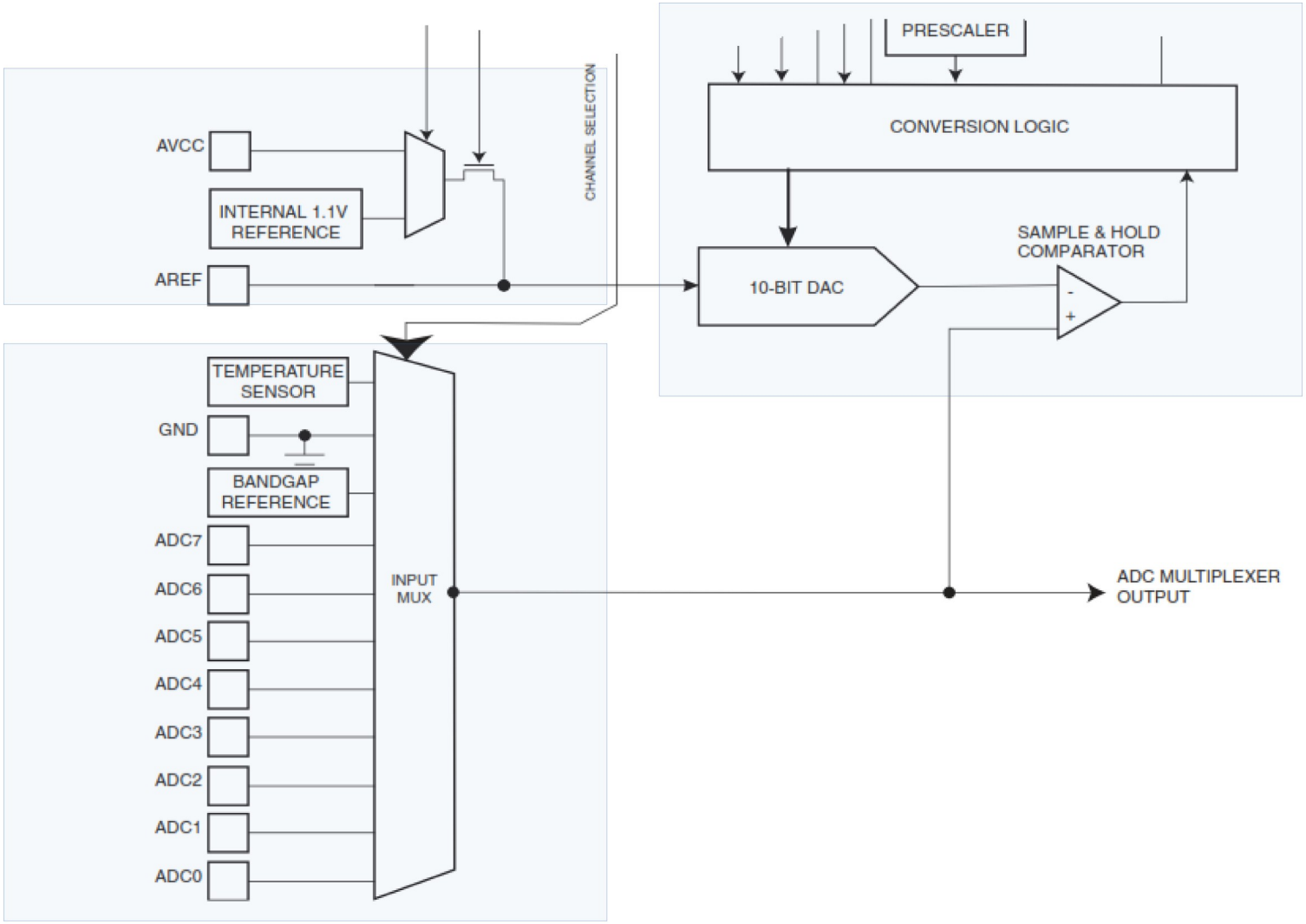
Commande d'un moteur

Hacheur 4 quadrants



Hardware : notre microcontrôleur ATmega328

Synoptique du CAN de l'ATmega328



- **Software**
  - **Utilisation de l'API arduino pour une mise en œuvre rapide !**
  - **Les étapes :**
    - **Initialisation :**
      - **voie à convertir, config voie en entrée, choix tension de référence**
    - **Conversion et lecture de données**
  - **Exemple 1 :**

Analog I/O

- analogReference()
- analogRead()
- analogWrite() - *PWM*

```
10  #define adcInputPin 14
11  #define tempoVal 1000
12
13  // the setup function runs once when you press reset or power the board
14
15  void setup() {
16      pinMode(adcInputPin, INPUT); // not mandatory - pin are in input state by default
17      analogReference(DEFAULT); // selecting ADC voltage reference: DEFAULT = 5V on arduino Uno
18      Serial.begin(57600); // // initialize serial port
19  }
20
21  // the loop function runs over and over again forever
22
23  void loop() {
24      unsigned int value; // 10bits converted value stored in 16 bit variable
25      value = analogRead(adcInputPin); // adcInputPut can be replaced by A0 ( see pins_arduino.h)
26      Serial.print("val conv=");
27      Serial.println(value,HEX);
28      delay(tempoVal); // wait for a second
29  }
```

Le delay est bloquant
Mauvaise pratique (pour test uniquement)

- **Software**
  - **Utilisation de l'API arduino pour une mise en œuvre rapide !**
  - **Contrainte :**
    - **fréquence fixe de 490Hz sur broche 3,9 et 10**
    - **fréquence fixe de 980Hz sur broche 5 et 6**
  - **Exemple (non bloquant)**

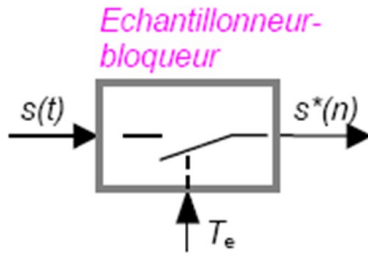
```
6   #include <Arduino.h>           // ne
7   extern HardwareSerial Serial; // ne
8
9   #define pwmPin 6
10  #define value1 255/4
11  #define value2 (3*4)*255
12  #define intervalChangePwmTask 2000
13  void doChangePwm();
14  // the setup function runs once whe
15  void setup() {
16      pinMode(pwmPin, OUTPUT); //
17  }
```

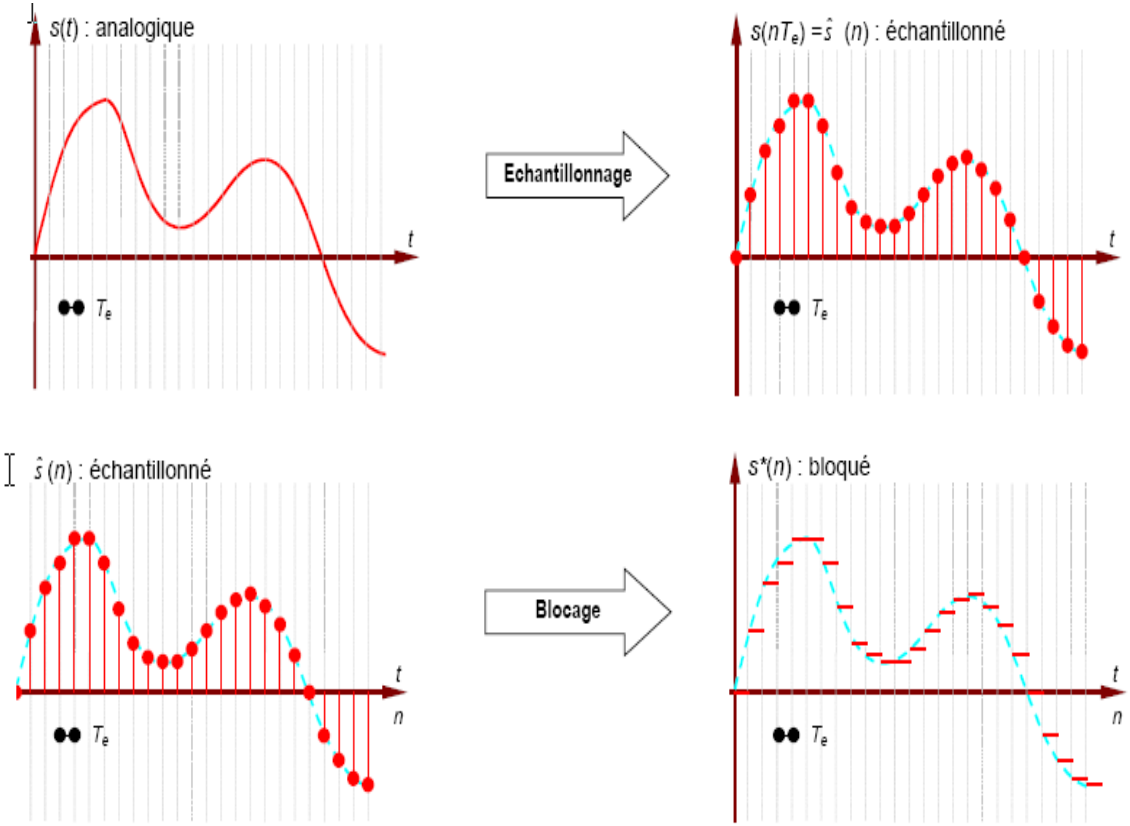
```
19  // global var section
20  unsigned long currentTime = 0;
21  unsigned long previousTimeChangePwmTask = 0;
22  // the loop function runs over and over again forever
23  void loop() {
24      currentTime = millis();
25      // TASK1: change PWM task
26      if ((currentTime - previousTimeChangePwmTask) > intervalChangePwmTask) {
27          doChangePwm();
28          previousTimeChangePwmTask = currentTime;
29      }
30  }
31
32  void doChangePwm() {
33      bool static valueState = false; // Avoid global var: Static is important
34      if (valueState == false) {
35          analogWrite(pwmPin, value1);
36          valueState = true;
37      } else {
38          analogWrite(pwmPin, value2);
39          valueState = false;
40      }
41  }
```

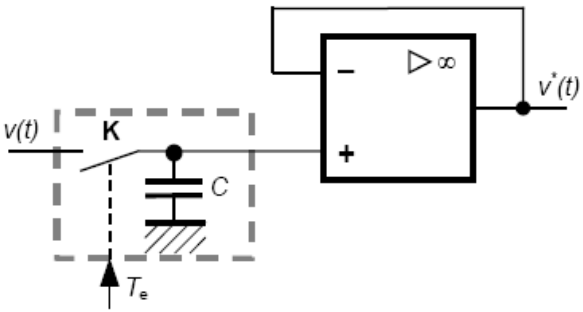
Les étapes clés d'une conversion analogique

L'échantillonnage et le blocage

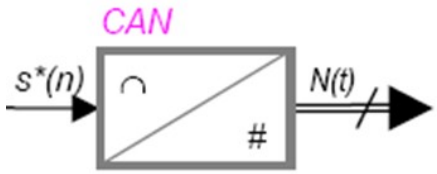
- Objectif : prélever la tension analogique et la maintenir le temps de la conversion
- Cet échantillonnage s'effectue en général à intervalles périodiques

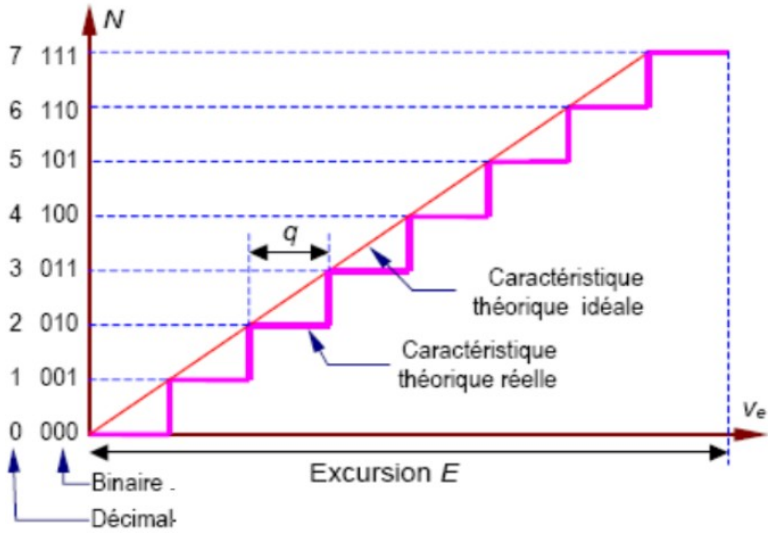






- la quantification
  - Objectif : transformer le niveau analogique en entrée en un mot numérique équivalent
  - Le can est caractérisé entre autres par :
    - sa fonction de transfert





    - sa résolution
      - C'est le nombre N de bits composant le mot
    - son quantum
      - incrément de tension d'entrée produisant une variation du résultat de une unité (soit une variation de 1LSB)
    - La pleine échelle
      - Full Scale Range : exercusion max du convertisseur
      - Pour exploiter au mieux le convertisseur la tension du capteur doit s'approcher le cette valeur

$$q = \frac{(V^{refH} - V^{refL})}{2^N}$$

$$FSR = (V^{refH} - V^{refL}) = N.q$$

$$V^{refL} \leqslant V_{capt} \leqslant (V^{refL} + N.q)$$

- **Gestion de la fréquence d'échantillonnage**
  - **Version non bloquante (sans timer)**

```
21  // global var section
22  unsigned long currentTime = 0;
23  unsigned long previousTimeTask = 0;
24  // the loop function runs over and over again forever
25  void loop() {
26      unsigned int value;
27
28      currentTime = millis();
29      // TASK1: convert value and send to serial com
30      if ((currentTime - previousTimeTask) > intervalAdcTask) {
31          previousTimeTask = currentTime;
32          value = analogRead(adcInputPin); // adcInputPut can be replaced by A0
33          Serial.print("val conv=");
34          Serial.println(value, HEX);
35      }
36  }
```

- **Déclenchement d'une conversion sur un événement**
  - **Exemple : conversion à chaque appui d'un bouton poussoir**

```
#include <Arduino.h>           //
extern HardwareSerial Serial; //

#define startButton 6
#define adcInputPin 14

#define intervalDebounceTask 20

void doActionStartButton(void);

// the setup function runs once
void setup() {
    pinMode(adcInputPin, INPUT);
    analogReference(DEFAULT); //
    pinMode(startButton, INPUT);
    Serial.begin(57600); // // i
}

```

```
// global var section
unsigned long currentTime = 0;
unsigned long previousTimeDebounceTask = 0;
boolean stateStartButton = HIGH, statePreviousStartButton;
// the loop function runs over and over again forever
void loop() {
    currentTime = millis();
    // TASK1: debounce task
    if ((currentTime - previousTimeDebounceTask) > intervalDebounceTask) {
        statePreviousStartButton = stateStartButton;
        stateStartButton = digitalRead(startButton);
        if ((stateStartButton == LOW)&&(statePreviousStartButton == HIGH)) {
            doActionStartButton();
        }
        previousTimeDebounceTask = currentTime;
    }
}

//
void doActionStartButton() {
    unsigned int value;
    value = analogRead(adcInputPin); // adcInputPut can be replaced by A0 ( 
    Serial.print("val conv=");
    Serial.println(value, HEX);
}
```

Mesure de température avec un LM35

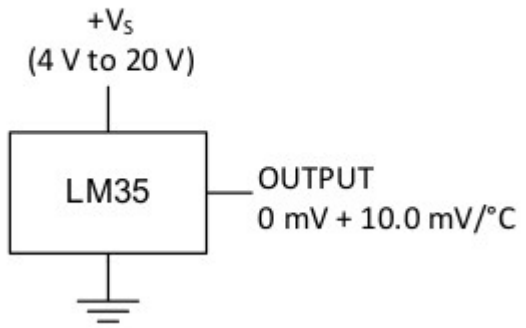
- On souhaite mesurer dans une plage de température de 0°C à 100°C
  - Calculer la résolution en température avec Vref+ = 5V

    Même calcul avec une référence interne de 1,1V

  - Quelle est l'instruction qui permet de changer cette référence ?

  - Choisir en fonction des caractèristiques du composant

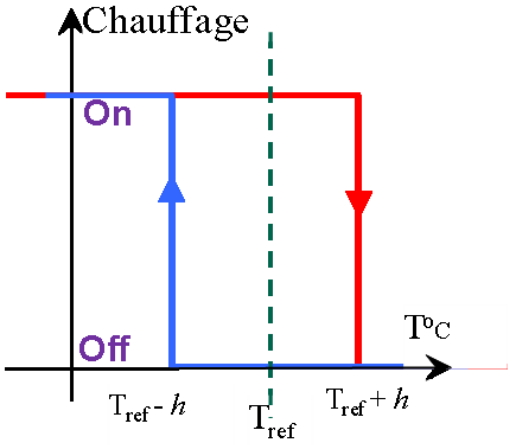
Basic Centigrade Temperature Sensor (2°C to 150°C)



1 Features

- Calibrated Directly in Celsius (Centigrade)
- Linear + 10-mV/°C Scale Factor
- 0.5°C Ensured Accuracy (at 25°C)
- Rated for Full −55°C to 150°C Range
- Suitable for Remote Applications
- Low-Cost Due to Wafer-Level Trimming
- Operates From 4 V to 30 V
- Less Than 60-μA Current Drain
- Low Self-Heating, 0.08°C in Still Air
- Non-Linearity Only ±¼°C Typical
- Low-Impedance Output, 0.1 Ω for 1-mA Load

- **commande tout ou rien**
  - **Principe**



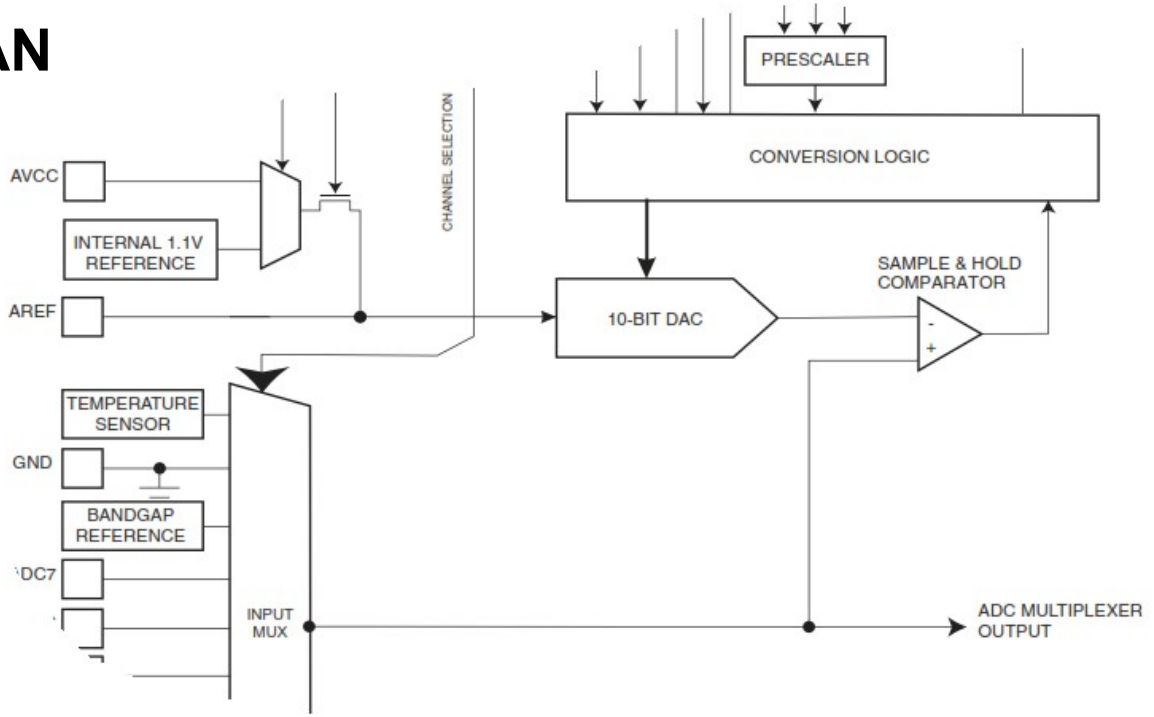
Modélisation logicielle par un grafcet à 2 états

À compléter

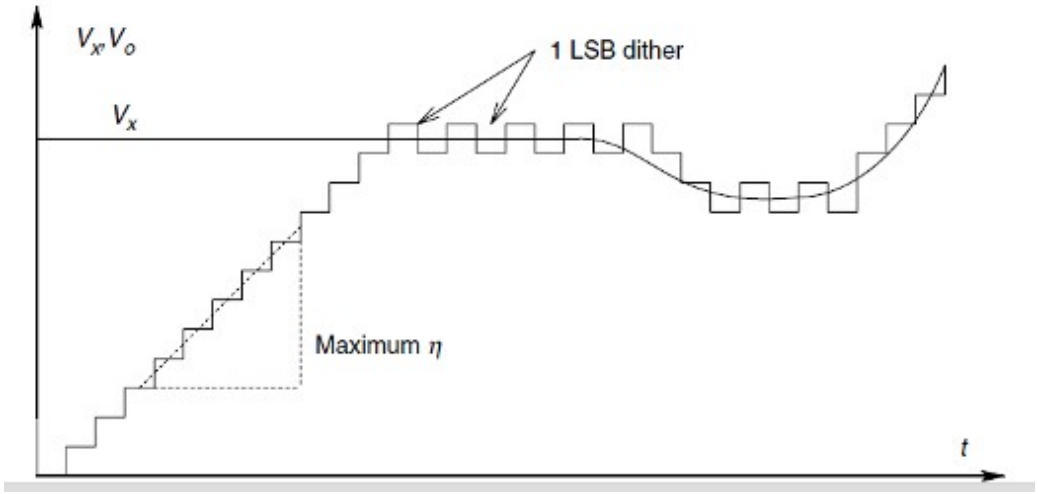
  - **Programmation**

```
33  void loop() {
34      byte currentTemperature = 20; //local var
35
36      currentTime = millis();
37      // TASK1: convert value, update grafcet , update output and send to serial com
38      if ((currentTime - previousTimeRegulationTask) > intervalRegulationTask) {
39
40          previousTimeRegulationTask = currentTime;
41          currentTemperature = temperatureAcquisition();
42          stateResistor = temperatureRegulation(currentTemperature, referenceTemperature);
43          digitalWrite(resistorOutputControl, stateResistor);
44          sendDataSerialCom(currentTemperature, referenceTemperature);
45      }
46
47      // TASK2: debounce task
48      if ((currentTime - previousTimeDebounceTask) > intervalDebounceTask) {
49          statePreviousPlusButton = statePlusButton;
50          statePlusButton = digitalRead(plusInputButton);
```

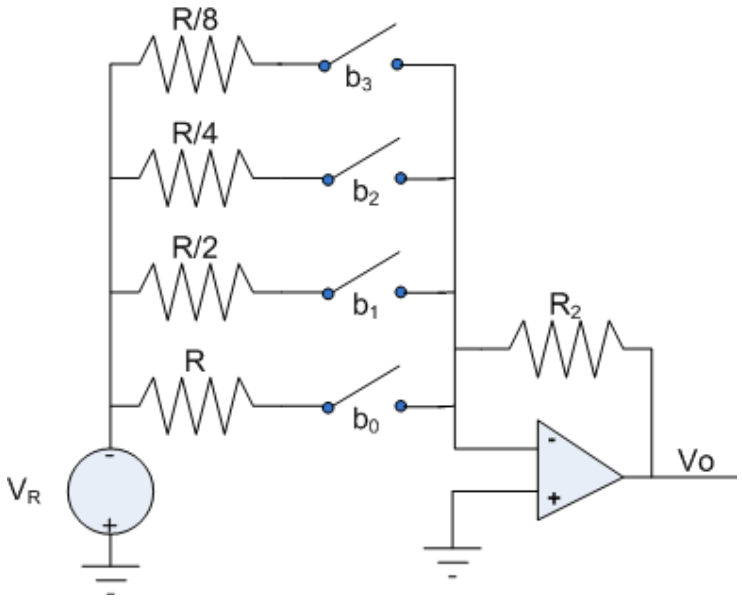
- **Fonctionnement d'un CAN**
  - **Cas de l'ATmega328**
    - **Convertisseur à pesée successives**



La tension du convertisseur numérique analogique est comparée en permanence à la tension d'entrée et le mot numérique est ajustée en conséquence

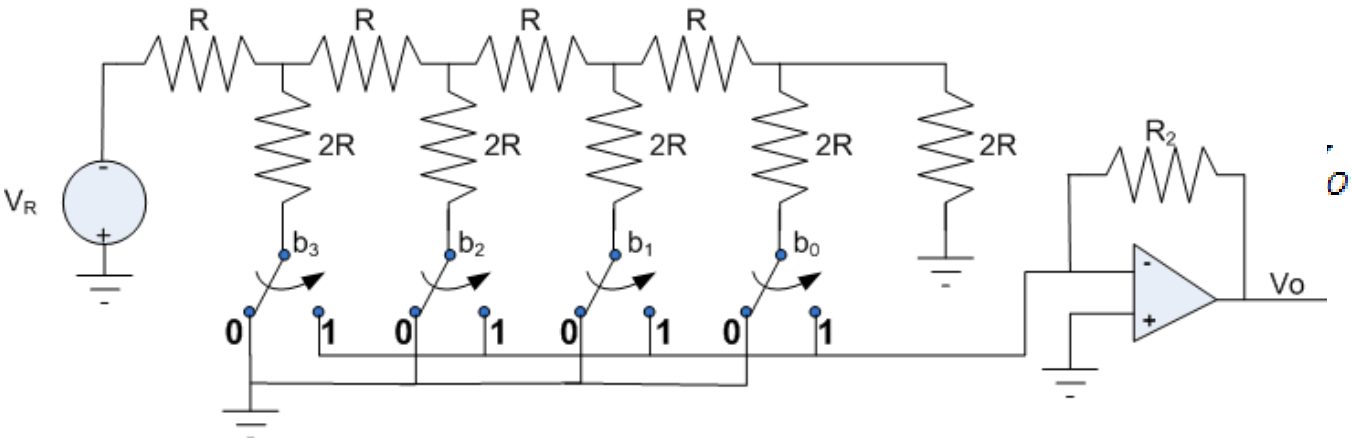


- **Pesée successive**
  - **Le CNA**



$$V_o = \frac{R_2}{R} V_R \left(b_0 + 2b_1 + 4b_2 + 8b_3 \right)$$

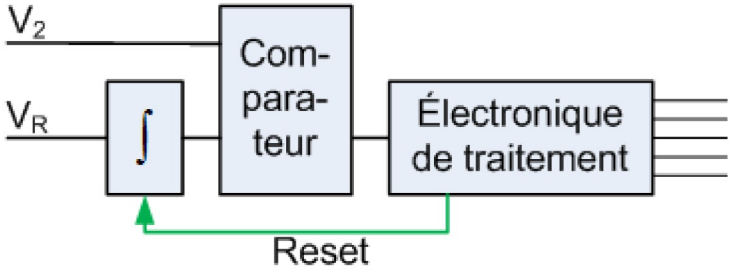
$$q = \frac{R_2}{R} V_R$$



$$V_{o\max} = \frac{R_2}{R} V_R \sum_{i=0}^{n-1} 2^i = q\left(2^n - 1\right)$$

Autres solutions technologiques

Convertisseur simple rampe

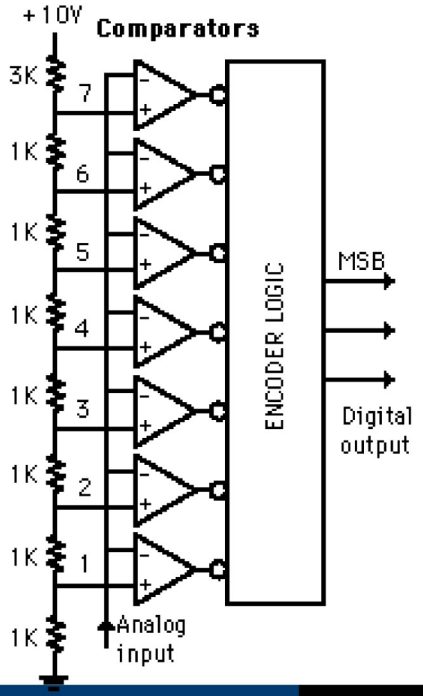


$$V_3 = \int_0^{\tau} V_R dt = V_R \tau$$

le CAN utilise un intégrateur qui reçoit le signal de référence V_R.

L'électronique de traitement comporte un compteur qui mesure le temps que mets V3 a atteindre V2 . La valeur du compteur reflète la valeur de V2 a un coefficient près qui dépend de C.
Un système à double rampe permet d'améliorer la rapidité et la précision de la mesure

Convertisseur Flash



La structure parallèle de ce type de convertisseur permet d'obtenir des temps de conversions des petits

Synthèse

- Le fréquence d'échantillonnage du cahier des charges est un des critères de choix de la technologie employée et donc du choix éventuel du microcontrôleur employé

Type	Erreur	Vitesse	Résolution
Simple rampe	Elevée	Faible (ms)	Moyenne à élevée (7 à 14 bits)
Double rampe	Faible	Faible (ms)	Elevée (10 à 18 bits)
Approximations successives	Moyenne 0,5 à 1 LSB	Moyenne (quelque 10 µS)	Moyenne à élevée (8 à 16 bits)
Flash	Moyenne 0,5 à 1 LSB	Elevée	Faible à élevée (4 à 10 bits)

Extrait datasheet ATmega328

- 10-bit Resolution
- 0.5 LSB Integral Non-linearity
- ± 2 LSB Absolute Accuracy
- 13 - 260µs Conversion Time
- Up to 76.9kSPS (Up to 15kSPS at Maximum Resolution)
- 6 Multiplexed Single Ended Input Channels

Programmation

- Utilisation des registres du micro pour accéder à toutes les fonctions du CAN !
- Se reporter à la doc pour avoir toutes les infos

ADCSRA – ADC Control and Status Register A

Bit	7	6	5	4	3	2	1	0	
(0x7A)	ADEN	ADSC	ADATE	ADIF	ADIE	ADPS2	ADPS1	ADPS0	ADCSRA
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	

ADEN ***ADC Enable*** Mise en marche du convertisseur avec la mise à 1 du bit, l'arrêt avec la mise à 0, la conversion en cours sera terminée.

ADSC ***ADC Start Conversion*** Lancement de la conversion de la voie sélectionnée (retourne à 0 en fin de conversion). En mode simple conversion il faut remettre à 1 à chaque nouvelle conversion. En mode libre, la première conversion dure 25 cycles puis les suivantes 15, il n'est pas nécessaire de remettre le bit à 1 à chaque conversion.

ADATE ***ADC Auto Trigger Enable*** La mis à 1 de ce bit permet de mettre en le convertisseur en fonction d'un déclencheur. La sélection du déclencheur est faite avec le bit **ADTS** du registre **SFIOR**.

ADIF ***ADC Interrupt Flag*** Passe à 1 une fois la conversion terminée et déclenche l'interruption si **ADIE** =1. Ce bit repasse automatiquement à 0 lors du traitement de la routine d'interruption.

ADIE ***ADC Interrupt Enable*** Validation de l'interruption **ADC**, déclenché lors du passage à 1 de **ADIF**.

ADSP2...ADSP0 ***Prescaler Select Bits*** Sélection du facteur de pré-division de l'horloge interne du convertisseur en fonction du quartz :

ADSP2	ADSP1	ADSP0	Facteur de division
0	0	0	2
0	0	1	2
0	1	0	4
0	1	1	8
1	0	0	16
1	0	1	32
1	1	0	64
1	1	1	128

fconvmax=200kHz mais en pratique possibilté d'overclocker sans perte de résolution

Bit	7	6	5	4	3	2	1	0	
(0x7B)	–	ACME	–	–	–	ADTS2	ADTS1	ADTS0	ADCSRB
Read/Write	R	R/W	R	R	R	R/W	R/W	R/W	
Initial Value	0	0	0	0	0	0	0	0	

En combinaison avec ADATE=1 les bits ADTS permettent de choisir la source de déclenchement d'un départ de conversion.
Par exemple, pour obtenir une fréquence d'échantillonnage fixe, la conversion sera déclenché sur le timer 1 -attention le timer 0 est utilisé pour la fonction millis() -

Table 24-6. ADC Auto Trigger Source Selections

ADTS2	ADTS1	ADTS0	Trigger Source
0	0	0	Free Running mode
0	0	1	Analog Comparator
0	1	0	External Interrupt Request 0
0	1	1	Timer/Counter0 Compare Match A
1	0	0	Timer/Counter0 Overflow
1	0	1	Timer/Counter1 Compare Match B
1	1	0	Timer/Counter1 Overflow
1	1	1	Timer/Counter1 Capture Event

Exemple2 : convertisseur déclenché par timer

INITIALISATION

```
void setup() {
    cli(); // stop IRQ
    pinMode(led, OUTPUT);
    initTimer1();
    initADC();
    Serial.begin(57600); // // initialize serial port
    sei(); // enable IRQ
}
```

```
73  void initTimer1() {
74      TCNT1 = 0;
75      TCCR1A = 0; // waveform generator not activated: OC1A et OCBB disconnect
76      OCR1B = 0x3D09; // compare match register B
77      TCCR1B = 0; // normal mode
78      TCCR1B |= ((1 << CS12) | (1 << CS10)); // prescale= 1/1024 and normal mode
79      TIMSK1 |= (1 << OCIE1B); // enable IRQ  timer 1 output compare B
80  }
81
82  void initADC() {
83      pinMode(adcInputPin, INPUT); // not mandatory - pin are in input state by default
84      ADMUX = 0b01000000; // REFS=b10 => AVCC as voltage reference, ADLAR=0 RIGHT ADJUST  p248 datasheet
85      ADCSRA = 0b10101111; // ADEN=1, ADATE=1=AutoTrigger ,ADIE=1=InterruptADC Enable, APCS= /128 prescale, ADI  p249 datasheet
86      ADCSRB = 0b00000101; // ADC autostart when timer1 compare Match B is true
87  }
```

ROUTINE d'INTERRUPTION

```
32  // ADC IRQ sub routine
33  // data is ready => read data and activate flagIT (deferred procedure)
34  // DO NOT: do not start a new conv since adc is timer triggered
35  // DO NOT: do not wait for data conv since we are in IRQ (conv is FINISHED=> data is ready!)
36  // DO NOT: do not clear ADIF since ADIF is cleared by hardware when executing the corresponding interrupt handling vector.
37  ISR(ADC_vect) {
38      adcValue = ADCL + (ADCH << 8); // you must read ADCL first
39      flagIT = true;
40  }
41
42  ISR(TIMER1_COMPB_vect) {
43      static bool stateLed = false;
44      stateLed = !stateLed;
45      digitalWrite(led, stateLed);
46      TCNT1 = 0;   //mandatory to obtain correct timing (in normal mode max timer value is 0xFFFF)
47                   // a another solution is to program timer in CTC mode with OCR1A=OCR1B
48                   // in this case timer reset counter is automatic
49  }
```

TÂCHES DE FOND

```
64  void loop() {
65      if (flagIT == true) {
66          Serial.print("val conv=");
67          Serial.println(adcValue, HEX);
68          flagIT = false;
69      }
70
71  }
```

Traitement différé de la valeur convertie
Communication au travers d'une variable globale volatile

```
volatile bool flagIT = false;
```

Schéma global